

**Various extensions in the theory of dynamic materials with a  
specific focus on the checkerboard geometry**

by

William C. Sanguinet

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE


In partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Mathematical Sciences

by



---

March 2017

APPROVED:



---

Professor Konstantin Lurie, Advisor  
Department of Mathematical Sciences  
Worcester Polytechnic Institute



---

Professor Suzanne L. Weekes, Advisor  
Department of Mathematical Sciences  
Worcester Polytechnic Institute



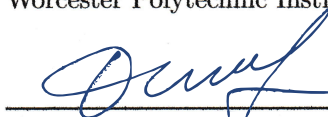
---

Professor Mayer Humi  
Department of Mathematical Sciences  
Worcester Polytechnic Institute



---

Professor Homer F. Walker  
Department of Mathematical Sciences  
Worcester Polytechnic Institute



---

Professor Daniel Onofrei  
Department of Mathematics,  
University of Houston

## Acknowledgements

I want to thank the many people who have helped me during the rewarding experience that has culminated in the completion of my PhD thesis. I would like to thank Professors Konstantin A. Lurie and Suzanne L. Weekes for their unending support, motivation, and time. I would not be the person I am today without the consistent help and friendship I have received from you both. I would like to thank Professor Vadim Yakovlev and Professor Mihhail Berezovski for their help and guidance. Specifically, I wish to thank Mihhail, his father Professor Arkadi Berezovski, and Professor Jüri Engelbrecht, for allowing me to spend time learning and researching at the Institute of Cybernetics at Tallinn University of Technology in Estonia. I want to thank the other members of my PhD committee: Professor Daniel Onofrei, Professor Homer Walker, and Professor Mayer Humi, the discussions I have had with you over the years I have known you have developed a lasting curiosity and appreciation for applied mathematics and physics. I would also like to thank other faculty (at WPI and elsewhere) for helping me grow first as a student and subsequently, as a mathematical researcher; specifically, Silvia Jimenez, Burt Tilley, Roger Lui, Luca Capogna, Bogdan Vernescu, Darko Volkov, Brigitta Servatius, Peter Christopher, Chris Larsen, John Goulet, Michael Johnson, Sarah Olson, Joseph Fehribach, Joseph Petrucci, Umberto Mosco, Bill Martin, L. Ramdas Ram-Mohan, Nancy Burnham, Frank Dick, Germano Iannacchione, Alex Zozulya, Mark Siemaszko, and George Whittemore. You have all helped nurture my the passion for mathematics, physics, and science in some way, and for that, I am truly grateful.

I want to thank my family and friends who have encouraged and supported me these past few years, specifically, the various people who have been unfortunate enough to share an office with me in Stratton Hall. specifically, Grigor Nika, Erin Kiley, Nguyenho Ho, Yiqing Li, Josh Plasse, Marisa Zemsky, Pan Liu, etc. I also want to extend a special thank you to the Math Dept staff, Ellen Mackin, Michael Malone, Rhonda Podell, and Deborah Sleith, you have helped me in many ways.

Lastly, I want to extend an extra special thank you going to my family. Specifically, to my parents, my cousins, and my little brother. Remember to never stop chasing your dreams. Over the years, I have found strength and guidance in the following passage, it means a good deal to me, and I hope you can appreciate its significance:

“Do not lose your knowledge that man’s proper estate is an upright posture, an intransigent mind and a step that travels unlimited roads. Do not let your fire go out, spark by irreplaceable spark, in the hopeless swamps of the approximate, the not-quite, the not-yet, the not-at-all. Do not let the hero in your soul perish, in lonely frustration for the life you deserved, but have never been able to reach. Check your road and the nature of your battle. The world you desired can be won, it exists, it is real, it is possible, it’s yours.” - John Galt, Atlas Shrugged



## Foreword

The first time I remember experiencing coordinated wave motion was as a child. Specifically, I remember the waves crashing onto a beach in Rhode Island. Watching each swell displaying a simultaneously symmetrical yet seemingly random motion was mesmerizing. To this day, I am transfixed by this motion, by even the smallest ripple, which serves as a testament to the mathematical symmetries governing it. How waves extend from an initial disturbance, simultaneously coordinated and erratic, behaving vastly different under various circumstances. I didn't realize it then, but after some time, it became apparent to me that the mathematics governing this motion underlies many of the most fundamental physical systems. It is present in every aspect of physics, from the quantum level up to the relativistic level of planets and even galaxies. Similar to this range of scale, the complexity of the mathematical description of wave motion ranges from the very simple to the very complex. This motion can be governed by a single linear equation or by systems of many complex non-linear equations. Regardless, it is astounding is that in each case these mathematical equations come directly from the physical laws governing a particular situation.

Of particular interest to me is control of this wave motion. Specifically, the question of whether it is possible to control the progression of a disturbance as it travels through a given system. A wave disturbance is a dynamic phenomenon, moving through space, and it is because of this that full control is only possible if one can specify certain material properties at a specified point simultaneously in space and time. A definite means of obtaining this control is by prescribing the wave-speed throughout a given material. This control must be achieved through a specified domain at each specific point and at each specific instant of time. An interesting question is what happens when one decides to enact this spatial-temporal control. Specifically, we will be controlling the material property pattern. This property pattern is the distribution of material properties in spacetime. How exactly does this pattern change the fundamental behavior of the propagating disturbance? Are there methods of control which produce novel and desirable affects on the wave? Can we pump or pull energy in to or out of a traveling wave? Is it possible to screen or guide disturbances as they travel through some medium?

This manuscript is an attempt to answer and expand on some of these questions. It contains a variety of results which summarize and extend the existing field of knowledge surrounding such materials. The text is separated into several self-contained portions. Chapter 1 is an introduction to the mathematical models and methods that will be used and a brief review of the various material geometries previously studied and results previously obtained for DM. Chapter 2 is an detailed summary of the results published in the paper [32] along with related results discovered after publication. Chapter 3 is a detailed exposition of new results pertaining

to wave propagation through a rectangular checkerboard geometry under the relaxation of assumptions from the papers [26] and [27]. Chapter 4 is a discussion of results on the energy of waves traveling through the material geometries discussed in Chapter 3. Chapter 5 is a conclusion and contains a discussion of new ideas I have for future research and extension of the theory of linear DM to the theory of non-linear DM.

# Contents

<b>Acknowledgements</b>	<b>2</b>
<b>Foreword</b>	<b>3</b>
<b>1 Introduction</b>	<b>10</b>
1.1 Mathematical and Physical Preliminaries . . . . .	10
1.1.1 Electromagnetics . . . . .	11
1.1.2 Elasticity . . . . .	14
1.2 Finite Volume Methods . . . . .	15
1.3 Previous work on Dynamic Materials . . . . .	17
1.3.1 Characterization of Dynamic Materials . . . . .	18
1.3.2 Laminar Material Geometry . . . . .	19
1.3.3 Activated Left-Hand Medium . . . . .	24
1.3.4 Checkerboard Material Geometry . . . . .	24
<b>2 Dilatation and Shear Wave Propagation in Dynamic Checkerboard</b>	<b>30</b>
2.1 Introduction . . . . .	31
2.2 Background . . . . .	32
2.3 Results . . . . .	32
2.4 Conclusion . . . . .	35
<b>3 Non-perfect Spatial Temporal Checkerboard</b>	<b>39</b>
3.1 Functionally Graded Dynamic Material . . . . .	40
3.1.1 Wave Route Calculation for FG material . . . . .	40
3.2 Linear FG Material . . . . .	47
3.2.1 Characteristic in Linear Region . . . . .	50
3.2.2 Inequality Constraints . . . . .	54
3.3 Conclusion . . . . .	65
<b>4 Energy Accumulation in Checkerboard Generalizations</b>	<b>69</b>
4.1 Numerical Procedure . . . . .	70
4.1.1 Basic Numerical Method . . . . .	71
4.1.2 Brute-Force Optimization . . . . .	75
4.2 Energy Accumulation in a Checkerboard . . . . .	78

4.2.1	Energy in a Sharp Checkerboard . . . . .	78
4.2.2	Energy Accumulation in a FG Checkerboard . . . . .	79
4.2.3	Energy Accumulation in a Checkerboard with mismatched wave impedances . . . . .	84
4.2.4	Conclusion . . . . .	85
<b>5</b>	<b>Conclusion</b>	<b>86</b>
5.1	Summary . . . . .	86
5.2	Future Work . . . . .	87
5.2.1	Nonlinear Dynamic Materials . . . . .	87
5.2.2	Optimization for Specific Geometries . . . . .	92
	<b>Appendices</b>	<b>94</b>
<b>A</b>	<b>Code</b>	<b>95</b>
A.1	Characteristic Plotter . . . . .	95
A.2	Inequality Plotter . . . . .	111
A.3	Clawpack Code . . . . .	133
A.3.1	Functions.f . . . . .	133
A.3.2	setrun.py . . . . .	141
A.3.3	Main Script . . . . .	150
A.3.4	SGE Script . . . . .	151
A.3.5	setrun.py . . . . .	152
	<b>Bibliography</b>	<b>162</b>

# List of Figures

1.1	Typical FVM cell for $n = 3$ . . . . .	16
1.2	Laminar material geometry for $m = \frac{2}{5}$ . . . . .	20
1.3	Screening effect in a laminar geometry . . . . .	23
1.4	Checkerboard structure in spacetime. Material 1 is colored in red and material 2 is colored in blue. . . . .	25
1.5	Checkerboard structure in spacetime. Material 2 is colored in yellow and material 1 is colored in blue. . . . .	26
1.6	Focusing effect in a checkerboard geometry for disturbance initially supported over one period. . . . .	28
1.7	Focusing effect in a checkerboard geometry for disturbance initially supported over one period. . . . .	29
2.1	Simultaneous focusing in both dilatation and shear waves along with the mapping onto the torus. . . . .	36
2.2	Plot of the average phase velocity of a characteristic after a large number of periods as a function of $m$ and $n$ . This demonstrates clearly the plateau region for each type of wave. . . . .	37
2.3	Plot of the average phase velocity of a characteristic after a large number of periods as a function of $m$ and $n$ . This demonstrates clearly the plateau region for each type of wave. . . . .	38
2.4	Plot of the average phase velocity as a function of phase velocity $a_1$ and $a_2$ . This demonstrates clearly the plateau region for each type of wave. . . . .	38
3.1	Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over one period. . . . .	42
3.2	Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over ten periods. . . . .	43
3.3	Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over 3 later periods. This shows that convergence persists up to a certain point. . . . .	44
3.4	Comparison of characteristics traveling through a perfect checkerboard and a FG checkerboard over 3 later periods. This image shows that convergence disappears around .55 . . . . .	45

3.5	Oscillatory phenomena immediately after convergence is killed by smoothing. . . . .	46
3.6	Example of smoothing in only one variable. . . . .	46
3.7	Linear FG Checkerboard over one spatial-temporal period. . . . .	48
3.8	Specific type of limit cycle studied. . . . .	51
3.9	Temporally Linear Transition from Material 1 to Material 2 . . . . .	52
3.10	Spatially Linear Transition from Material 1 to Material 2 . . . . .	54
3.11	Plateau for certain values of material parameters. . . . .	61
3.12	Existence region for plateau region in $p$ and $q$ space. . . . .	64
3.13	Characteristic behavior after smoothing. . . . .	66
3.14	Characteristic behavior after 10 periods for each case. Limit cycle behavior is observed, however, only the first two figures show proper $\mathcal{C}_{z_0}^{p,q}$ behavior. . . . .	67
4.1	Comparison of 3 levels of AMR patches at time .5 . . . . .	72
4.2	$E(t)$ in a sharp checkerboard for $a_1 = 0.6$ and $a_2 = 1.1$ using 6 AMR levels and 1000 initial grid cells . . . . .	76
4.3	Energy evolution in a sharp checkerboard for $a_1 = .6$ and $a_2 = 1.1$ using 5 AMR levels and 3000 initial grid cells. . . . .	77
4.4	Energy ratio between periods as a function of $m$ and $n$ . . . . .	79
4.5	$m$ - $n$ Plateau zone for parameters $a1=.55$ and $a2=1.1$ . . . . .	80
4.6	Plot showing the evolution of energy growth over time for 4 different combinations of $m, n$ . In each case, we have apparent energy growth. . . . .	81
4.7	Plot showing the evolution of energy growth over time for 3 different combinations of $m, n$ . In each case, there is no energy growth. . . . .	82
4.8	Plot showing the evolution of energy growth over time for several different values for the smoothing parameter. . . . .	83
4.9	Plot showing the evolution of energy growth over time for several different impedance mismatch. In each case, there is exponential energy growth. . . . .	84

# List of Tables

3.1	Coefficients of linear inequality constraints. . . . .	58
3.2	Coefficients of 6 linear inequality constraints, $f$ , $g$ , and $h$ are given by equations (3.38), (3.39), and (3.40), respectively. . . . .	59
3.3	Partial derivative of plateau boundary line equation with respect to smoothing parameters $p$ and $q$ . . . . .	62

# Chapter 1

## Introduction

The following chapter will serve as an introduction to the topic of *dynamic metamaterials* (DMs). It is by no means exhaustive, and I sincerely apologize for any relevant omission. In Section 1.1, I present a discussion of the mathematical framework used to pose problems in dynamic materials. Section 1.2 contains a discussion of some very useful methods for numerically solving systems of conservation laws discussed in the previous section. Section 1.3 is a review of previously obtained results on DMs.

### 1.1 Mathematical and Physical Preliminaries

A natural starting place for this thesis is a discussion of systems of hyperbolic conservation laws. Physically, these come naturally from statements of conservation found in nature and take the following mathematical form,

$$\mathbf{q}_{,t} + \mathbf{f}(\mathbf{q}; \mathbf{r}, t)_{,x} + \mathbf{g}(\mathbf{q}; \mathbf{r}, t)_{,y} + \mathbf{h}(\mathbf{q}; \mathbf{r}, t)_{,z} = \psi(\mathbf{q}; \mathbf{r}, \mathbf{t}), \quad (1.1)$$

where  $\mathbf{q}(\mathbf{r}, t)$  is a vector valued function of position  $\mathbf{r} \in \mathbb{R}^3$  and time  $t \in \mathbb{R}^+$ , the entries of which represent certain physical quantities being conserved,  $\mathbf{f}$ ,  $\mathbf{g}$ , and  $\mathbf{h}$  are vector valued functions that govern the flux of the relevant physical quantities (a.k.a., “flux”- functions), and  $\psi$  is a vector valued source term that governs generation or loss of a particular physical quantity. It is important to note that with DMs we allow for the possibility of having both the “flux”-functions and the source term simultaneously depend explicitly on space and time, this dependence is crucial to controlling the behavior of propagating disturbances. For the examples discussed in this work, this dependence will be by means of the material property distribution in space-time. If the flux functions do not depend on  $\mathbf{q}$  then Equation (1.1) is a linear system of conservation laws. This level of generality of Equation (1.1) will not always be necessary, however, in preparation for extensions and future research, we allow for a flux function of this form (i.e., for non-linearity).



Equation (1.1) presents a differential expression of these laws. In reality, this expression of these laws is derived from a more general integral form that it is often necessary to consider, primarily because it allows for a weaker notion of solution. The primary example of this being the case of a non-linear flux-function where shocks develop. Another important example is in the case of a discontinuous material property interface. In either case, i.e., for both shocks and discontinuous material properties, correct interface conditions must be considered by integrating over the discontinuity.

Equations of this type have been extensively studied at many different levels and are intimately related to the laws that govern most physical systems. In this thesis, we are primarily interested in several physical examples and we will now give several brief examples of how these type of laws appear naturally.

### 1.1.1 Electromagnetics

As a first example we will consider the macroscopic Maxwell's Equations governing the electromagnetic field in a dielectric,

$$\begin{aligned}\nabla \cdot \mathbf{D} &= \rho_f, \\ \nabla \cdot \mathbf{B} &= 0, \\ \mathbf{B}_{,t} + \nabla \times \mathbf{E} &= 0, \\ \mathbf{D}_{,t} - \nabla \times \mathbf{H} &= -\mathbf{J}_f.\end{aligned}$$

where  $\mathbf{E}(\mathbf{r}, t)$  is the electric-field,  $\mathbf{D}(\mathbf{r}, t)$  is the electric displacement field,  $\mathbf{B}(\mathbf{r}, t)$  is the magnetic field,  $\mathbf{H}(\mathbf{r}, t)$  is the magnetization field,  $\rho_f(\mathbf{r}, t)$  is the free charge density, and  $\mathbf{J}_f(\mathbf{r}, t)$  is the free current density.

To complete this system of equations we must impose constitutive relations relating the various physical fields, we will use lowercase letters to denote the specific vector functions describing the uppercase field,

$$\begin{aligned}\mathbf{D} &= \mathbf{d}(\mathbf{E}, \bar{\epsilon}), & \mathbf{E} &= \mathbf{e}(\mathbf{D}, \bar{\epsilon}), \\ \mathbf{H} &= \mathbf{h}(\mathbf{B}, \bar{\mu}), & \mathbf{B} &= \mathbf{b}(\mathbf{H}, \bar{\mu}).\end{aligned}$$

The primary example used in this thesis is that of the linear dielectric. This is given as follows:

$$\begin{aligned}\mathbf{D} &= \bar{\epsilon} \mathbf{E}, & \mathbf{E} &= \bar{\epsilon}^{-1} \mathbf{D}, \\ \mathbf{H} &= \bar{\mu}^{-1} \mathbf{B}, & \mathbf{B} &= \bar{\mu} \mathbf{H},\end{aligned}$$

where  $\overline{(\cdot)}$  denotes a second order material tensor, specifically,  $\bar{\epsilon}$  is the dielectric permittivity tensor and  $\bar{\mu}$  is the magnetic permeability tensor.

After applying the general form for the constitutive relationships we obtain the following equations governing the fields,

$$\begin{aligned}\nabla \cdot \mathbf{D} &= \rho_f, \\ \nabla \cdot \mathbf{B} &= 0, \\ \mathbf{B}_{,t} + \nabla \times (\mathbf{e}(\mathbf{D}, \bar{\epsilon})) &= 0, \\ \mathbf{D}_{,t} - \nabla \times (\mathbf{h}(\mathbf{B}, \bar{\mu})) &= -\mathbf{J}_f.\end{aligned}$$

The first two of these equations are not evolution equations and they must be satisfied throughout the material domain at each instant of time. In this exposition, we will typically be concerned with the effect of material parameter change on a propagating disturbance. It is for this reason that we will typically be assuming that the source terms are zero, i.e.,  $\rho_f = 0$  and  $\mathbf{J}_f = 0$ , though this need not always be the case. It can be shown that if the initial conditions satisfy these divergence equations then they remain satisfied. The second two equations are a system of conservation laws governing  $\mathbf{B}$  and  $\mathbf{D}$ , specifically,

$$\mathbf{q}_{,t} + \mathbf{f}(\mathbf{q}, \epsilon, \mu)_{,x} + \mathbf{g}(\mathbf{q}, \epsilon, \mu)_{,y} + \mathbf{h}(\mathbf{q}, \epsilon, \mu)_{,z} = 0,$$

where,

$$\mathbf{q} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix}, \mathbf{f} = \begin{bmatrix} 0 \\ -e_3 \\ e_2 \\ 0 \\ h_3 \\ -h_2 \end{bmatrix}, \mathbf{g} = \begin{bmatrix} e_3 \\ 0 \\ -e_1 \\ -h_3 \\ 0 \\ h_1 \end{bmatrix}, \mathbf{h} = \begin{bmatrix} -e_2 \\ e_1 \\ 0 \\ h_2 \\ -h_1 \\ 0 \end{bmatrix},$$

and  $\bar{\epsilon}$  and  $\bar{\mu}$  are prescribed tensor-valued functions of space  $\mathbf{r}$  and time  $t$ .

For example, in the case of a linear, isotropic material, these second order tensors reduce to scalars and we obtain the following system of six conservation laws relating  $\mathbf{B}$  and  $\mathbf{D}$ ,

$$\begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ D_1 \\ D_2 \\ D_3 \end{bmatrix}_{,t} + \begin{bmatrix} 0 \\ -D_3/\epsilon \\ D_2/\epsilon \\ 0 \\ B_3/\mu \\ -B_2/\mu \end{bmatrix}_{,x} + \begin{bmatrix} D_3/\epsilon \\ 0 \\ -D_1/\epsilon \\ -B_3/\mu \\ 0 \\ B_1/\mu \end{bmatrix}_{,y} + \begin{bmatrix} -D_2/\epsilon \\ D_1/\epsilon \\ 0 \\ B_2/\mu \\ -B_1/\mu \\ 0 \end{bmatrix}_{,z} = 0.$$

In this case, the material parameters  $\epsilon$  and  $\mu$  are being prescribed scalar-valued functions of space  $\mathbf{r}$  and time  $t$ .

For further simplification, assume that the fields only depend on  $t$  and  $z$ . The above system then reduces to a system of four conservation laws,

$$\begin{bmatrix} B_1 \\ B_2 \\ D_1 \\ D_2 \end{bmatrix}_{,t} + \begin{bmatrix} -D_2/\epsilon \\ D_1/\epsilon \\ B_2/\mu \\ -B_1/\mu \end{bmatrix}_{,z} = 0.$$

This is really two uncoupled systems, i.e., a system governing  $B_1$  and  $D_2$ , and a separate system governing  $B_2$  and  $D_1$ , we now consider the latter,

$$\begin{bmatrix} B_2 \\ D_1 \end{bmatrix}_{,t} + \begin{bmatrix} D_1/\epsilon \\ B_2/\mu \end{bmatrix}_{,z} = 0,$$

This is a hyperbolic system of two conservation laws for  $B_2$  and  $D_1$ . This system is equivalent to the acoustics system in the following sense. Introduce potential functions  $\phi$  and  $\xi$  such that  $\phi_t \equiv -D_1/\epsilon$ ,  $\phi_z \equiv B_2$ ,  $\xi_z \equiv D_1$ , and  $\xi_t \equiv -B_2/\mu$ , then the above system is equivalent to the following system,

$$\begin{bmatrix} \phi \\ \xi \end{bmatrix}_t - \begin{bmatrix} 0 & 1/\epsilon \\ 1/\mu & 0 \end{bmatrix} \begin{bmatrix} \phi \\ \xi \end{bmatrix}_z = 0.$$

It is interesting that this is the same as the system of acoustics equations for  $u$  and a potential  $v$ ,

$$\begin{bmatrix} u \\ v \end{bmatrix}_t - \begin{bmatrix} 0 & 1/\rho \\ k & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}_z = 0.$$

The expression for energy remains consistent in either case. The energy for the acoustics equation is given by

$$E = \frac{1}{2} \int_a^b (\rho u_t^2 + k u_z^2) dz. \quad (1.2)$$

We see that if we match up the analogous terms in the electromagnetic case, we arrive at the correct expression for the electromagnetic energy,

$$\begin{aligned} E &= \frac{1}{2} \int_a^b (\epsilon \phi_t^2 + (1/\mu) \phi_z^2) dz, \\ E &= \frac{1}{2} \int_a^b (D_1^2/\epsilon + B_2^2/\mu) dz, \\ E &= \frac{1}{2} \int_a^b \left( \epsilon E_1^2 + \frac{1}{\mu} B_2^2 \right) dz. \end{aligned}$$

### 1.1.2 Elasticity

As a second example, we will consider the equations of solid mechanics governing the stress in an elastic body,

$$(\rho \mathbf{u}_t)_t - \nabla \cdot \bar{\bar{\sigma}} = 0,$$

where  $\rho(\mathbf{r}, t)$  is the mass density, a scalar function of space and time, and  $\bar{\bar{\sigma}}(\nabla \mathbf{u})$  is the second order stress tensor which depends on the deformation gradient in a specified manner. This is determined by the constitutive relationship of the material under consideration. Specifically of interest to us is the example of linear elasticity where we have the following constitutive relationship,

$$\bar{\bar{\sigma}} = \bar{\bar{\bar{D}}} : \bar{\bar{\epsilon}},$$

where  $\epsilon = \frac{1}{2} (\nabla \mathbf{u} + \nabla \mathbf{u}^T)$  is the linear strain,  $\bar{\bar{\bar{D}}}$  is the fourth-order elasticity tensor, and the symbol  $:$  denotes the double convolution of the tensors to the left and right of the symbol.

For example, the constitutive relationship for an isotropic linear material is

$$\bar{\bar{\sigma}} = \lambda \text{tr}(\bar{\bar{\epsilon}}) \mathbf{I} + 2\mu \bar{\bar{\epsilon}}$$

where  $\lambda$  and  $\mu$  are the material parameters and are collectively known as the Lamé moduli of the material.

Introduction of the symbol  $\mathbf{M}$  for momentum vector gives the following relationship

$$\begin{aligned} \mathbf{M}_t - \nabla \cdot \bar{\bar{\sigma}}(\bar{\bar{\epsilon}}) &= 0, \\ \bar{\bar{\epsilon}}_t - \nabla \left( \frac{\mathbf{M}}{2\rho} \right) - \nabla \left( \frac{\mathbf{M}}{2\rho} \right)^T &= 0. \end{aligned}$$

Which is equivalent to the following system,

$$\mathbf{q}_t + \mathbf{f} \left( \mathbf{q}, \rho, \bar{\bar{\bar{D}}} \right)_x + \mathbf{g} \left( \mathbf{q}, \rho, \bar{\bar{\bar{D}}} \right)_y + \mathbf{h} \left( \mathbf{q}, \rho, \bar{\bar{\bar{D}}} \right)_z = 0$$

where

$$\mathbf{q} = \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \epsilon_{12} \\ \epsilon_{13} \\ \epsilon_{23} \end{bmatrix}, \mathbf{f} = \begin{bmatrix} \sigma_{11} \\ \sigma_{12} \\ \sigma_{13} \\ M_1/\rho \\ 0 \\ 0 \\ M_2/2\rho \\ M_3/2\rho \\ 0 \end{bmatrix}, \mathbf{g} = \begin{bmatrix} \sigma_{21} \\ \sigma_{22} \\ \sigma_{23} \\ 0 \\ M_2/\rho \\ 0 \\ M_1/2\rho \\ 0 \\ M_3/2\rho \end{bmatrix}, \mathbf{h} = \begin{bmatrix} \sigma_{31} \\ \sigma_{32} \\ \sigma_{33} \\ 0 \\ 0 \\ M_3/\rho \\ 0 \\ M_1/2\rho \\ M_2/2\rho \end{bmatrix}.$$

In a similar manner to the electromagnetic case, we can look to the example of a linear, isotropic material to simplify the above system of nine conservation laws relating  $\mathbf{M}$  and  $\bar{\epsilon}$ . Furthermore, we can place specific constraints on the above system which reduce it to the system of acoustics equations mentioned earlier.

## 1.2 Finite Volume Methods

Obtaining results regarding the energy accumulation effect (specifically the results discussed in chapter 4) is a difficult task to pursue by purely analytical means. To this end, it is imperative that a proper numerical procedure must be implemented to solve conservation laws of form mentioned in Section 1.1. Of the many possible choices, the high-resolution finite volume methods (FVM) summarized in [16, 17] best fit the purposes of this study. The main reason for this is that these methods allow for high-resolution solution of the PDEs governing the conservation of underlying physical quantities. Specifically, these methods allow for smart cell-wise control of material properties, which will allow us to suitably implement the various material geometries discussed in Section 1.3.

In the 1D case, we will be numerically solving the following type of law for the vector function  $\mathbf{q}(z, t)$ ,

$$\mathbf{q}_t + A(z, t)\mathbf{q}_z = 0, \tag{1.3}$$

where  $A(z, t)$  is an  $n \times n$  matrix with real eigenvalues  $\lambda_1 < \lambda_2 < \dots < \lambda_n$ . These eigenvalues represent the speed of the  $i$ -th family of waves and are a fundamental component to solving the system of equations. We assume that we have eigenvectors  $\mathbf{r}^1, \dots, \mathbf{r}^n$  corresponding to each eigenvalue.

To numerically solve equation (1.3) we must accurately approximate how the solution evolves in time. This can be done in a straightforward manner by using more advanced implementations of Godunov's (upwind) method. This method uses (or, in the nonlinear case, approximates) the solution to a Riemann problem between computational cells using the cell-wise averages as initial data. Recall that a Riemann problem is the problem of solving equation (1.3) with piecewise constant initial data. The main idea is that the solution (or approximate solution) of a cell-wise Riemann problem will allow for computation of flux into and out of each cell.

For an in-depth look at this see [16, 17]. We will now consider a specific example from [16] to illustrate the basic numerical procedure. Assume that we wish to solve three PDES ( $n = 3$ ) and assume that  $\lambda_1 < 0 < \lambda_2 < \lambda_3$ . The solution to a Riemann problem at every interface will result in the propagation of 3 discontinuities, each discontinuity propagating at the wave speed corresponding to the eigenvalue of the  $i$ th family. Specifically, the flux across each interface is found by decomposing

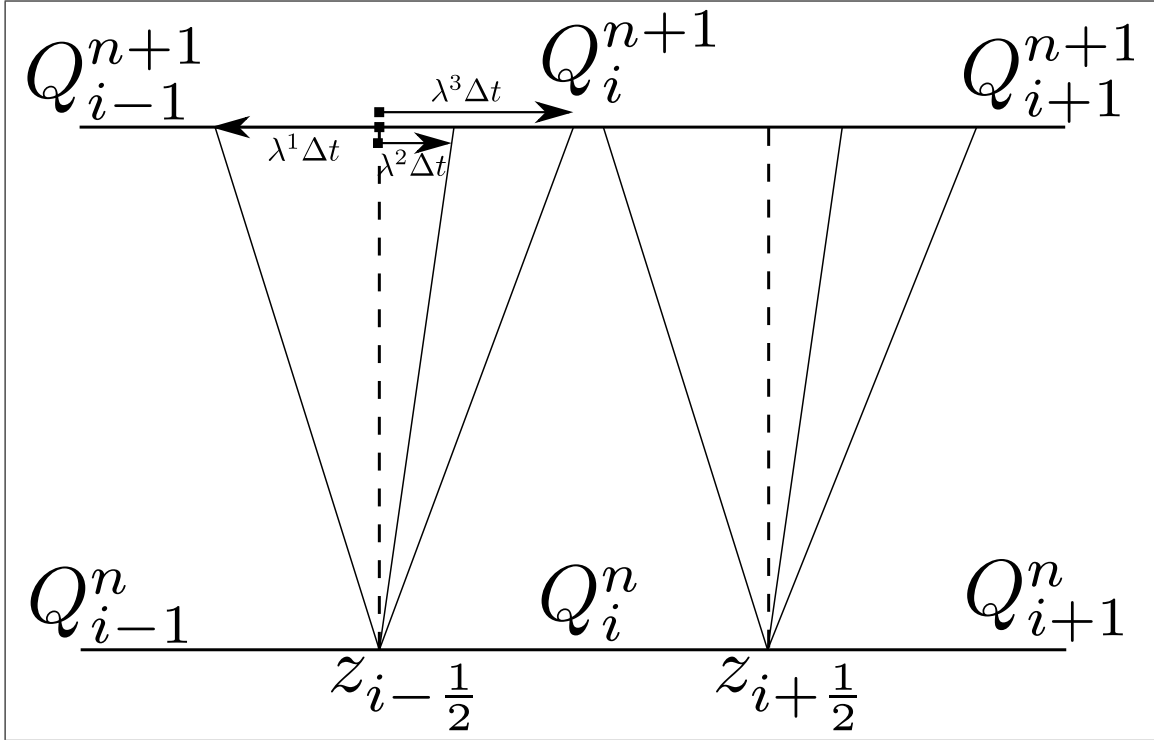


Figure 1.1: Typical FVM cell for  $n = 3$ .

the jump in solution at a given interface into respective  $i$ -th wave component, i.e.,  $Q_i^n - Q_{i-1}^n = \alpha_{i-\frac{1}{2}}^1 \mathbf{r}_{i-1}^1 + \alpha_{i-\frac{1}{2}}^2 \mathbf{r}_i^2 + \alpha_{i-\frac{1}{2}}^3 \mathbf{r}_i^3$ . The update to cell  $i$  will therefore be given by  $Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x} \left[ \lambda^2 (\alpha_{i-\frac{1}{2}}^2 \mathbf{r}_i^2) + \lambda^3 (\alpha_{i-\frac{1}{2}}^3 \mathbf{r}_i^3) + \lambda^1 (\alpha_{i+\frac{1}{2}}^1 \mathbf{r}_i^1) \right]$ . It is an interesting exercise to show that this example reduces to exactly what we would expect for Godunov's method in the case of the acoustics system, where  $\lambda_1 = -\sqrt{\frac{k}{\rho}}$ ,  $\lambda_2 = 0$ ,  $\lambda_3 = \sqrt{\frac{k}{\rho}}$ .

Notice that the above method conveniently allows us to use the computational grid to encode all information regarding the changing material geometry. For the checkerboard structure, the above method is relatively straightforward to implement, because the material geometry aligns nicely with the standard grid. For different spatial-temporal geometries (laminar, or more complicated, etc, ) this can be accomplished by introducing a moving grid that follows the spatial-temporal material geometry [37, 38, 11].

Godunov's method provides a good start, however, it is only first order accurate. The next logical step is to consider construction of smart and efficient high-resolution versions of the algorithm. This can be done with the procedure of limiting and is explained in depth in [16, 17]. Practically, it is implemented by adding an appropri-

ately constructed 2nd order term to update the cell-wise averages from  $n$  to  $n + 1$ . This term makes the method more accurate by ensuring known properties of conservation laws, e.g., variation diminishing (TVD), etc.

Another enhancement that will be used is called adaptive mesh refinement (AMR). In AMR, one keeps track of specific properties of the solution and (typically, some measure of the gradient) and uses this information to locally refine the computational grid. Specifically, around the cells where this value becomes more than a given threshold, the grid is refined in an attempt to better resolve the solution in this region. This allows for accurate solution of the governing conservation laws in regions where the gradient is very large. Much research has been done in this area [4]. Applying this method to the FVM procedure described above involves providing a course mesh (Level 1) along with multiple other refinement levels.

The simulations produced in this work were produced by using suitably modified standard Clawpack solvers [9]. Clawpack is a highly developed open source suite of computational tools that implement the above methods for FVM. Originally developed in Fortran by Randall Leveque, it has since grown into an amazing tool for researchers investigating wave-propagation problems.

### 1.3 Previous work on Dynamic Materials

The study of *dynamic metamaterials* (DM) is the study of materials that have properties which are controllable in space and time, i.e., formations assembled from materials that are distributed on a particular scale in space-time. This material concept takes into consideration inertial, elastic, electromagnetic and other material properties that affect the dynamic behavior of various mechanical, electrical and environmental systems. In static or non-smart applications, design variables, such as dielectric permittivity and magnetic permeability, material density and stiffness, yield force and other structural parameters are position dependent but invariant in time. When it comes to dynamic applications, we also need temporal variability in the material properties in order to adequately match the changing environment. To this end, in dynamic material design, dynamic materials will take up the role played by ordinary composites in static material design. Due to this variability, DM may adequately react to environmental challenges and changing circumstances, which makes them ideal for optimal control problems in space-time. These materials, also known as dynamic materials, have been introduced by K. Lurie [24] and studied in many different forms by various authors [26, 27, 25, 37, 38, 39, 31, 30, 12, 32, 28]. DM have been shown to demonstrate many effects unthinkable with ordinary materials; effects including screening of extended spatial domains from the intrusion of disturbances, the idea of left-handed materials with negative “effective” material parameters, accumulating and storing energy in pulses of high power, compressing

signals, creation of left-handed materials, appearance of Psuedo-Coriolis effect, etc.

These results have been obtained through both analytical and numerical methods and this introduction will summarize some results that have been discovered. In the following we assume that we have a material that supports linear wave propagation in either 1D or 2D.

1D wave propagation through a DM is governed by the variable coefficient wave equation,

$$(\rho u_t)_t - (k u_z)_z = 0, \quad (z, t) \in [a, b] \times [0, T], \quad (1.4)$$

$$u(z, 0) = f(z), \quad u_t(z, 0) = g(z), \quad (1.5)$$

$$u(a, t) = u_L(t), \quad u(b, t) = u_R(t), \quad (1.6)$$

where,  $\rho(z, t)$  and  $k(z, t)$  act as controls, i.e., they are functions of space and time that are prescribed by some material designer. These equations describe wave propagation in an elastic context where  $\rho$  is the mass density and  $k$  is the stiffness. In an electromagnetic context we replace  $\rho$  with  $\epsilon$  - the dielectric permittivity, and  $k$  with  $\frac{1}{\mu}$  - the inverse of the magnetic permeability. In either case, the point-wise velocity of the wave will be  $a(z, t) = \sqrt{k(z, t)/\rho(z, t)}$ , and as shown before, the energy of a traveling wave at some time  $t$  is given by the integral:

$$E(t) = \int_a^b (\rho u_t^2 + k u_z^2) dz. \quad (1.7)$$

This thesis primarily deals with the 1D case. However, this section presents a general framework for viewing these problems, because of the many possible extensions into various realms, e.g., multiple dimensions and non-linear conservation laws.

### 1.3.1 Characterization of Dynamic Materials

Dynamic materials may appear in very diverse physical implementations, including mechanical and electromagnetic. Nevertheless, we may distinguish two principal ways of making them, specifically, by the spatial-temporal mixing of ordinary materials via the processes of either activation or kinetization [5, 22]. Dynamic materials of the first type are obtained by instantaneous or gradual change of the material parameters (stiffness, self-induction, capacitance, etc.) in various parts of the system in the absence of relative motion of those parts. This procedure has been called activation [5, 22]. and the corresponding materials termed dynamic materials of the first kind, or activated dynamic materials.

Many examples of activated dynamic materials originate in electrical engineering. As an illustration, consider a transmission line assembled as an array of LC-circuits



connected in series with the inductance  $L$  and capacitance  $C$  changeable in each circuit by switching. A pump “wave of linear capacitance” may be generated through the use of  $p$ - $n$  junction diodes distributed along the line and appropriately activated in space-time [18]. A similar “wave of inductance” may be created through the use of a series arrangement of non-linear inductors. Magnetic nanoparticles exhibit the unique phenomena of superparamagnetism and quantum tunneling of magnetization, accompanied by unusually high coercivities. These effects are observed in magnetic materials such as  $\gamma$ - $\text{Fe}_2\text{O}_3$  nanocrystals and ferrogels [40] at room temperature. One can effectively control the inductance of those materials by varying the magnetic field.

Dynamic materials of the second type are obtained when various parts of the system are exposed to relative motion that is prearranged and generated in a certain way. This procedure has been called kinetization, and the relevant materials termed the dynamic materials of the second kind, or kinetic dynamic materials. The materials of this type can be perceived as mixtures of two or more ordinary materials that alternate in space on a microscale; this alternation occurs due to the fact that every constituent participates in its individual material motion taking the material pattern along with it. Vibrational motion is most important in this respect; in particular, high frequency standing waves represent a mechanism of creating kinetic dynamic materials, refer to reference [7] for more on this. Dynamic materials rarely appear to be of natural origin (living tissue being a notable exception). They are, almost exclusively, products of modern technology. Specifically, we refer to ferroelastic and ferromagnetic materials when it comes to electronic frequencies, and to laser techniques when it comes to optical frequencies. Both ways work to create permittivity and permeability  $(\epsilon, \mu)$  patterns that are tunable in space and time and therefore completely fall into the category of activated dynamic materials.

In the next subsections we will look at two different geometries that produce interesting effects on waves that propagate through the respective DM. In Section 1.3.2 we review effects produced by a laminar geometry, and in Section 1.3.4 we look at interesting effects produced by a checkerboard geometry. Extensions of this checkerboard geometry is the main focus of this thesis. In Section 1.3.3 we see that dynamic materials make it possible to construct left-handed materials with negative effective values or material parameters.

### 1.3.2 Laminar Material Geometry

It has been shown analytically and numerically in [21, 23] and [37] that by appropriately controlling the design factors of a dynamic laminate it is possible to selectively screen large domains in space-time from the invasion of long wave disturbances. Achieving this “screening effect” is impossible in an ordinary static composite.

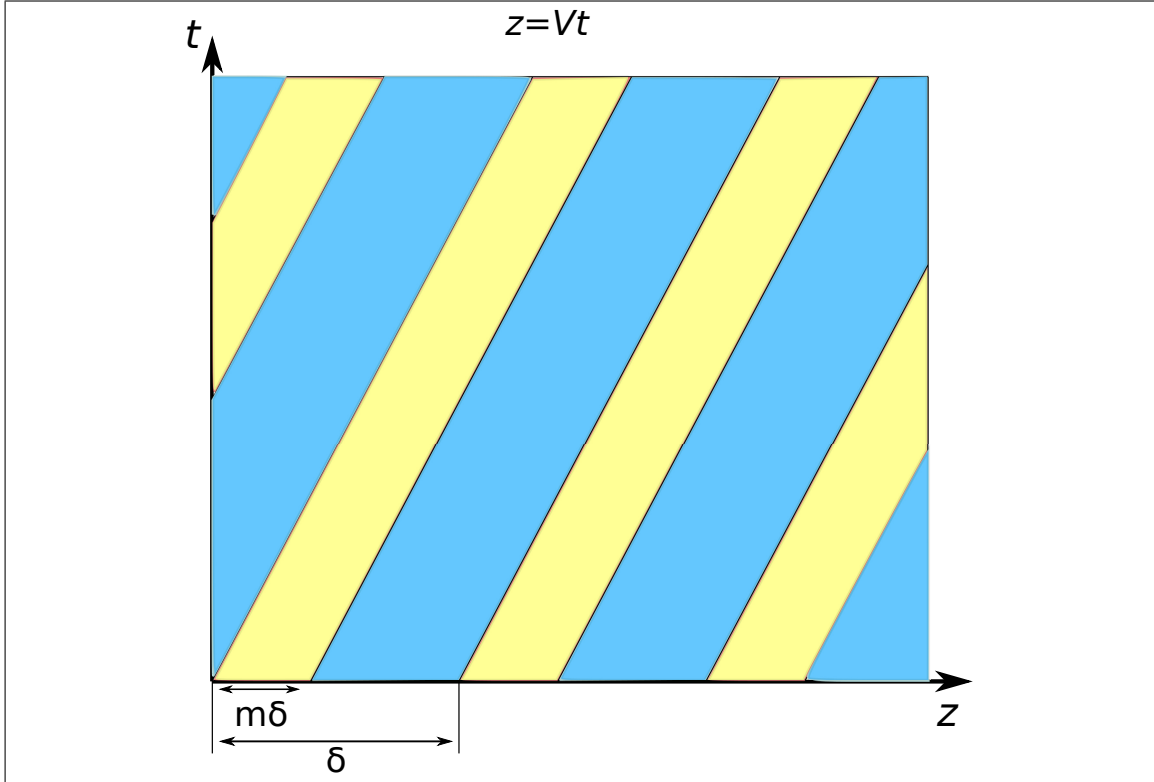


Figure 1.2: Laminar material geometry for  $m = \frac{2}{3}$

Assume that the material parameters  $\rho$  and  $k$  can take two values,  $(\rho_1, k_1)$  or  $(\rho_2, k_2)$ . Furthermore, assume that initially these materials alternate periodically along the  $z$ -axis with period  $\delta$ , and material 1 has volume fraction  $m \in [0, 1]$ . Lastly, assume that the property pattern travels at fixed velocity  $V$ . This generates a laminar property pattern in space-time (see Figure 1.2), with  $\rho$  and  $k$  given by the following functions:

$$\rho(z, t) = \begin{cases} \rho_1, & (z - Vt) \bmod m\delta \in [0, m\delta), \\ \rho_2, & (z - Vt) \bmod m\delta \in [m\delta, \delta), \end{cases}$$

and,

$$k(z, t) = \begin{cases} k_1, & (z - Vt) \bmod m\delta \in [0, m\delta), \\ k_2, & (z - Vt) \bmod m\delta \in [m\delta, \delta), \end{cases}$$

where  $m \in [0, 1]$ .

It is important to guarantee that compatibility conditions are observed on the material interfaces. Due to the hyperbolic nature of equation (1.4), this condition amounts to allowing for regular transmission of characteristics across the interface.

As explained in reference [24], this condition means that we cannot allow  $|V| \in [a_1, a_2]$ ; mathematically, this condition is equivalent to requiring that

$$(V^2 - a_1^2)(V^2 - a_2^2) > 0.$$

Following standard homogenization procedures [24], it becomes possible to derive an “effective” wave equation (see eq (2.41) in [24]) that governs the effective motion of disturbances through this medium. The solution of the corresponding homogenization problem gives the following “effective” PDE governing disturbances to be

$$r(u_0)_{tt} + 2q(u_0)_{zt} + p(u_0)_{zz} = 0, \quad (1.8)$$

where  $r$ ,  $q$ , and  $p$  are values depending on  $m$ ,  $a_1$ ,  $a_2$ , and  $V$ . This is a hyperbolic PDE that can be solved as follows.

Furthermore, looking for the solution to this equation in the form  $f(z - vt)$  gives that the phase velocities  $v_{\frac{1}{2}} = q \pm \sqrt{q^2 - rp}$  of solutions to equation (1.8) are roots of the equation

$$rv^2 - 2qv - p = 0.$$

Thus, the solution to equation (1.8) will consist of two waves, each traveling at velocity  $v_1$  and  $v_2$ . Proper choice of parameters  $m$ ,  $a_1$ ,  $a_2$ , and  $V$  allows  $v_1$  and  $v_2$  to have the same sign, which guarantee coordinated wave motion in that direction. This effect has been extensively studied in the literature, and termed the “screening” effect for its ability to protect large regions of space-time from the intrusion of disturbances. This effect is demonstrated in Figure 1.3, where a screened region is created between two laminations oriented in opposite directions.

These ideas are further extended in [39] where the techniques of Floquet analysis and asymptotic expansions are used to reveal the dispersive nature of the effective lamination, resulting in the following effective equation

$$u_{tt} + (\Sigma_+ + \Sigma_-)u_{tz} + (\Sigma_+\Sigma_-)u_{zz} + (\Gamma_+ - \Gamma_-)\delta^2u_{zzzt} + (\Gamma_+\Sigma_- - \Gamma_-\Sigma_+)u_{zzzz} = 0$$

where  $\Gamma_{\pm}, \Sigma_{\pm}$  are values depending on  $m$ ,  $a_1$ ,  $a_2$ , and  $V$ . The effects are supported by direct numerical simulation of the heterogeneous problem. These results are compared with the exact solution of the effective equation with the second order equation (1.8), and the higher order equation for static materials ( $V = 0$ ) in [33] and are shown to provide insight into the dispersive nature of wave propagation through laminates.

Some 1D results were extended to 2D in [31]. In this paper, the equations of motion for an elastic laminar spatial-temporal composite are investigated. The composite under consideration is assumed to be binary, that is, it is assembled of two original constituents capable of changing (in space-time) their material density, as well as their material stiffness. The condition of plane strain was then imposed on the composite. The paper begins by attempting to evaluate the materials' average Lagrangian (action density). In doing so, it immediately becomes apparent that expressions are needed for average momentum and stress. Both quantities are found to depend linearly on average strain and average velocity.

Assume the lamination is composed of two different elastic materials (1 and 2) with average stiffness tensors in each material given by  $D_1$  and  $D_2$  respectively.

$$D(z, t) = \begin{cases} D_1, & (z - Vt) \bmod m\delta \in [0, m\delta), \\ D_2, & (z - Vt) \bmod m\delta \in [m\delta, \delta), \end{cases}$$

where  $m \in [0, 1]$ .

Specifically, the effective motion of an elastic bar under plane strain was found to be given by the following homogenized equations of motion:

$$(M_{eff} \cdot \mathbf{u}_t)_t - \nabla \cdot (D_{eff} : e) - \frac{\partial}{\partial t} (e : \Lambda) - \nabla \cdot (\Lambda \cdot \mathbf{u}_t) = 0,$$

where  $D_{eff}$  and  $M_{eff}$ , are respectively, “effective” material tensors of elastic and inertial properties arising from the homogenization process. This is a direct generalization of equation (1.8) and the effective tensors are shown to reduce to their previously found formulas under the assumption of isotropy of the individual material constituents. This reduction is shown in [31]. In brief, after calculating the general homogenized Euler equations of motion, isotropy was assumed and it immediately becomes apparent that material tensor  $\Lambda$ , termed the “tensor of kinetic stresses” produces two additional forces  $\underline{F}_{\dot{e}}$  and  $\underline{F}_{\dot{\Omega}}$  due to the moving lamination. These two forces are due to the presence of simultaneous change in both inertial and elastic properties of the original material constituents.  $\underline{F}_{\dot{e}}$  is due to the symmetric portion of the deformation gradient while  $\underline{F}_{\dot{\Omega}}$  is due to the antisymmetric portion and is special because it is mathematically equivalent to a Coriolis-type force. Typically, the Coriolis Effect is due to a rotating reference frame, however, this new force term is an actual force that is directly due to the moving material property pattern. The appearance of these two forces is a consequence of both dynamics and plane strain; the Coriolis type force disappears in the case of one dimensional strain that arises when longitudinal dynamic disturbances propagate along an elastic bar while the force  $\underline{F}_{\dot{e}}$  reduces to the mixed derivative term in equation 1.8. Furthermore, the effective tensors  $M_{eff}$  and  $D_{eff}$  can also be shown to reduce to  $p$  and  $q$  from equation 1.8. This extension fully reduces to the case of a 1D elastic bar.

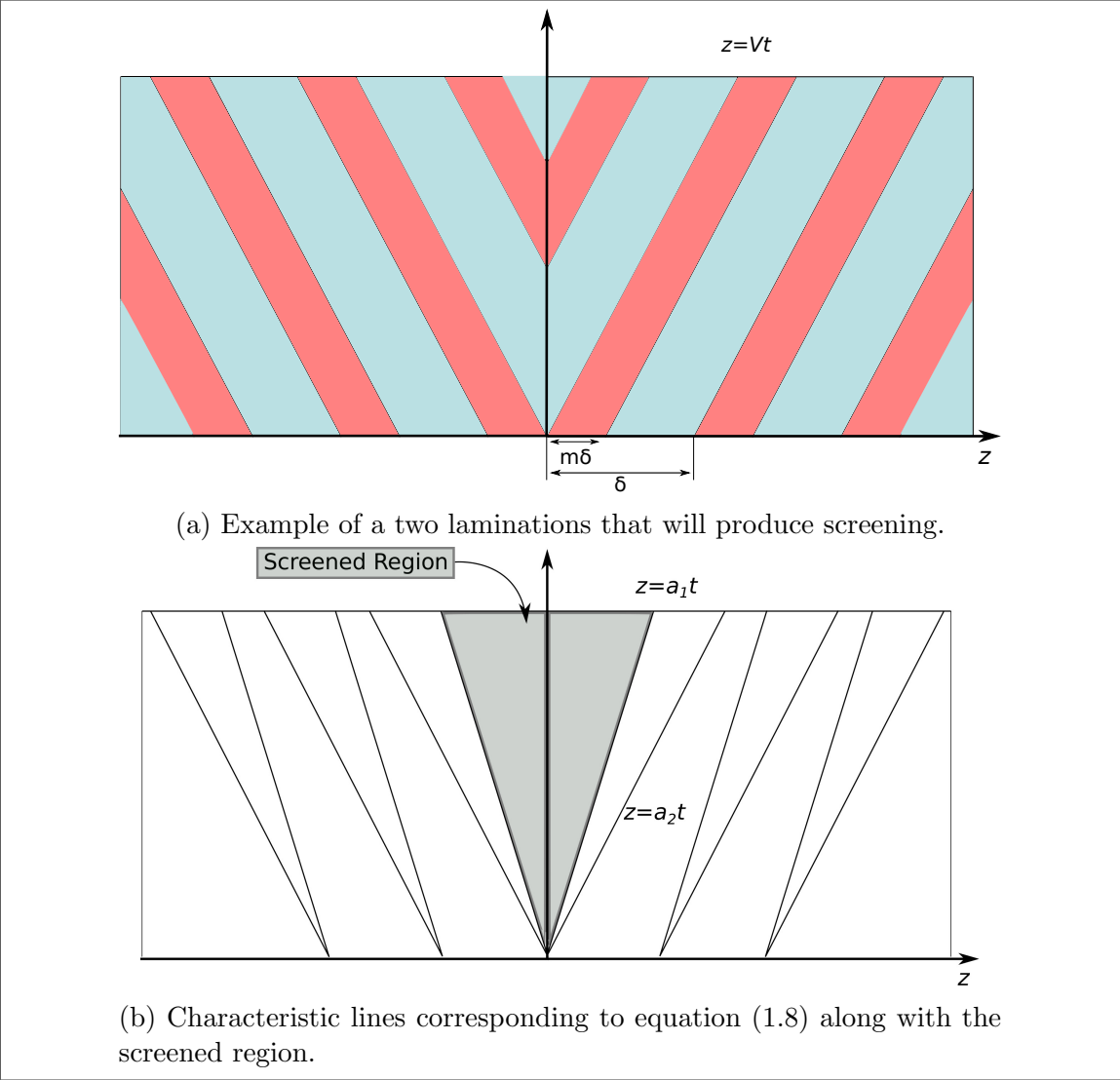


Figure 1.3: Screening effect in a laminar geometry

### 1.3.3 Activated Left-Hand Medium

It has been discovered [25] that, for certain ranges of parameters, an activated dynamic laminate made up of two materials with positive  $\epsilon$  and  $\mu$ , yields negative values of its effective permittivity and permeability. This conclusion is obtained after applying homogenization to the laminate in a laboratory frame, and then moving to a proper frame to see that the eigenvalues of the material tensor are both negative. The eigenvalues are invariant, so this composite is left-handed [36] with regard to long-wave disturbances. As far as we know, this is the first attempt to produce left-handed materials via spatio-temporal material assemblages. It is important that such a material may be tunable, i.e. its properties may be controlled. The left-handed materials otherwise offered all involve static elements as their basic components [29].

### 1.3.4 Checkerboard Material Geometry

Another material geometry that has shown to produce interesting effects is the “checkerboard” geometry. Similar to the laminar case, assume that the material parameters  $\rho$  and  $k$  can take two values,  $(\rho_1, k_1)$  or  $(\rho_2, k_2)$ . Furthermore, assume that initially these materials alternate periodically along the  $z$ -axis with period  $\delta$ , and material 1 has volume fraction  $m \in [0, 1]$ . Assume that we allow for an instantaneous switching of properties from material 1 to material 2, and vice-versa, and that this switching occurs periodically at period  $\tau$ , and that the first switch occurs at time  $n\tau$  where  $n \in [0, 1]$ . This generates a checkerboard property pattern in space-time (see Figure 1.4). Specifically,  $\rho$  and  $k$  are given by the following functions

$$\rho(z, t) = \begin{cases} \rho_1, & (z \bmod m\delta, t \bmod n\tau) \in [0, m\delta) \times [0, n\tau) \cup [m\delta, \delta) \times [n\tau, \tau), \\ \rho_2, & (z \bmod m\delta, t \bmod n\tau) \in [m\delta, \delta) \times [0, n\tau) \cup [0, m\delta) \times [n\tau, \tau), \end{cases} \quad (1.9)$$

and,

$$k(z, t) = \begin{cases} k_1, & (z \bmod m\delta, t \bmod n\tau) \in [0, m\delta) \times [0, n\tau) \cup [m\delta, \delta) \times [n\tau, \tau), \\ k_2, & (z \bmod m\delta, t \bmod n\tau) \in [m\delta, \delta) \times [0, n\tau) \cup [0, m\delta) \times [n\tau, \tau). \end{cases} \quad (1.10)$$

To simplify the problem, we assume that the wave impedances  $\gamma_i = \sqrt{\rho_i k_i}$  of both materials are equal, but their phase velocities  $a_i = \sqrt{\frac{k_i}{\rho_i}}$  are different with  $a_2 > a_1$ .

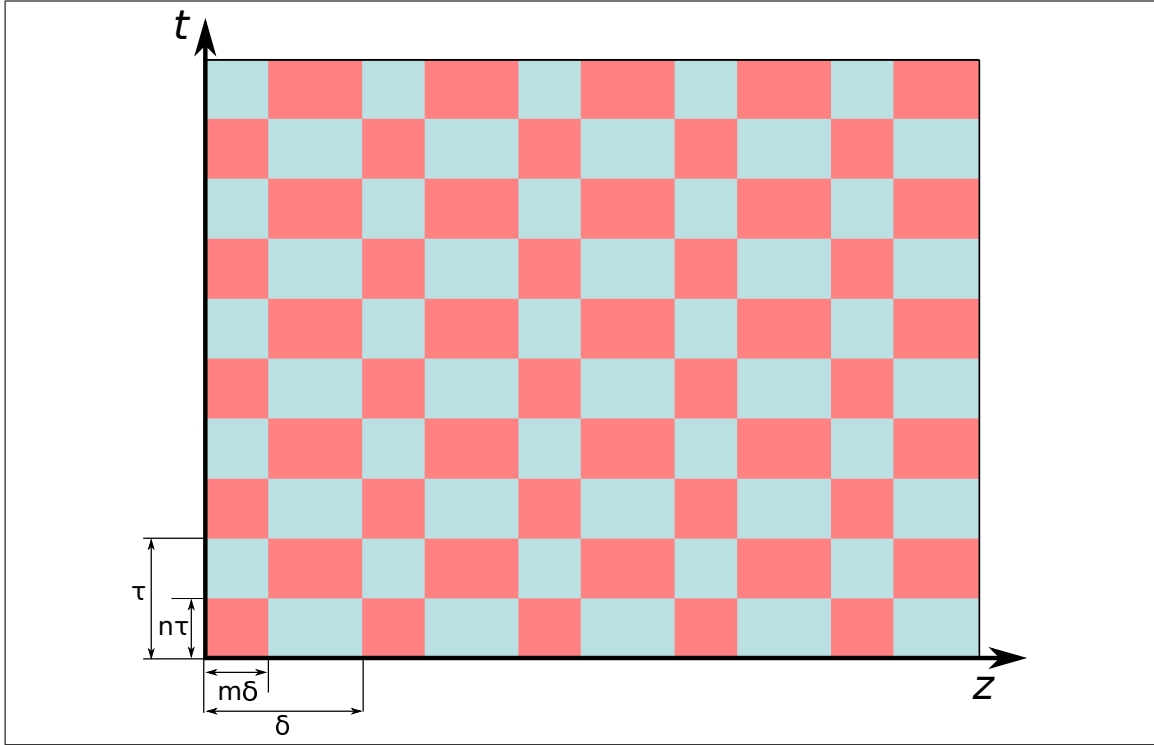


Figure 1.4: Checkerboard structure in spacetime. Material 1 is colored in red and material 2 is colored in blue.

Equation (1.4) can be reduced to a first order hyperbolic system by introduction of a potential function  $v$ , defined as follows:

$$\rho u_t = v_z, \quad k u_z = v_t. \quad (1.11)$$

Thus, a plane wave traveling in the  $z$ -direction through such a structure is governed by the preceding hyperbolic system. Under the assumption of matching wave impedance mentioned above, this problem reduces to two first order problems for the Riemann invariants  $R = u - \frac{v}{\gamma}$  and  $L = u + \frac{v}{\gamma}$  propagating independently of each other according to the equations

$$R_t + a R_z = 0, \quad L_t - a L_z = 0. \quad (1.12)$$

The compatibility conditions on material interfaces show that on every such boundary, vertical or horizontal, an incident primary wave initiates only one secondary wave that travels into adjacent material.

Consider the motion of one Riemann invariant, say  $R$ . Assume that initially,  $R$  takes the shape of a smooth impulse, e.g., a Gaussian, over one spatial period. This impulse is viewed as an array of ordinates, each of them propagating without change

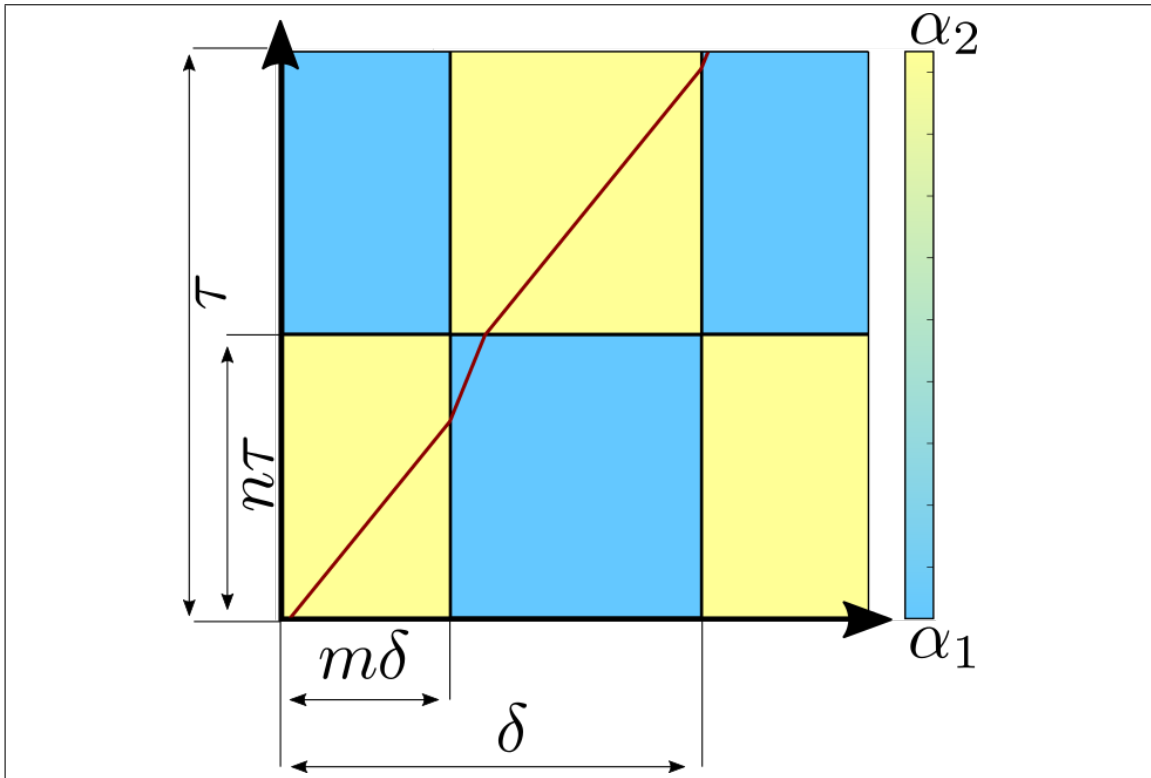


Figure 1.5: Checkerboard structure in spacetime. Material 2 is colored in yellow and material 1 is colored in blue.



along the relevant characteristic originating on the  $z$ -axis. Each characteristic path is represented in the  $(z, t)$ -plane as a series of line segments with slope equal to the reciprocal of the phase speeds in each rectangle. Due to the regularity conditions mentioned earlier, no two characteristics will intersect.

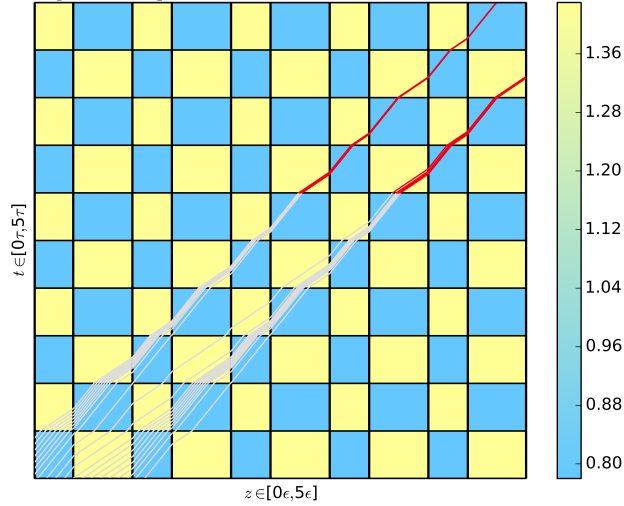
Proper choice of material parameters  $k$  and  $\rho$  along with geometrical parameters  $m$  and  $n$  give rise to a “focusing” effect on the characteristic lines propagating through the checkerboard geometry. An example of this can be seen in Figure 1.6. In this image, the dark region is the slow material and the light region is the fast material. Notice that the characteristic lines enters the fast material through the temporal gate and they enter the slow material through the spatial gate, this is a fundamental property of the limit cycles studied.

Particularly, for the structure shown in Figure 1.6a with parameters, the characteristics are gathered into identical groups, one group per period; all of the characteristics in each group converge to one of them that becomes a stable limit cycle (a stable cycle on a torus). This is clearly demonstrated in Figure 1.6a. The computed rotation number or speed of the limit cycle is  $\frac{\dot{\phi}}{\tau} = 1$  in this example. The neighboring groups of characteristics that converge to stable limit cycles are separated by a special characteristic path, termed an unstable limit cycle. We see that the array of characteristic ordinates are almost all tightened to a single point, the location of the stable limit cycle. The evolution of a disturbance is shown in Figure 1.6b. This was obtained as a result of direct numerical simulation of the wave motion as governed by Equation (1.4) through the checkerboard structure. The disturbance is initially supported over one period around a stable limit cycle. It is clear that this disturbance sharpens into a spike, gradually getting sharper and sharper. Figure 1.7 shows that if the initial disturbance is taken over multiple periods, this convergence effect is again observed, and the location. In this case, the location of the stable limit cycles correspond to the “steps” in the wave profile, while the location of the unstable limit cycles correspond to the center of each stair.

The focusing effect is remarkable because it corresponds to a pumping of energy into the wave exponentially at the rate of  $(a_2/a_1)^2$ , as is shown in [24, 26, 27]. A close look at Figure 1.6a reveals that, in this example, within a close vicinity of a limit cycle, an array of characteristics always leaves material 2 across the vertical (spatial) interface whereas it enters this material across the horizontal (temporal) interface. The energy of the array experiences a finite increment each time the array enters material 2, specifically, the energy increases by the factor  $\frac{a_2}{a_1} > 1$ . Having passed through material 2, the array enters material 1 across the vertical boundary with its energy remaining unaffected. The next increment in energy occurs at the following invasion into material 2 across the horizontal boundary, etc. We obtain an exponential growth of energy and its concentration within narrow peaks, occurring at the cost of the work produced by an external agent (e.g. laser beam) against

(a) Convergence of characteristic lines in a checkerboard.

Characteristics for  $\delta=1.0$ ,  $\tau=1.0$ ,  $m=0.4$ ,  $n=0.5$ ,  $\alpha_1=0.78$ ,  $\alpha_2=1.43$ ,  
 $p=1e-05$ ,  $q=1e-05$ ,  $dt=0.00250125062531$



(b) Typical behavior of a wave propagating through several periods of a checkerboard. The initial support is over one period.

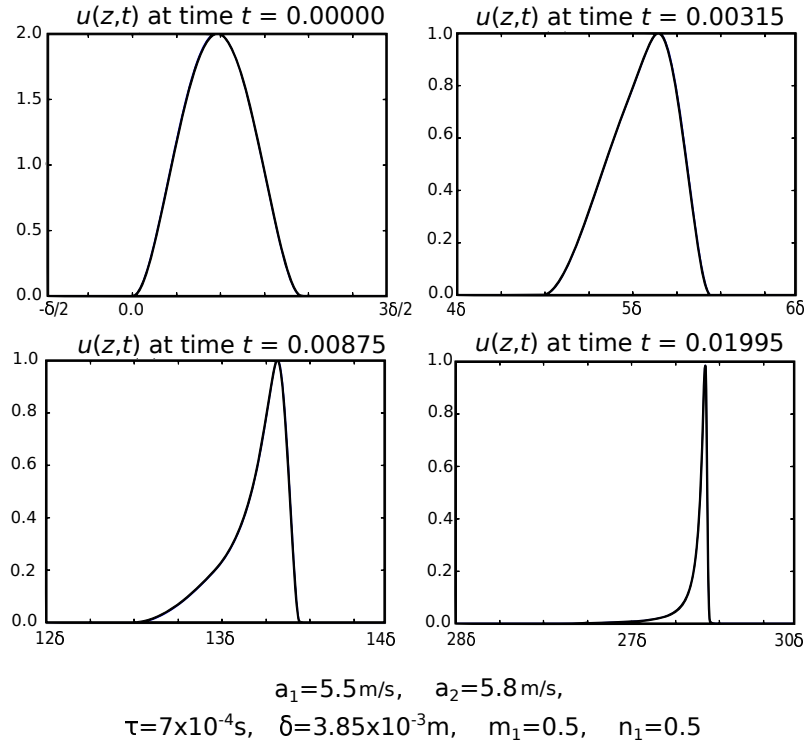


Figure 1.6: Focusing effect in a checkerboard geometry for disturbance initially supported over one period.

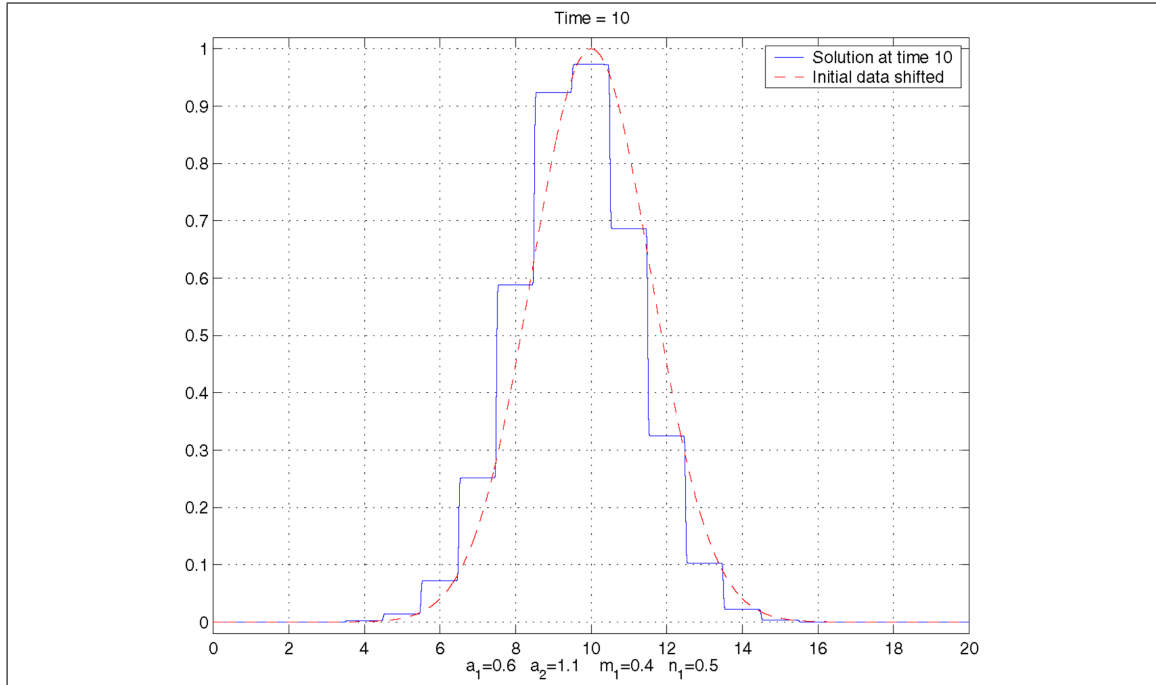


Figure 1.7: Focusing effect in a checkerboard geometry for disturbance initially supported over one period.

the electromagnetic forces at the horizontal interfaces. Transition from material 1 to material 2 occurs twice every period, each time giving the  $\frac{a_2}{a_1}$  factor increase in energy [26]. This represents an obvious analogy with a swing in a space-distributed version. This simple mechanism may well work towards optical pumping and high-energy pulse compression.

In [26], the convergence phenomena was shown to occur only when the parameters were found to be in the “plateau” region, i.e., a region of parameter space that generates a plateau in the average velocity profile. This effect has been theoretically categorized in [27], and agrees very well with computational results.

In the next chapter, we will summarize the results from the paper [32]. It is an extension of the checkerboard geometry to linear elastic wave propagation.

## Chapter 2

# Dilatation and Shear Wave Propagation in Dynamic Checkerboard

In paper [32], I studied elastic plane waves traveling normal to the spatial interfaces of a checkerboard structure in space time. For linear isotropic elastic media, there are two different families of waves, dilatation waves (d-waves) and transverse shear waves (s-waves), which can both propagate through the elastic material along some axis. Paper [32] examines the propagation of dilatational and shear waves through an isotropic elastic material having the dilatation and shear moduli variable in space and time. Specifically, two isotropic materials alternate occupying rectangular cells in 3D space + time producing a double periodic checkerboard material assembly. The materials are assumed to differ in their wave velocities (dilatational and shear) but to have pairwise equal values of wave impedances for each type of wave. The two wave types are governed by distinctly different wavespeeds and are independent of one another.

The main question investigated was whether or not characteristic focusing could be achieved simultaneously in both type of wave. We found that this simultaneous focusing can be achieved and that there is a “joint” plateau region where characteristics for both type of wave converge. Perhaps the most surprising, but perfectly reasonable, result is that despite having a different wavespeed in each material section, the average velocity of dilatation and shear waves must be the same if they are to be on the same plateau!

We show that, for both types of waves traveling normally to spatial interfaces between the materials, the average velocity of propagation is the same for certain ranges of material and structural parameters. Also, the energy is accumulated in those waves, and such accumulation occurs in very narrow pulses. This is unlike the wave propagation in a uniform static material, where both types of waves propagate

at different speeds. The coincidence of average speeds of propagation appears to be due to the checkerboard material geometry. It creates the “plateau effect” within the ranges of material and structural parameters mentioned above [24, 26, 27]. These ranges do not include the purely uniform material. In elastodynamics, the concept of dynamic materials has been previously studied in references [31, 15].

## 2.1 Introduction

In the case of wave propagation along a 1D elastic bar, it was possible to support energy accumulation through the use of a “checkerboard geometry” [24, 26, 27]. This geometry is detailed in Fig. 1.4. This configuration introduces spatial and temporal interfaces, along which either spatial or temporal coordinate remains constant. These interfaces separate material domains with different physical properties. To achieve energy accumulation in traveling waves, it was noticed that such properties must be arranged so as to match wave impedances in the different material cells. This results in the absence of wave reflection through both spatial and temporal property switches. A wave traveling through such a structure will, for certain ranges of the checkerboard material and structural parameters, accumulate energy supplied by an external agent at the moments of temporal switching.

At such moments, this structure would turn the work applied by the external agent against the wave, adding to the wave’s energy. The “properly tuned” checkerboard material geometry supports nonstop energy pumping due to its special ability to maintain favorable arrangement of the traveling wave routes, accompanied by simultaneous elimination of reflections. Another remarkable feature of a checkerboard has been called the “plateau effect”. As shown in reference [27], in the absence of reflections in a spatial-temporal checkerboard, there are continuous ranges of structural parameters  $m, n$ , and phase velocities  $c_1$  and  $c_2$ , for which the characteristics of a traveling wave converge to limit cycles (see Figure 1.6a). Figure 1.6b illustrates the evolution of an initially Gaussian-shaped disturbance during its travel through a spatial-temporal checkerboard composite. This evolution corresponds to the wave routes illustrated in Figure 1.6a. This system appears to be robust, and this robustness is the main reason why the phenomenon of wave propagation at the same average velocity may be extended to elastic waves of different types.

Specifically, we consider waves propagating in along a specified axis in a dynamic 3D isotropic elastic checkerboard. In this scenario, both waves (dilatation and shear) travel through each material with their own phase velocities and wave impedances. We ask if the wave energy can be simultaneously accumulated in both types of waves as they propagate normal to the spatial interfaces. This is not apparent, and represents the development of the situation observed before in a 1D elastic bar. The

reason for our claim is that both types of waves are governed by the homogeneous wave equation, and their phase velocities  $c_D$  and  $c_S$  may both belong with the same plateau, i.e., both waves have the same average phase velocity.

## 2.2 Background

For a linear elastic media, we will denote the displacement field as  $\mathbf{u}(z, t)$ . This vector field satisfies the elastic wave equation in each material domain:

$$(\rho \mathbf{u}_t)_t - \operatorname{div}[\sigma] = 0, \quad (2.1)$$

where  $\sigma(\mathbf{u}, \lambda, \mu) = \lambda I \operatorname{div} \mathbf{u} + \mu (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$  is the stress tensor,  $\lambda$  and  $\mu$  are the Lamé moduli of the first and second kind,  $\rho$  is the mass density,  $I$  is the identity matrix, and  $()^T$  denotes the matrix transpose operation. For uniform materials (i.e., pure materials), this equation reduces to

$$\rho \mathbf{u}_{tt} - (\mu \nabla^2 \mathbf{u} + (\lambda + \mu) \nabla (\nabla \cdot \mathbf{u})) = 0. \quad (2.2)$$

A plane wave propagating along the  $z$ -axis is given by the following expression  $\mathbf{u} = f(z - ct)\mathbf{d}$ , where  $\mathbf{d}$  is the fixed direction of the material displacement. Plane wave solutions to equation (2.2) are possible if either  $\mathbf{d} \cdot \mathbf{k} = 0$ , or  $\mathbf{d} \times \mathbf{k} = 0$ , where  $\mathbf{k}$  is the unit vector along the  $z$ -axis. Waves of the first type, where material displacement is transverse to the direction of wave motion, are termed transverse waves, shear waves, or  $s$ -waves. Waves of the second type, where material displacement is parallel to the direction of wave motion are termed longitudinal waves, dilatational waves, or  $d$ -waves. Within uniform materials, both types of waves are governed by the wave equation  $u_{tt} - c^2 u_{zz} = 0$ , however they travel at distinct wave speeds  $c_d = \sqrt{\frac{\lambda+2\mu}{\rho}}$  and  $c_s = \sqrt{\frac{\mu}{\rho}}$ .

## 2.3 Results

I considered the motion of plane waves through a dynamic checkerboard composite assembled from two isotropic linear elastic materials that fill  $\mathbb{R}^3$ . The Lamé moduli ( $\lambda$  and  $\mu$ ) and mass density  $\rho$  are given by doubly periodic piecewise constant functions of space and time:

$$(\lambda, \mu, \rho) = \begin{cases} (\lambda_1, \mu_1, \rho_1) & (z \bmod m\delta, t \bmod n\tau) \in [0, m\delta) \times [0, n\tau) \cup [m\delta, \delta) \times [n\tau, \tau), \\ (\lambda_2, \mu_2, \rho_2) & (z \bmod m\delta, t \bmod n\tau) \in [m\delta, \delta) \times [0, n\tau) \cup [0, m\delta) \times [n\tau, \tau). \end{cases} \quad (2.3)$$

As these waves propagate through the material structure, they interact with various material interfaces, i.e., the spatial and temporal material property switches.

Across these interfaces, to maintain the material integrity, kinematic and dynamic compatibility conditions must be satisfied, these are as follows:

- *Kinematic Compatibility Condition*

- Purely Spatial Interface:

$$\left[ \frac{\partial u_i}{\partial t} \right]_1^2 = 0, \quad i = 1, 2, 3 \quad (2.4)$$

- Purely Temporal Interface

$$\left[ \frac{\partial u_i}{\partial z} \right]_1^2 = 0, \quad i = 1, 2, 3 \quad (2.5)$$

- *Dynamic Compatibility condition*

- Purely Spatial Interface:

Dilatational Wave	Shear Wave
$\left[ (\lambda + 2\mu) \frac{\partial u_3}{\partial z} \right]_1^2 = 0$	$\left[ \mu \frac{\partial u_i}{\partial z} \right]_1^2 = 0, \quad i = 1, 2$

- Purely Temporal Interface

$$\left[ \rho \frac{\partial u_i}{\partial t} \right]_1^2 = 0 \quad i = 1, 2, 3 \quad (2.7)$$

The main question asked was whether or not there exists a parameter range for which both type of waves would experience focusing for a specific checkerboard geometry, i.e., can both longitudinal and transverse waves be simultaneously in the plateau region described in references [26, 27]. To do this, we needed to find physically attainable conditions that eliminate reflections in both types of wave at the spatial and temporal interfaces. Because these disturbances are both governed by the wave equation with different wavespeeds, and so, these conditions amount to setting the respective wave impedances equal to one another

$$\begin{aligned} \rho_1 (\lambda_1 + 2\mu_1) &= \rho_2 (\lambda_2 + 2\mu_2), \\ \rho_1 \mu_1 &= \rho_2 \mu_2, \end{aligned}$$

which is the statement that wave impedances for both dilatation and shear waves must match. These conditions imply that

$$\frac{\lambda_2}{\lambda_1} = \frac{\mu_2}{\mu_1} = \frac{\rho_1}{\rho_2} = C \quad (2.8)$$

where  $C$  is the common ratio between the material parameters.

Physically, this equation means that in order to eliminate reflections in both types of wave, the elastic parameters must be inversely proportional to the inertial parameters. Further manipulation of this expression shows that materials 1 and 2 must have equal Poisson's ratio  $\nu = \frac{\lambda}{2(\lambda+\mu)}$ ,

$$\nu_2 = \frac{\lambda_2}{2(\lambda_2 + \mu_2)} = \frac{\lambda_1}{2(\lambda_1 + \mu_1)} = \nu_1. \quad (2.9)$$

From this relationship, we obtain an interesting relationship between the Poisson's ratio and the ratio of dilatation wave velocity to shear velocity. Specifically, we have that

$$\frac{c_d}{c_s} = \sqrt{\frac{2(1 - 2\nu)}{(1 - \nu)}}. \quad (2.10)$$

It is important to note that because  $\nu$  is identical in either material, this dependence implies that the ratio of dilatation wave velocity to shear wave velocity must also be identical between the two materials.

The Poisson's ratio of a stable, isotropic, linear elastic material will be greater than -1 or less than .5 because we require, respectively, that  $\lambda + 2\mu > 0$  and  $\mu > 0$ . Thus, we conclude  $\nu$  takes values in the interval  $(-1, \frac{1}{2})$ , and therefore, we conclude that the admissible ratio of the phase velocities takes values in the interval  $(\frac{2}{\sqrt{3}}, \infty)$ . Any two materials which have matching Poisson's ratio and with mass density inversely to the Poisson's ratio can support dual focusing of dilatation waves and shear waves. Perhaps the most striking conclusion of this is that when both types of wave are on the same plateau, they will travel at the same average velocity because they are on two parallel limit cycles. This can be clearly seen in Figure 2.1. This is in stark contrast to what happens in a "pure" material where longitudinal waves travel at a distinctly different (faster) velocity from shear waves. It is also shown numerically that there exists a wide parameter range where geometric focusing is observed for both types of wave, this can be seen clearly in Figure 4.5. Therefore, we conclude that the question posed at the beginning of section 2 has a positive answer, i.e., there most certainly exists a parameter range for which geometric focusing is observed in both types of wave.

Figure 2.1 also shows the mapping of the checkerboard geometry, characteristics, and resulting limit cycles onto the torus. This visualization clearly shows that the limit cycle is a closed trajectory on the torus. Another example of this is seen in



4.5. This mapping was first discussed in references [26, 27].

To verify the robustness of the plateau zone, we consider a different plot. Specifically, in Figure 2.4 we consider a different average velocity plot, specifically, we plot the average velocity for fixed values of  $m$  and  $n$  but for varying wave velocities  $a_1$  and  $a_2$ .

It is also possible to show that there are parameter ranges for which there is convergence in one type of wave but not in the other. With this idea, we envision the creation of a device that focuses on detecting or manipulating a specific wave family (either dilatation or shear), while leaving the other wave family untouched. This property could be useful in a variety of applications to sensors or detectors.

## 2.4 Conclusion

In this chapter we have shown that the checkerboard convergence phenomena can be extended for plane waves. Specifically, the  $\mathcal{C}_{20}$  plateau zones studied in reference [27] can be used to find regions of parameters that guarantee simultaneous focusing in shear and dilatation waves propagating through an elastic medium.

In the next chapter, we will discuss several extensions of the checkerboard geometry. These extensions will serve as examples of the robustness of the checkerboard focusing affect. Specifically, we will investigate the case of the so-called functionally graded (FG) checkerboard.

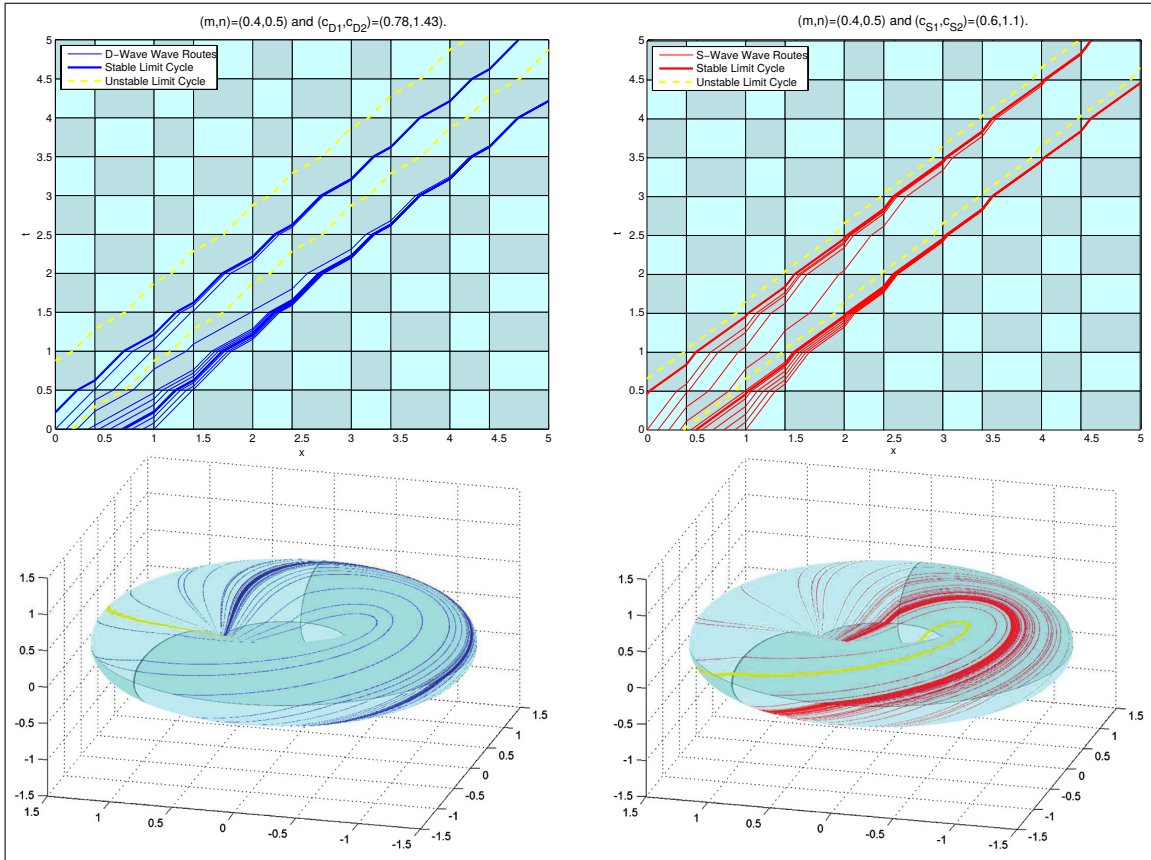


Figure 2.1: Simultaneous focusing in both dilatation and shear waves along with the mapping onto the torus.

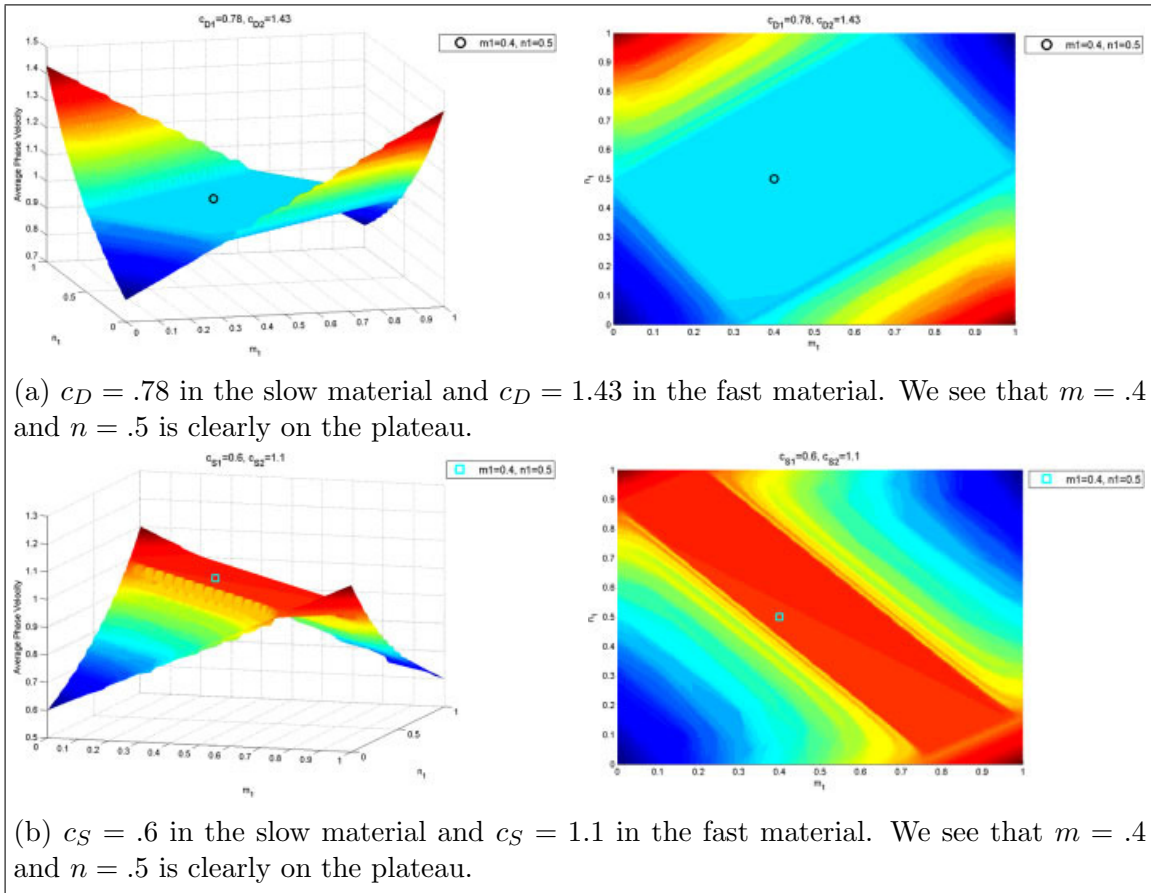


Figure 2.2: Plot of the average phase velocity of a characteristic after a large number of periods as a function of  $m$  and  $n$ . This demonstrates clearly the plateau region for each type of wave.

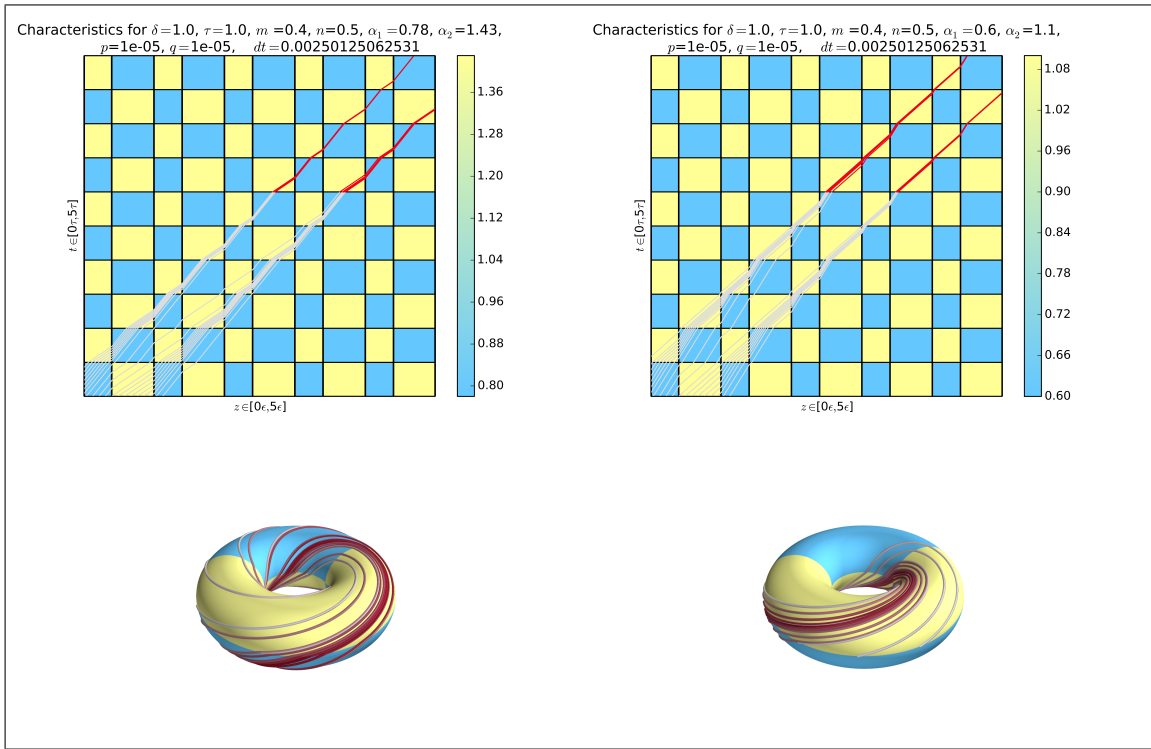


Figure 2.3: Plot of the average phase velocity of a characteristic after a large number of periods as a function of  $m$  and  $n$ . This demonstrates clearly the plateau region for each type of wave.

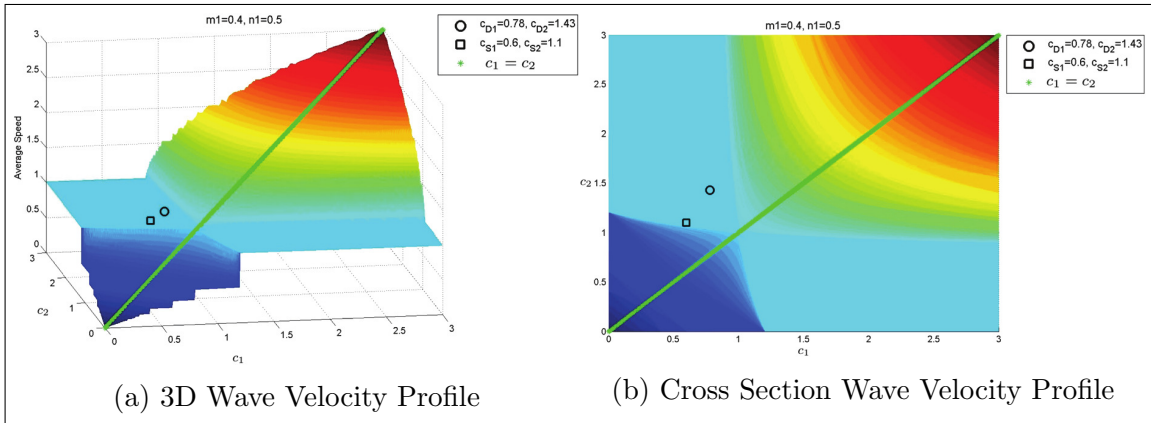


Figure 2.4: Plot of the average phase velocity as a function of phase velocity  $a_1$  and  $a_2$ . This demonstrates clearly the plateau region for each type of wave.

# Chapter 3

## Non-perfect Spatial Temporal Checkerboard

This chapter presents new theoretical results on generalizations to the perfect checkerboard geometry previously studied in references [26, 27, 25, 24]. In these papers, several assumptions were made that were important to achieving analytical results about wave propagation through these material geometries. These assumptions are valid under very specific circumstances, however, a major goal in this research effort is to investigate the robustness of DM phenomena. Specifically, we are interested in determining if the effects produced by the “perfect” spatial-temporal checkerboard material geometry persists under a variety of different “non-perfect” conditions.

Much of the original research on DM relies on the concept of a “perfect” structure in space-time. For a rectangular checkerboard [26], it is assumed that the checkerboard has discontinuous material interfaces and the materials composing the checkerboard have equal wave impedances. These conditions might be difficult to realize physically and should be relaxed. In the following two chapters, we show analytically and numerically that the previously discovered phenomena of characteristic focusing and energy accumulation are present even when the checkerboard interfaces are smooth or there is a mismatch in wave impedance. This is exciting, because it greatly increases potential applications of the checkerboard energy focusing effect and truly shows the wide applicability.

The main focus of Section 3.1 will be about an extension to the checkerboard termed a functionally graded (FG) checkerboard. Section 3.2 will take the concept of FG checkerboard and investigate a very specific type of FG checkerboard termed a linear FG checkerboard. In this section, exact bounds are found for how much the perfect structures can be spoiled while still achieving the desired effects of focusing and energy accumulation. Another generalization that has been investigated numerically is the checkerboard with mismatch in wave impedances.

## 3.1 Functionally Graded Dynamic Material

A natural relaxation to the standard checkerboard is a “smoothed” checkerboard, where the material change is continuous rather than discontinuous. We call this type of DM has been a functionally graded (FG) dynamic material. The motivation for this investigation comes from a basic argument that instantaneous change in material properties may not be physically realizable in time or in space, and so, the idea of a graded transition zone is more natural in many cases. An interesting question to ask is whether effects which are observed for the perfect checkerboard, such as energy accumulation and focusing, are also found to be present in a FG checkerboard. We will see that this is definitely the case.

### 3.1.1 Wave Route Calculation for FG material

In any hyperbolic problem, solving for the trajectory of characteristics is fundamental to understanding the behavior of the solution to the wave propagation problem. For the sharp checkerboard, wave route (characteristic) plotting was based on finding the exact trajectory for each characteristic. This task becomes more complicated with the introduction of transition regions between materials. Specifically, it is not possible to obtain an analytic expression for the trajectory through a general functionally graded material due to the possible complexity of the underlying nonlinear ordinary differential equation governing the characteristics. We decided to develop another method, specifically, we considered accurate numerical solution of the ordinary differential equations governing each characteristic.

For example, suppose we wish to solve the continuity equation in 1D,

$$\rho_t + (a(z, t)\rho)_z = 0, \quad (3.1)$$

where  $c(z, t)$  is the velocity profile. By introducing the potential function  $\phi$  by the equation  $\rho = \phi_z$ , and using the above equation we arrive at the following hyperbolic PDE for the Riemann invariant  $\phi(z, t)$ ,

$$\phi_t + a(z, t)\phi_z = 0. \quad (3.2)$$

To solve equation (3.2), we use the method of characteristics and solve for the curve  $z(t)$ . Specifically, this curve satisfies the following ODE:

$$\frac{dz}{dt} = a(z(t), t), \quad z(t_0) = z_0. \quad (3.3)$$

From the chain rule, we can interpret (3.2) as the total derivative of  $\phi$  along  $z(t)$ , so that  $\frac{d\phi}{dt} = 0$ , and thus  $\phi$  is constant along the characteristic paths. We applied the standard 4th order Runge-Kutta (RK-4) method to solve this ODE for characteristics.

Interestingly enough, this method not only generated characteristics for a smooth functionally graded DM, it was also found to successfully generate characteristic paths for a standard checkerboard with discontinuous interfaces, though, it is important to note that traditional error bounds relying on the derivative of the RHS of equation 3.3 do not hold under these relaxed regularity assumptions.

For the sake of discussion, assume that the following function governs the phase velocity distribution in spacetime,

$$a(z, t) = \left( \frac{a_1 + a_2}{2} \right) + \left( \frac{a_1 - a_2}{2} \right) \tanh \left( \frac{\sin \left( \frac{2\pi x}{\delta} \right)}{\alpha} \right) \tanh \left( \frac{\sin \left( \frac{2\pi t}{\tau} \right)}{\beta} \right). \quad (3.4)$$

Here,  $\alpha$  and  $\beta$  are, respectively, measures of the smoothness of the spatial and temporal interfaces. The phase velocity  $a(z, t)$  is chosen in this form so that we have control over the smoothness of the spatial-temporal interfaces, i.e., smaller  $\alpha$  or  $\beta$  give, respectively, sharper spatial or temporal interfaces. Specifically, as  $\alpha, \beta \rightarrow 0$ ,  $a(z, t)$  becomes the standard checkerboard given by

$$a(z, t) = \begin{cases} a_1, & (z \bmod m\delta, t \bmod n\tau) \in [0, m\delta) \times [0, n\tau) \cup [m\delta, \delta) \times [n\tau, \tau), \\ a_2, & (z \bmod m\delta, t \bmod n\tau) \in [m\delta, \delta) \times [0, n\tau) \cup [0, m\delta) \times [n\tau, \tau), \end{cases} \quad (3.5)$$

where  $m = \frac{1}{2}$  and  $n = \frac{1}{2}$ .

Using the characteristic plotting method discussed above, we will illustrate that the convergence phenomenon originally found for the perfect checkerboard persists for the FG checkerboard as well. To show this, we look at characteristic behavior for a perfect checkerboard and compare it to the FG checkerboard described by Equation (3.4) with spatial period  $\delta = 1$ , temporal period  $\tau = 1$ , geometrical parameters  $m = \frac{1}{2}, n = \frac{1}{2}$ , and phase velocities  $a_1 = 1.1, a_2 = 0.6$ . These parameters were chosen to guarantee convergence in the case of the perfect checkerboard.

Figure 3.1 directly shows the difference between the perfect and FG checkerboard over the course of one spatial-temporal period. In Figure 3.1a, the characteristics are plotted for a perfect checkerboard, i.e., in the limit as  $\alpha, \beta \rightarrow 0$ . In Figures 3.1b-3.1d, a functionally graded checkerboard is shown with wave speed  $a(z, t)$  given by equation (3.4), the main difference being the smooth transition regions between materials 1 and 2. Even though the characteristics in each subfigure are plotted over only one period, there is evidence that the characteristics are beginning to focus. Figure 3.2 definitively confirms that characteristic convergence to limit cycles is still possible. In this figure, the characteristics are plotted for the same cases as Figure 3.1, but they are plotted over ten spatial-temporal periods. In each case, it is clear that there is some amount of focusing. It is interesting to note that as the amount of smoothing increases, there appears to be a corresponding decrease in the amount of

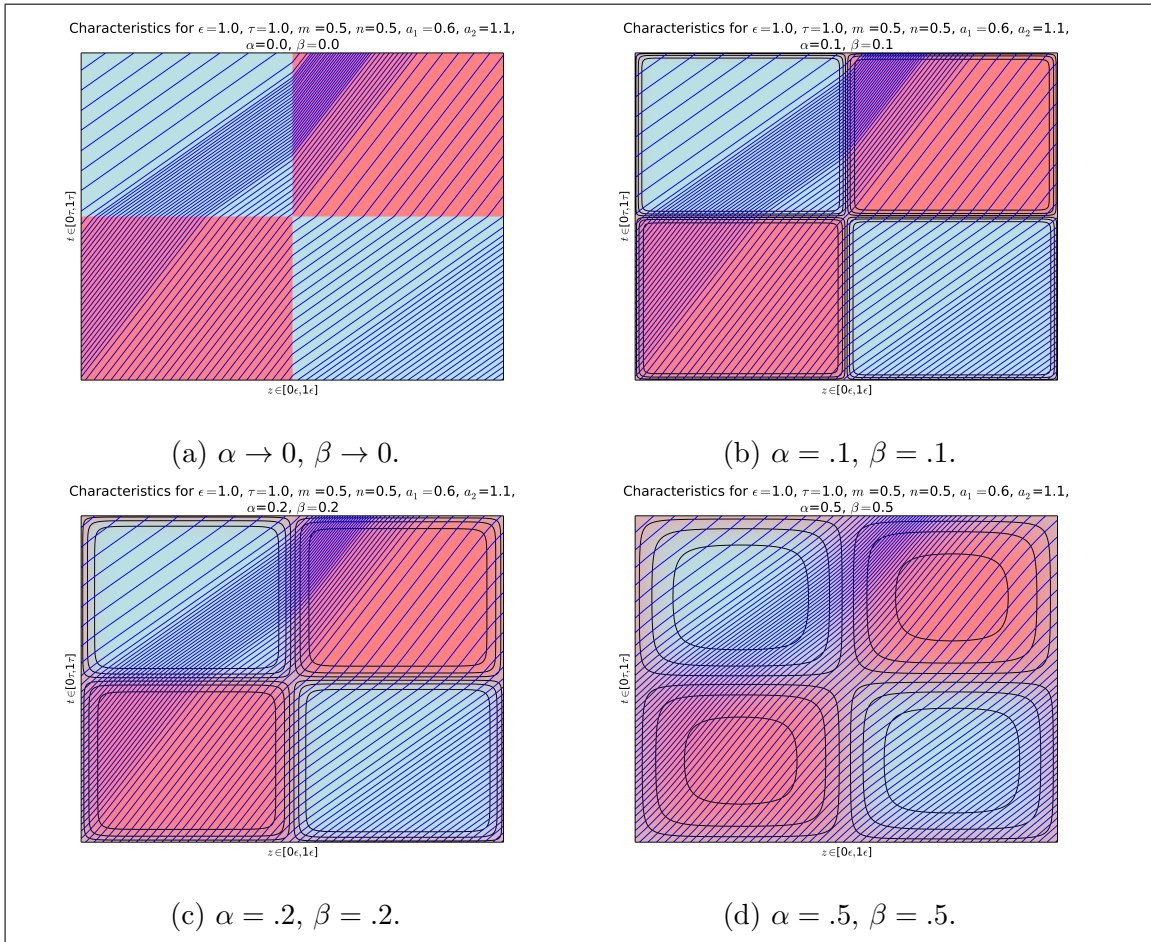


Figure 3.1: Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over one period.



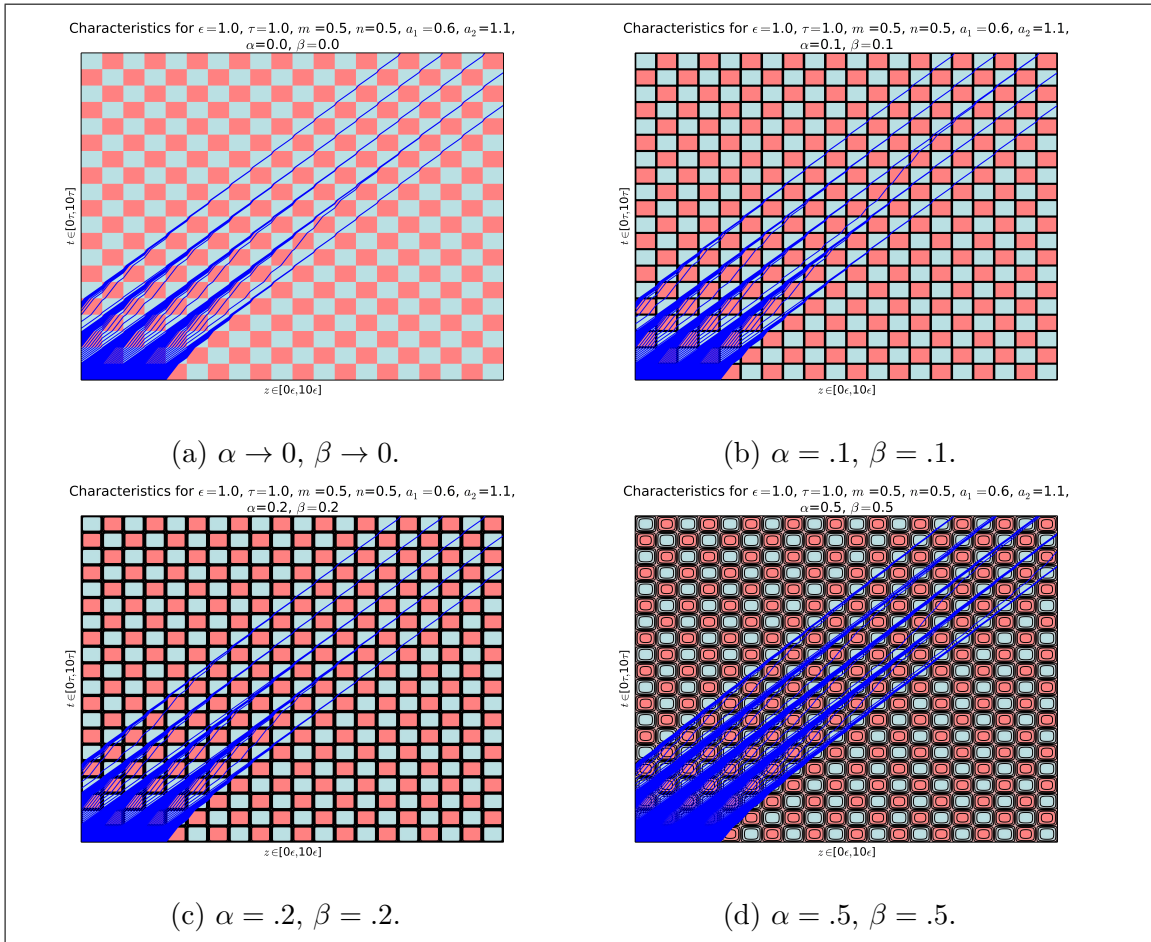


Figure 3.2: Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over ten periods.

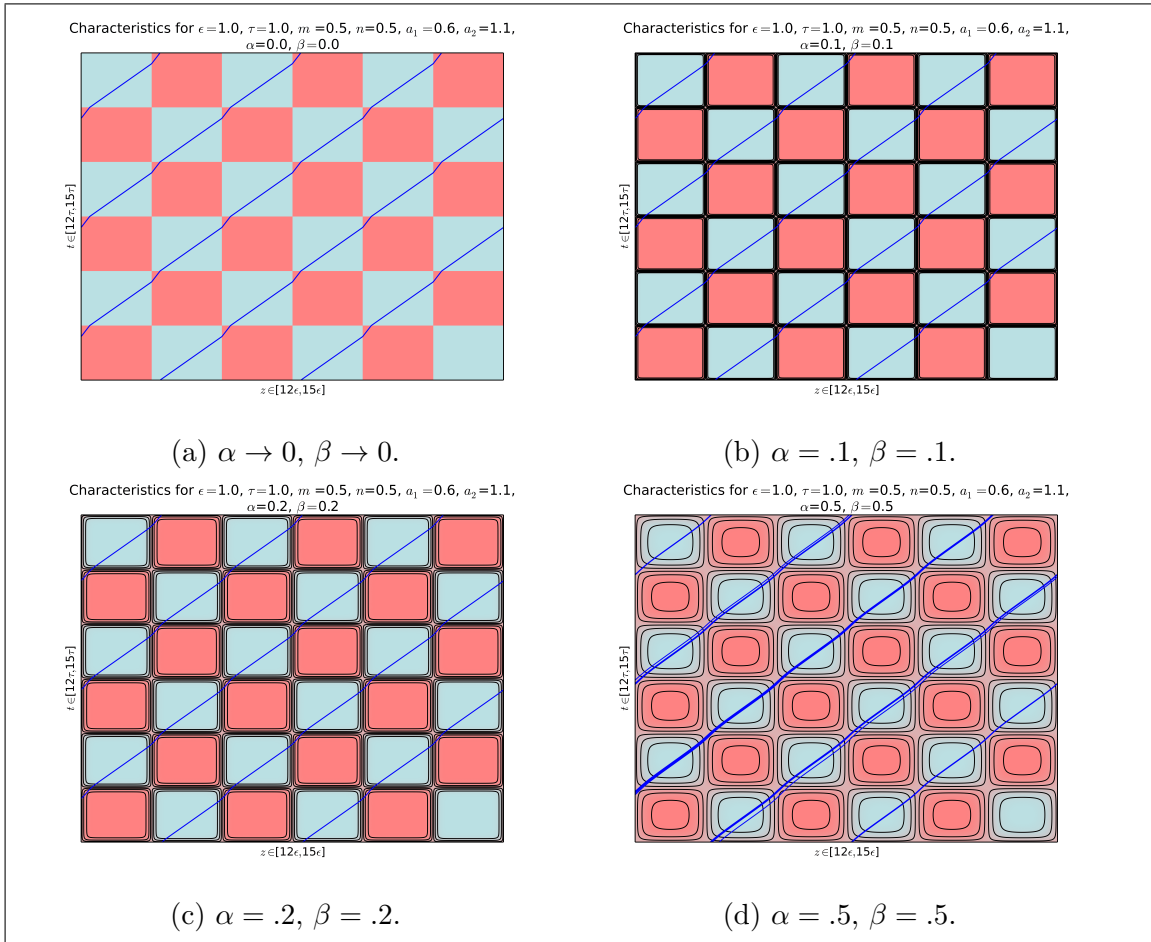


Figure 3.3: Comparison of characteristics traveling through a standard checkerboard and a FG checkerboard over 3 later periods. This shows that convergence persists up to a certain point.

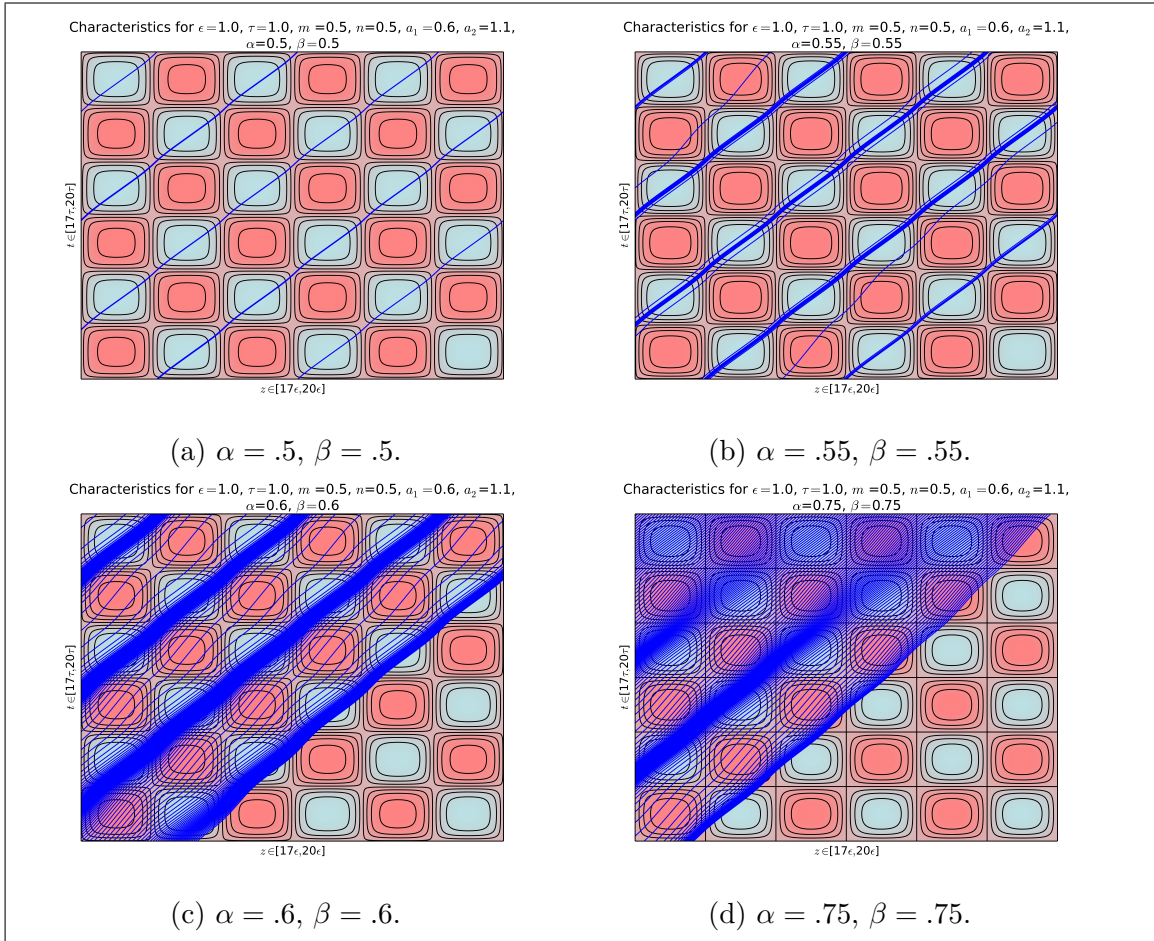


Figure 3.4: Comparison of characteristics traveling through a perfect checkerboard and a FG checkerboard over 3 later periods. This image shows that convergence disappears around .55 .

focusing. Perhaps the most convincing image is Figure 3.3 , in which, we show the same characteristics from Figure 3.1 and Figure 3.2 only the axes are restricted to the 12th spatial-temporal period to the 15th spatial-temporal period. This clearly shows that in each case, the characteristics are focused into distinct limit cycles. This example is special and serves to illustrate the robustness of the characteristic focusing effect.

The persistence of the convergence property exemplifies the robustness of the checkerboard structure, and further work should be done to evaluate how far this effect persists, i.e., how much smoothing it takes to destroy the characteristic focusing effect. To this end, we will be looking to extend some of the results in [27]. Thus, we want to establish how far one can smooth the checkerboard without eliminating the convergence phenomenon. Figure 3.4 shows an example of how the focusing effect disappears for smoothing parameters  $\alpha$  and  $\beta$  between 0.55 and 0.6. If we

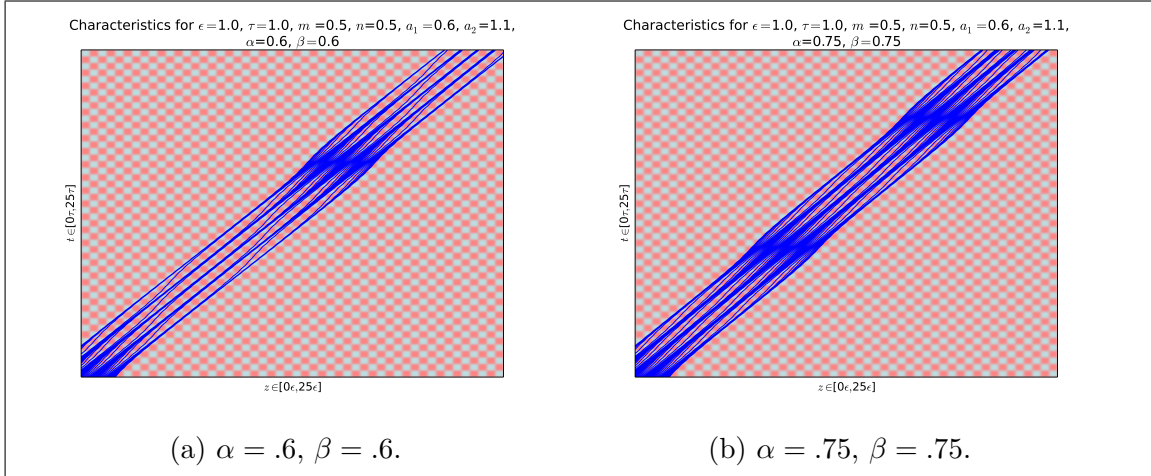


Figure 3.5: Oscillatory phenomena immediately after convergence is killed by smoothing.

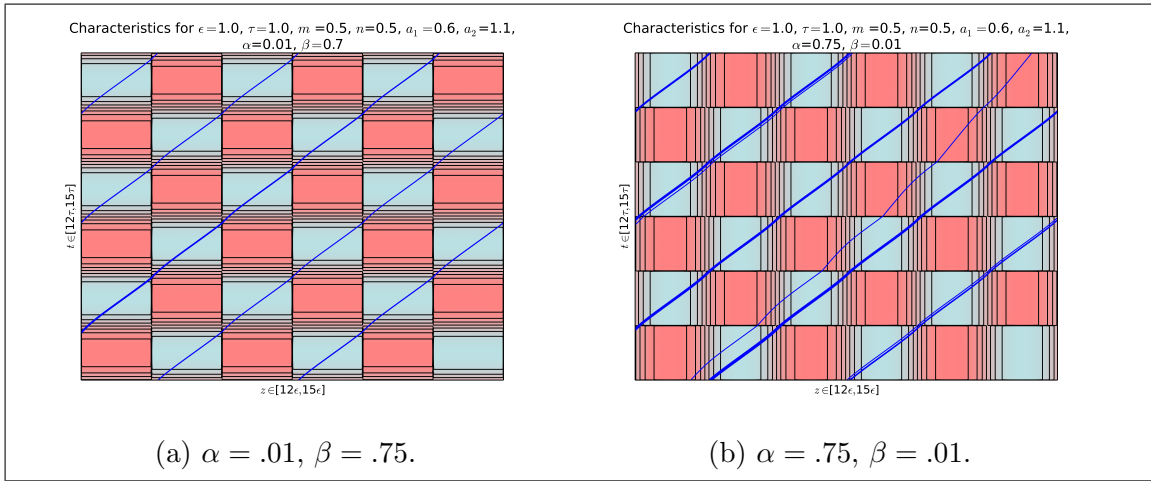


Figure 3.6: Example of smoothing in only one variable.

look more at what happens in this case, we see that there is an oscillatory regime (similar to the perfect checkerboard) for which the characteristics periodically tend to come together and then separate. This can be seen more clearly in Figure 3.5. We also consider smoothing in only one variable. Figure 3.6 shows that we can smooth either the spatial or the temporal interfaces substantially and still have convergence of the wave routes.

Section 3.2 will focus on derivation of new theoretical results for a very special kind of FG material termed a linear FG material. Specifically, in this section we will derive four new inequalities that give existence region for a specific type of limit cycle termed the  $\mathcal{C}^{p,q}$  limit cycle.

## 3.2 Linear FG Material

In this section we consider a special case of the FG graded checkerboard described in the last section. This will be termed a linear functionally graded (FG) checkerboard. The linear FG is a perfect checkerboard for which the sharp interfaces are replaced with gradual transition regions, in which, we assume that the material parameters are chosen to be a linear function of space and time. With this choice, it becomes possible to analytically determine the path of a specific characteristic, thereby allowing us to analytically study a possible focusing effect in this regime.

It is important to emphasize that the quantity that possesses “linear” change is the wave velocity  $a$ . This is important to emphasize, because it means that the material properties may necessarily display non-linear change in the region. For example, if we assume that the wave impedance  $\gamma$  is constant, then in a region where  $a$  is a linear function of space, i.e., if  $a(z, t) = Cz + D$ , then the stiffness is also a linear function of space  $k = \gamma Cz + \gamma D$ , however, the mass density is nonlinear, i.e., it is given by the expression,  $\rho = \gamma/(Cz + D)$ , which is a non-linear function of  $z$ .

We consider regions of linear grading from material 1 to material 2. One spatial-temporal period of a linear FG checkerboard is shown in Figure 3.7. In Figure 3.7, we denote the spatial period as  $\delta$ , the spatial volume fraction as  $m \in [0, 1]$ , the temporal period as  $\tau$ , and the temporal volume fraction as  $n \in [0, 1]$ , the wave velocity in each material is shown to be in the range on the right. The width of each of the spatial transition region is given by  $2p\delta$  and the width of the temporal transition region is  $2q\tau$ . It is clear that for this description to make sense, we must have that  $p < \frac{\min\{m, (1-m)\}}{2}$  and  $q < \frac{\min\{n, (1-n)\}}{2}$  which will guarantee that the amount of the transition region does not exceed the size of the smallest checkerboard section.

At this point it is convenient to non-dimensionalize. We introduce nondimensional variables  $\tilde{z} = \frac{z}{\delta}$ ,  $\tilde{t} = \frac{t}{\tau}$  and  $\tilde{a} = a\frac{\tau}{\delta}$ . With this convention introduced, we will examine what the wave velocity will be in each region. It is clear that in the constant regions, the wave speed will just be equal to either  $a_1$  or  $a_2$  depending on whether the characteristic is in region 1 or region 2, respectively.

Due to the double periodicity, we only need to define the wave velocity in one spatial-temporal period  $(\tilde{z}, \tilde{t}) \in [0, 1] \times [0, 1]$ . In the regions  $(\tilde{z}, \tilde{t}) \in [p, m - p] \cup [m + p, 1 - p] \times [q, n - q] \cup [n + q, 1 - q]$  the wave velocity is given by a piecewise function similar to the original checkerboard.

$$a(z, t) = \begin{cases} a_2, & (\tilde{z}, \tilde{t}) \in ([p, m - p] \times [q, n - q]) \cup ([m + p, 1 - p] \times [n + q, 1 - q]), \\ a_1, & (\tilde{z}, \tilde{t}) \in ([p, m - p] \times [n + q, 1 - q]) \cup ([m + p, 1 - p] \times [q, n - q]). \end{cases} \quad (3.6)$$

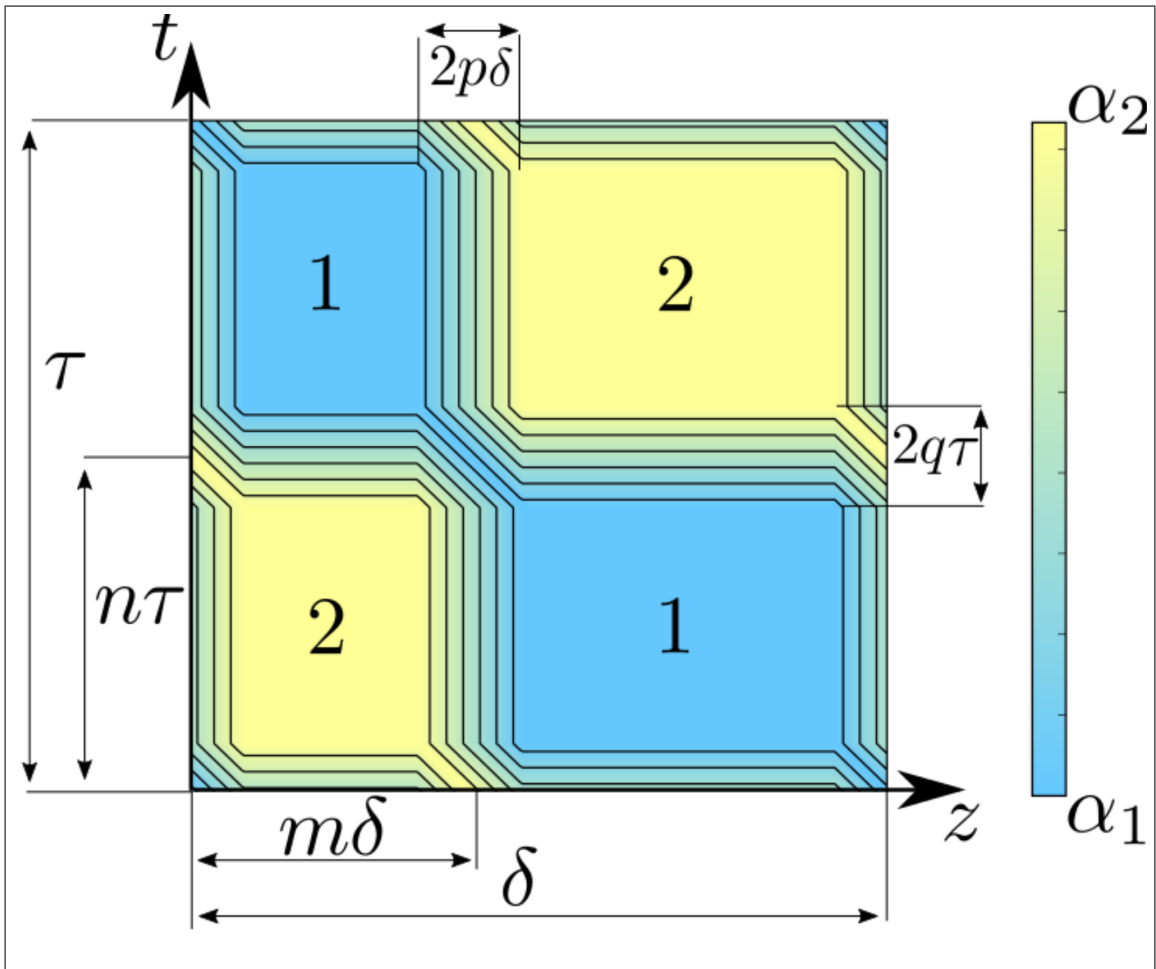


Figure 3.7: Linear FG Checkerboard over one spatial-temporal period.

As for the transition regions, we have three separate wave speed patterns that must be described. The first case is the case of linear transition in space alone, e.g., if  $(\tilde{z}, \tilde{t}) \in [m - p, m + p] \times [q, n - q]$ , we have that

$$\tilde{a}(z, t) = \tilde{a}_2 + \frac{(\tilde{a}_1 - \tilde{a}_2)}{2p}(\tilde{z} - m + p). \quad (3.7)$$

Another alternative is the case of linear transition in time alone, e.g., if  $(\tilde{z}, \tilde{t}) \in [p, m - p] \times [n - q, n + q]$ , we have that

$$\tilde{a}(z, t) = \tilde{a}_1 + \frac{(\tilde{a}_2 - \tilde{a}_1)}{2q}(t - n + q). \quad (3.8)$$

Finally, we will describe wave velocity being variable in space and in time must be discussed, e.g., if  $(\tilde{z}, \tilde{t}) \in [m - p, m + p] \times [n - q, n + q]$ , we have that

$$\tilde{a}(z, t) = \begin{cases} \tilde{a}_2 + \frac{(\tilde{a}_1 - \tilde{a}_2)}{2p}(\tilde{z} - m + p) + \frac{(\tilde{a}_1 - \tilde{a}_2)}{2q}(\tilde{t} - n + q), & \tilde{z} - m + p \leq s(t - m - q), \\ \tilde{a}_2 - \frac{(\tilde{a}_1 - \tilde{a}_2)}{2p}(\tilde{z} - m - p) - \frac{(\tilde{a}_1 - \tilde{a}_2)}{2q}(\tilde{t} - n - q), & \tilde{z} - m + p > s(t - n - q) \end{cases} \quad (3.9)$$

where  $s = -p/q$ . This gives rise to a rectangular region that is cut along one of the diagonals and will be termed the ‘‘roof’’ region (regardless of whether it takes the shape of a roof or a valley). For brevity, we have omitted the other linear regions, the above explanations serve to clarify exactly what we mean by linear in each region.

The derivatives of this linear function certainly jump, however, the function itself is continuous and the characteristic differential equation can certainly be integrated in any of these regions. The differential equation governing characteristics is

$$\frac{d\tilde{z}}{dt} = \tilde{a}(\tilde{z}(\tilde{t}), \tilde{t}), \quad \tilde{t} \in \mathbb{R}^+, \quad z(0) = z_0. \quad (3.10)$$

We will study limit cycles that are generalizations of limit cycles from the class of characteristics  $\mathcal{C}_{z_0}$ , first described in the paper [27]. A characteristic in this class is defined based on how it propagates through the perfect checkerboard structure. Specifically, a characteristic in class  $\mathcal{C}_{z_0}$  starts at the point  $z_0 \in [0, m\epsilon]$  and alternates between intersecting vertical and horizontal interfaces. This type of limit cycle can be seen in Figure 1.5. In this thesis, we define a related class of characteristics  $\mathcal{C}_{z_0}^{p,q}$ . These are paths that closely resemble characteristics in the class  $\mathcal{C}_{z_0}$ . Specifically, we say that a characteristic is in class  $\mathcal{C}_{z_0}^{p,q}$  if it begins at the point  $\tilde{z}_0 \in [p, m - p]$  and subsequently alternates between vertical and horizontal interfaces. We emphasize that if it intersects a vertical interface at some point  $\tilde{t}^*$  this time must not allow the characteristic to enter into the rectangular ‘‘roof’’ region described before, i.e.,  $\tilde{t}^* \bmod 1 \in (q, n - q) \cup (n + q, 1 - q)$ . Additionally, if the characteristic intersects a horizontal interface at some point  $\tilde{z}^*$  then it also must not be allowed to enter into

the roof region, so we require  $\tilde{z}^* \bmod 1 \in (p, m - p) \cup (m + p, 1 - p)$ . It is important to mention that limit cycles not obeying these constraints do exist, however, this different type of characteristic behavior that present its own unique challenges. We restrict our attention to the study of limit cycles in the class  $\mathcal{C}_{z_0}^{p,q}$ . In the presence of a  $\mathcal{C}_{z_0}$  limit cycle in a sharp checkerboard we can always introduce a small amount of smoothing to give a  $\mathcal{C}^{p,q}$  limit cycle. If a non  $\mathcal{C}^{p,q}$  exists, i.e., the roof region is entered, we can prevent entrance into a roof region can typically be achieved by shrinking the amount of smoothing to ensure origination outside of the roof region (i.e., by shrinking  $p$  and  $q$ ). We define a limit cycle in this class as a path that repeats itself after a number of periods. In this thesis we will restrict study to limit cycles that repeat themselves every period. This implies that the average speed is  $\frac{\delta}{\tau}$ . We can always obtain a  $\mathcal{C}_{z_0}^{p,q}$  limit cycle by starting with  $\mathcal{C}_{z_0}$  limit cycle and introducing a small amount of smoothing. As long as the limit cycle does not originate on a corner, we can always choose a small enough amount of smoothing that the characteristic does not intersect any of the roof regions around the corner points.

In a similar manner to [27], we will derive exact linear inequalities that give regions of convergence in the parameter space for  $\mathcal{C}_{z_0}^{p,q}$  type limit cycles. These regions were termed “plateau” regions [26], due to the plateau shaped structure that appears in plots of the average phase velocity over regions in which characteristics convergence is observed.

Figure 3.8 shows the specific type of limit cycle we will be studying. It obeys all of the constraints imposed earlier, and by definition, belongs to class  $\mathcal{C}_{z_0}^{p,q}$ . Define  $w_i$  to be the distance between the  $i$ -th vertical checkerboard interface with the characteristics intersection with the  $i$ -th temporal checkerboard interface, see Figure 3.8. We will investigate the 1 periodic limit cycle. We will derive a general formula for  $w_3$  as a function of  $w_1$ . A limit cycle is a path for which  $w_1 = w_3$ . We will number the intersection points of the red  $\mathcal{C}_{z_0}^{p,q}$  characteristic path from Figure 3.8 and denote the set of them as  $\mathcal{I} = \{(z_i, t_i) : i = 1, \dots, 9\}$ . These inequalities will be described in detail in Section 3.2.2. In the next subsection, we will show how the characteristic equation is solved in the linear regions.

### 3.2.1 Characteristic in Linear Region

In this subsection we show how the characteristic path equation (3.10) is integrated in each linear section.

**Temporally Linear Region:** In Figure 3.9, we show a temporally linear transition zone. The wave velocity in this medium is given by the following function of space



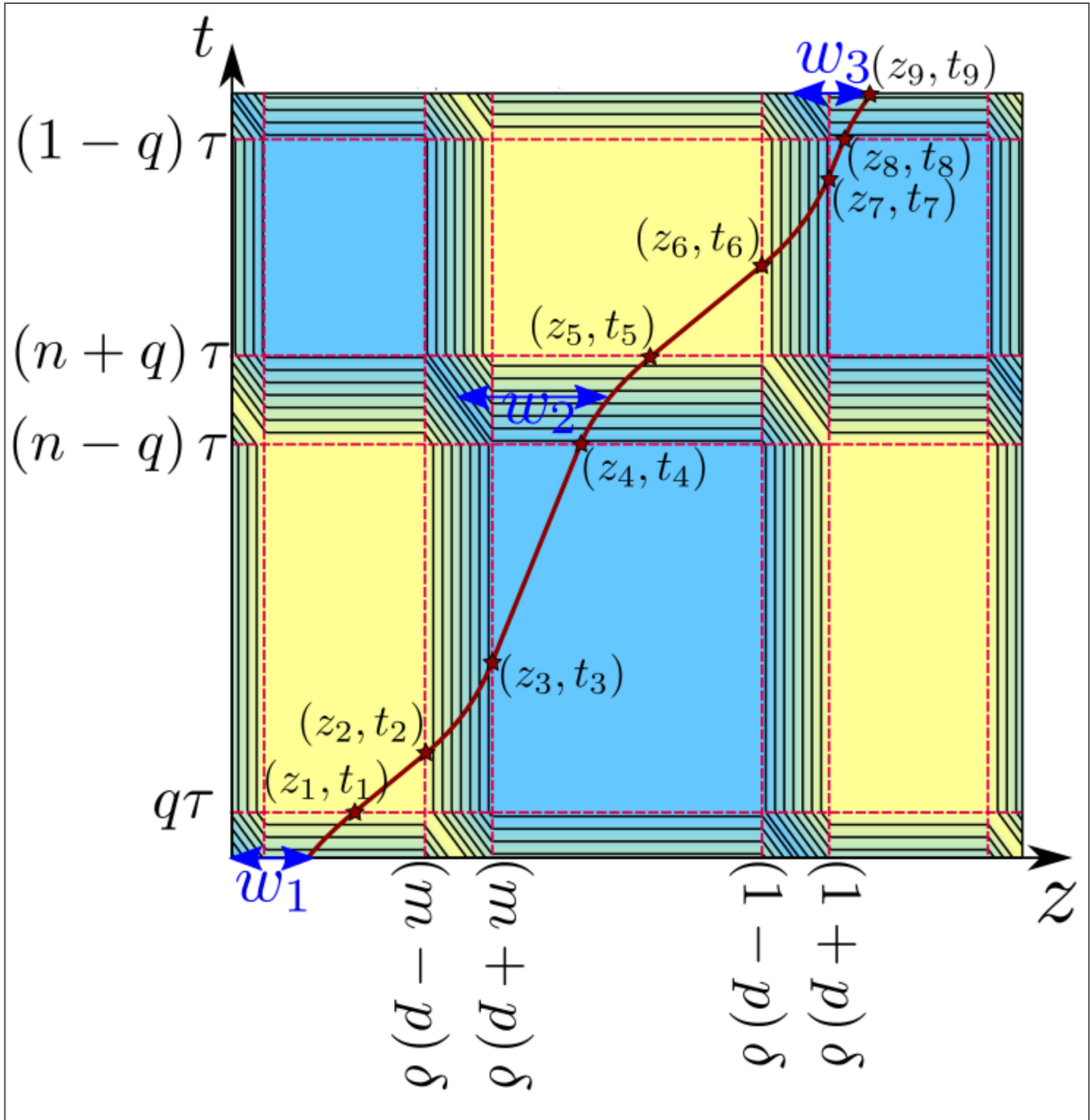


Figure 3.8: Specific type of limit cycle studied.

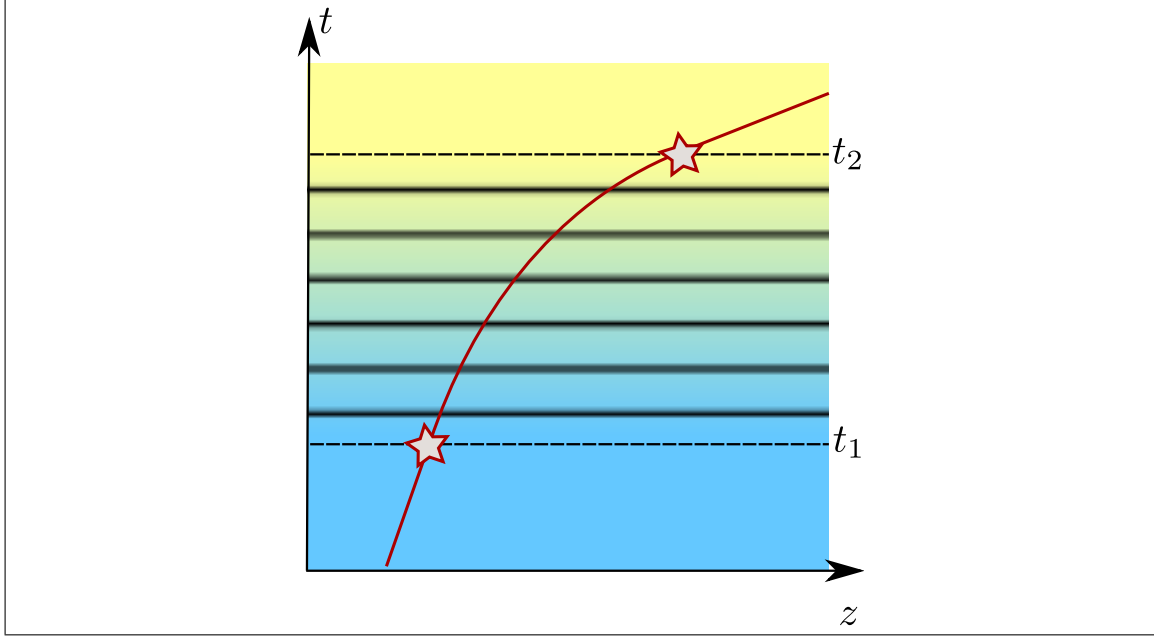


Figure 3.9: Temporally Linear Transition from Material 1 to Material 2

and time.

$$\begin{aligned}
 a(z, t) &= a_1 + \left( \frac{a_2 - a_1}{t_2 - t_1} \right) (t - t_1), \\
 &= \left( \frac{a_2 - a_1}{t_2 - t_1} \right) t + \left( \frac{a_1(t_2 - t_1) - t_1(a_2 - a_1)}{t_2 - t_1} \right), \\
 &= \left( \frac{a_2 - a_1}{t_2 - t_1} \right) t + \left( \frac{a_1 t_2 - t_1 a_2}{t_2 - t_1} \right), \\
 &= m_H t + d_H,
 \end{aligned} \tag{3.11}$$

where  $m_H = \frac{a_2 - a_1}{t_2 - t_1}$  and  $d_H = \frac{a_1 t_2 - t_1 a_2}{t_2 - t_1}$ .

Thus, ordinary differential equation governing the characteristic (equation (3.10)) is given by

$$z' = m_H t + d_H.$$

Integration of this equation gives that

$$\begin{aligned}
 z(t) &= \frac{m_H}{2} t^2 + d_H t + C_H, \\
 &= \left( \frac{m_H}{2} t + d_H \right) t + C_H.
 \end{aligned}$$

From this, we see that  $z_1 = \left(\frac{m_H}{2}t_1 + d_H\right)t_1 + C_H$  which allows us to solve for  $C_H = z_1 - \left(\frac{m_H}{2}t_1 + d_H\right)t_1$ ,

$$\begin{aligned}
z(t) &= \left(\frac{m_H}{2}t + d_H\right)t + z_1 - \left(\frac{m_H}{2}t_1 + d_H\right)t_1, \\
&= z_1 + \frac{m_H}{2}(t^2 - t_1^2) + d_H(t - t_1), \\
&= z_1 + \frac{m_H}{2}(t - t_1)(t + t_1) + d_H(t - t_1), \\
&= z_1 + \left(\frac{m_H}{2}(t + t_1) + d_H\right)(t - t_1), \\
&= z_1 + \left((a_2 - a_1)\frac{(t + t_1)}{2} + a_1t_2 - t_1a_2\right)\frac{(t - t_1)}{t_2 - t_1}.
\end{aligned}$$

**Spatially Linear Region:** In Figure 3.10, there is a purely spatial lamination. The wave velocity in this medium is given by

$$\begin{aligned}
a(z, t) &= a_1 + \left(\frac{a_2 - a_1}{z_2 - z_1}\right)(z - z_1), \\
&= \left(\frac{a_2 - a_1}{z_2 - z_1}\right)z + \left(\frac{a_1(z_2 - z_1) - z_1(a_2 - a_1)}{z_2 - z_1}\right), \\
&= \left(\frac{a_2 - a_1}{z_2 - z_1}\right)z + \left(\frac{a_1z_2 - z_1a_2}{z_2 - z_1}\right), \\
&= m_Vt + d_V,
\end{aligned} \tag{3.12}$$

where  $m_V = \frac{a_2 - a_1}{z_2 - z_1}$  and  $d_V = \frac{a_1z_2 - z_1a_2}{z_2 - z_1}$ .

In this case, equation (3.10) is given by

$$z' = m_Vz + d_V.$$

Rearranging the above equation gives

$$\frac{z'}{m_Vz + d_V} = 1.$$

Integration of this equation gives that

$$(1/m_V) \ln(m_Vz + d_V) = t + C_V. \tag{3.13}$$

Which can be inverted to find

$$z(t) = \frac{D_V}{m_V} \exp(m_Vt) - \frac{d_V}{m_V}. \tag{3.14}$$

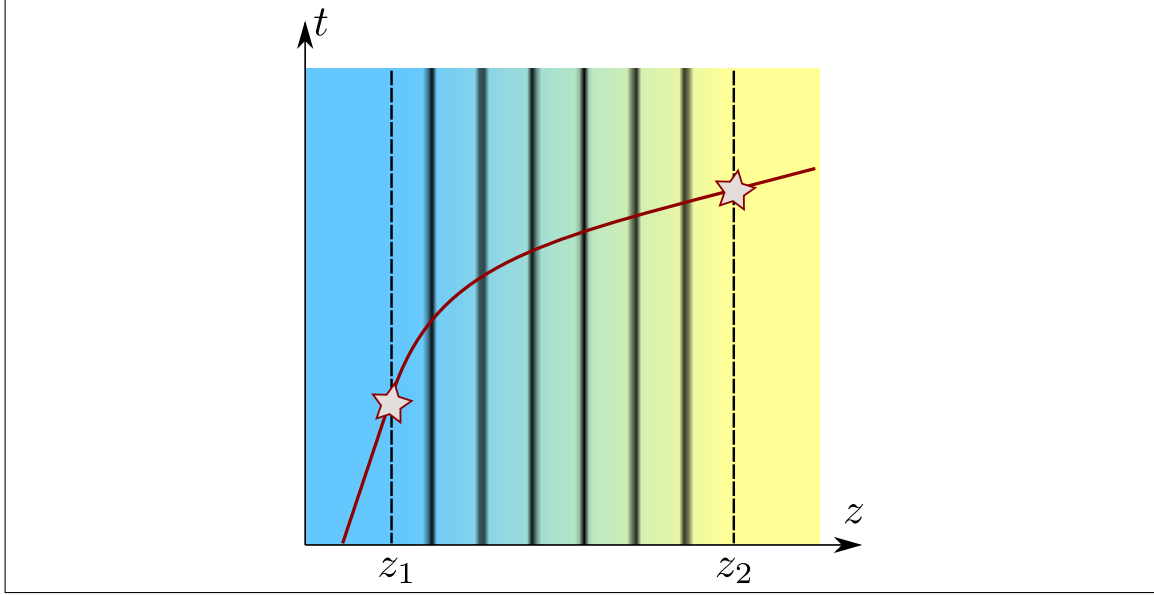


Figure 3.10: Spatially Linear Transition from Material 1 to Material 2

From this, we see that  $z_* = \frac{D_V}{m_V} \exp(m_V t_*) - \frac{d_V}{m_V}$  which allows us to solve for  $D_V = (m_V z_* + d_V) \exp(-m_V t_*)$ , giving the final expression for  $z(t)$  as

$$\begin{aligned} z(t) &= \frac{(m_V z_* + d_V)}{m_V} \exp(m_V(t - t_*)) - \frac{d_V}{m_V}, \\ &= \left( z_* + \frac{d_V}{m_V} \right) \exp(m_V(t - t_*)) - \frac{d_V}{m_V}. \end{aligned}$$

In terms of parameters, we have that

$$z(t) = \left( z_* + \frac{a_1 z_2 - a_2 z_1}{a_2 - a_1} \right) e^{\frac{a_2 - a_1}{z_2 - z_1}(t - t_*)} - \frac{a_1 z_2 - a_2 z_1}{a_2 - a_1}.$$

### 3.2.2 Inequality Constraints

The first condition that must be obeyed is on the smoothing parameters  $p$  and  $q$ . Specifically, we require that the width of both the spatial and temporal transition zones must be smaller than half the minimum width in the checkerboard geometry:

$$q < \frac{\min\{n, (1 - n)\}}{2}, \quad p < \frac{\min\{m, (1 - m)\}}{2}. \quad (3.15)$$

This is the maximum range of smoothing that we can add and still be consistent with the geometry shown in Figure 3.7. We will place a further restrictions on the smoothing in the following exposition.

We assume that the limit cycle starts in the first spatial period. This assumption leads to the inequality

$$p < \tilde{w}_1 < \tilde{z}_1. \quad (3.16)$$

Next the characteristic travels through a temporally linear region to point  $(z_1, t_1)$ . The location of this point is obtained by integration of equation (3.10) and is given by

$$\tilde{z}_1 = \tilde{w}_1 + \left[ \frac{1}{4}\tilde{a}_1 + \frac{3}{4}\tilde{a}_2 \right] q. \quad (3.17)$$

The details of this calculation can be found in section 3.2.1. From the constraints discussed earlier, this point must also obey

$$\tilde{w}_1 < \tilde{z}_1 < m - p. \quad (3.18)$$

Next the limit cycle travels through a pure material to  $(z_2, t_2)$ . Direct integration of equation (3.10) gives that

$$\tilde{t}_2 = q + \frac{1}{\tilde{a}_2} (m - p - \tilde{z}_1). \quad (3.19)$$

Due to the constraints discussed earlier, this point must obey the inequality

$$q < \tilde{t}_2 < \tilde{t}_3. \quad (3.20)$$

The limit cycle now travels through a spatially linear region to the point  $(z_3, t_3)$ . The location of this point is obtained by integration of equation (3.10) and is given by

$$\tilde{t}_3 = \tilde{t}_2 + \frac{2p}{(\tilde{a}_1 - \tilde{a}_2)} \ln \left( \frac{\tilde{a}_1}{\tilde{a}_2} \right). \quad (3.21)$$

The details of this calculation can be found in section 3.2.1. From the constraints discussed earlier, this point must obey

$$\tilde{t}_2 < \tilde{t}_3 < n - q. \quad (3.22)$$

Next the limit cycle travels through the pure material to point  $(z_4, t_4)$ . Direct integration of equation (3.10) gives that

$$\tilde{z}_4 = m + p + \tilde{a}_1 (n - q - \tilde{t}_3). \quad (3.23)$$

Due to the constraints discussed earlier, this point must obey the inequality

$$m < \tilde{z}_4 < \tilde{z}_5. \quad (3.24)$$

Next the characteristic travels through a temporally linear region to point  $(z_5, t_5)$ . The location of this point is obtained by integration of equation (3.10) and is given by

$$\tilde{z}_5 = \tilde{z}_4 + (\tilde{a}_1 + \tilde{a}_2) q. \quad (3.25)$$

The details of this calculation can be found in section 3.2.1. From the constraints discussed earlier, this point must obey

$$\tilde{z}_4 < \tilde{z}_5 < 1 - p. \quad (3.26)$$

Next the limit cycle travels through the pure material to  $(z_6, t_6)$ . Direct integration gives that

$$\tilde{t}_6 = n + q + \frac{1}{\tilde{a}_2} (1 - p - \tilde{z}_5). \quad (3.27)$$

From the constraints discussed earlier, this point must satisfy the following inequality

$$n + q < \tilde{t}_6 < \tilde{t}_7. \quad (3.28)$$

Next the limit cycle travels through the spatially linear material to the point  $(z_7, t_7)$ . The location of this point is obtained by integration of equation (3.10) and is given by:

$$\tilde{t}_7 = \tilde{t}_6 + \frac{2p}{(\tilde{a}_1 - \tilde{a}_2)} \ln \left( \frac{\tilde{a}_1}{\tilde{a}_2} \right). \quad (3.29)$$

This point must satisfy the following condition

$$\tilde{t}_6 < \tilde{t}_7 < 1 - q. \quad (3.30)$$

Next the limit cycle travels through the pure material to the point  $(z_8, t_8)$ , given by

$$\tilde{z}_8 = 1 + p + \tilde{a}_1 (1 - q - \tilde{t}_7), \quad (3.31)$$

This point must satisfy the following condition

$$1 + p < \tilde{z}_8 < \tilde{z}_9. \quad (3.32)$$

Lastly, the limit cycle travels through the temporally linear material to the final point  $(z_9, t_9)$ , given by

$$\tilde{z}_9 = \tilde{z}_8 + \left[ \frac{3}{4} \tilde{a}_1 + \frac{1}{4} \tilde{a}_2 \right] q, \quad (3.33)$$

which must obey

$$\tilde{z}_8 < \tilde{z}_9 < (1 + m) - p. \quad (3.34)$$

By substitution of the preceding expressions into one another, we arrive at a relationship between  $w_3$  and  $w_1$ , specifically, we have that

$$\begin{aligned} \tilde{w}_3 = & \frac{1}{\lambda^2} \tilde{w}_1 + \frac{\lambda - 1}{\lambda^2} m + \frac{\tilde{a}_2(1 - \lambda)}{\lambda^2} n \\ & + \frac{(\lambda + 1)(\lambda^2 - 2\lambda \ln(\lambda) - 1)}{\lambda^2(\lambda - 1)} p + \frac{a_2(\lambda + 1)(\lambda - 1)^2}{4\lambda^3} q + \frac{a_2 - 1}{\lambda}, \end{aligned} \quad (3.35)$$

where we have let  $\lambda = \frac{\tilde{a}_2}{\tilde{a}_1}$  for simplification.

Equation (3.35) is an expression for the final location of the characteristic for any  $\mathcal{C}_{z_0}^{p,q}$  characteristic as a function of its starting location. Therefore, to solve for the location of a period 1  $\mathcal{C}_{z_0}^{p,q}$  limit cycle, set  $\tilde{w}_3 = \tilde{w}_1$  and solve for  $\tilde{w}_1$ . We will denote this solution  $(\tilde{w}_1)_{LC}$ , we find that

$$\begin{aligned} (\tilde{w}_1)_{LC} = & \lambda \frac{(\tilde{a}_2 - 1)}{(\lambda^2 - 1)} + \left( \frac{1}{\lambda + 1} \right) m + \left( \frac{-\tilde{a}_2}{\lambda + 1} \right) n \\ & + \left( \frac{\lambda^2 - 2\lambda \ln(\lambda) - 1}{\lambda^2 - 2\lambda + 1} \right) p + \left( \frac{a_2(\lambda - 1)}{4\lambda} \right) q, \end{aligned} \quad (3.36)$$

which gives an exact expression for the starting location of a limit cycle in class  $\mathcal{C}_{z_0}^{p,q}$ .

Overall, we have 18 separate inequalities, which in addition to inequalities (3.15), must be satisfied by the geometrical and material parameters. After substitution of all of the  $z_i$ 's and the  $t_i$ 's into the above inequalities, we see that they are linear in  $m$ ,  $n$ ,  $p$ , and  $q$ , and they can be written in the following form

$$B_i m + A_i n + C_i p + D_i q + E_i < 0 \quad \text{where, } i = 1, \dots, 22, \quad (3.37)$$

and the coefficients are listed in Table 3.1. In this table, we have defined the functions  $f, g$ , and  $h$  for notational convenience as

$$f(\lambda) = \lambda \ln(\lambda) - \lambda + 1, \quad (3.38)$$

$$g(\lambda) = \lambda - \ln(\lambda) - 1, \quad (3.39)$$

$$h(\lambda) = -\lambda^2 + 2\lambda \ln(\lambda) + 1. \quad (3.40)$$

Inequalities 1, 4, 5, 7, 10, 11, 14, 15, and 18 are automatically satisfied for any value of the material parameters. We rewrite these inequalities the same manner

$i$	$B_i$	$A_i$	$C_i$	$D_i$	$E_i$
1	0	0	0	$\frac{-a_2(3\lambda+1)}{4\lambda}$	0
2	$\frac{-\lambda}{\lambda+1}$	$\frac{-\tilde{a}_2}{\lambda+1}$	$\frac{2\lambda g(\lambda)}{(\lambda-1)^2}$	$\tilde{a}_2$	$\frac{\lambda(\tilde{a}_2-1)}{\lambda^2-1}$
3	$\frac{-\lambda}{\tilde{a}_2(\lambda+1)}$	$\frac{-1}{\lambda+1}$	$\frac{2\lambda g(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	1	$\frac{\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda^2-1)}$
4	0	0	0	$\frac{-2\lambda \ln(\lambda)}{a_2(\lambda-1)}$	0
5	0	0	$\frac{-2\lambda \ln(\lambda)}{a_2(\lambda-1)}$	0	0
6	$\frac{\lambda}{\tilde{a}_2(\lambda+1)}$	$\frac{-\lambda}{\lambda+1}$	$\frac{2\lambda f(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	1	$\frac{-\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda^2-1)}$
7	0	0	0	$-a_2(1 + \frac{1}{\lambda})$	0
8	$\frac{\lambda}{\lambda+1}$	$\frac{\tilde{a}_2}{\lambda+1}$	$\frac{2\lambda g(\lambda)}{(\lambda-1)^2}$	$\tilde{a}_2$	$\frac{\tilde{a}_2-\lambda^2}{\lambda^2-1}$
9	$\frac{\lambda}{\tilde{a}_2(\lambda+1)}$	$\frac{1}{\lambda+1}$	$\frac{2\lambda g(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	1	$\frac{\tilde{a}_2-\lambda^2}{\tilde{a}_2(\lambda^2-1)}$
10	0	0	$\frac{-2\lambda \ln(\lambda)}{\tilde{a}_2(\lambda-1)}$	0	0
11	0	0	$\frac{-2\lambda \ln(\lambda)}{\tilde{a}_2(\lambda-1)}$	0	0
12	$\frac{-\lambda}{\tilde{a}_2(\lambda+1)}$	$\frac{\lambda}{\lambda+1}$	$\frac{2\lambda f(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	1	$\frac{\lambda^2(-\tilde{a}_2+1)}{\tilde{a}_2(\lambda^2-1)}$
13	$\frac{-1}{\lambda+1}$	$\frac{\tilde{a}_2}{\lambda+1}$	$\frac{2f(\lambda)}{(\lambda-1)^2}$	$\frac{\tilde{a}_2}{\lambda}$	$\frac{-\lambda(\tilde{a}_2-1)}{\lambda^2-1}$
14	0	0	0	$\frac{-\tilde{a}_2(\lambda+3)}{4\lambda}$	0
15	0	0	0	$\frac{-\tilde{a}_2(\lambda+3)}{4\lambda}$	0
16	$\frac{-\lambda}{\lambda+1}$	$\frac{-\tilde{a}_2}{\lambda+1}$	$\frac{-h(\lambda)}{(\lambda-1)^2}$	$\frac{\tilde{a}_2}{4}(1 - \frac{1}{\lambda})$	$\frac{\lambda(\tilde{a}_2-1)}{\lambda^2-1}$
17	$\frac{-1}{\lambda+1}$	$\frac{\tilde{a}_2}{\lambda+1}$	$\frac{h(\lambda)}{(\lambda-1)^2}$	$\frac{\tilde{a}_2}{4}(-1 + \frac{1}{\lambda})$	$\frac{-\lambda(\tilde{a}_2-1)}{\lambda^2-1}$
18	0	0	0	$\frac{-\tilde{a}_2(3\lambda+1)}{4\lambda}$	0

Table 3.1: Coefficients of linear inequality constraints.



$i$	$<_i$	$b_i$	$c_i$	$d_i$	$e_i$
1	$>$	$-\frac{\lambda}{\tilde{a}_2}$	$\frac{2\lambda(\lambda+1)g(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$\lambda + 1$	$\frac{\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda-1)}$
2	$>$	$\frac{1}{\tilde{a}_2}$	$\frac{2(\lambda+1)f(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$\frac{\lambda+1}{\lambda}$	$\frac{\lambda(-\tilde{a}_2+1)}{\tilde{a}_2(\lambda-1)}$
3	$<$	$-\frac{\lambda}{\tilde{a}_2}$	$\frac{-2\lambda(\lambda+1)g(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$-(\lambda + 1)$	$\frac{-\tilde{a}_2+\lambda^2}{\tilde{a}_2(\lambda-1)}$
4	$<$	$\frac{1}{\tilde{a}_2}$	$\frac{-2(\lambda+1)f(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$-\frac{\lambda+1}{\lambda}$	$\frac{\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda-1)}$
5	$>$	$\frac{-\lambda}{\tilde{a}_2}$	$\frac{-(\lambda+1)h(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$\frac{\lambda^2-1}{4\lambda}$	$\frac{\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda-1)}$
6	$<$	$\frac{1}{\tilde{a}_2}$	$\frac{-(\lambda+1)h(\lambda)}{\tilde{a}_2(\lambda-1)^2}$	$\frac{\lambda^2-1}{4\lambda}$	$\frac{\lambda(\tilde{a}_2-1)}{\tilde{a}_2(\lambda-1)}$

Table 3.2: Coefficients of 6 linear inequality constraints,  $f$ ,  $g$ , and  $h$  are given by equations (3.38), (3.39), and (3.40), respectively.

as in [27]. The remaining inequalities can be re-written in the  $n - m$  plane. Once this is done, it is immediately apparent that many of these conditions are actually equivalent to one another, specifically, 2 is equivalent to 3, 8 is equivalent to 9, and 12 is equivalent to 13. Thus, there are actually only six independent inequalities and we write them as

$$n <_i b_i m + c_i q + d_i p + e_i, \quad i = 1, 2, \dots, 6 \quad (3.41)$$

where the coefficients are given in Table 3.2 and  $\lambda = \frac{\tilde{a}_2}{\tilde{a}_1}$ .

We will denote the boundary line of inequality  $i$  as the line  $L_i$ . Specifically, this is the set of points  $L_i = \{(m, n) \in [0, 1] \times [0, 1] : n = b_i m + c_i q + d_i p + e_i\}$ . For notational purposes, we will define the following notation  $l_i = b_i m + c_i q + d_i p + e_i$ .

Inequalities (3.41) govern the plateau region for a very specific type of limit cycle, i.e., one that does not intersect any of the rectangular roof regions at the checkerboard corner points. If the checkerboard does intersect any of these regions, one or more of the above conditions change and the inequalities and equation for  $w_1$  are different. In fact, we can show that if the  $C_{z_0}^{p,q}$  conditions are violated, then a number of these inequalities become non-linear equations and as such solution of them is a more difficult problem and will not be discussed in this thesis. We hypothesize that there is a wider plateau zone outside of the region bounded by 1-6.

In Figure 3.11 we see an example plateau region. The shaded region is the region that obeys each of the six inequalities (3.41). From this figure it appears that in actuality, only four of the equations play a role in determining this region. This is easily verified by observing that  $l_1$ ,  $l_3$ , and  $l_5$  have the same slope in the  $m - n$  plane and lines  $l_2$ ,  $l_4$ , and  $l_6$  have the same slope in the  $m - n$  plane. This means that these lines will always be parallel with one another, so if a plateau region exists, than it will be a parallelogram with boundaries given by these lines.

If we further examine the relationship between these lines we see that only lines 1, 2, 3, and 4 are needed to specify the actual plateau zone. This is true because we can prove that for all values of  $\lambda > 0$  that  $l_5 \leq l_1$  and  $l_4 \leq l_6$ .

To show that  $l_5 \leq l_1$ , we note that

$$\begin{aligned} b_5 &= b_1, \\ d_5 \leq d_1 &\iff \frac{(\lambda + 1)(\lambda - 1)}{4\lambda} \leq \lambda + 1 \iff \lambda - 1 \leq 4\lambda \iff -1 \leq 3\lambda, \\ e_5 &= e_1, \\ c_5 \leq c_1 &\iff -h(\lambda) \leq 2\lambda g(\lambda) \iff \lambda^2 - 2\lambda \ln(\lambda) - 1 \leq 2\lambda^2 - 2\lambda \ln(\lambda) - 2\lambda \\ &\iff 0 \leq \lambda^2 - 2\lambda + 1 = (\lambda - 1)^2, \end{aligned}$$

and to show that  $l_4 \leq l_6$ , we note that

$$\begin{aligned} b_4 &= b_6, \\ e_4 &= e_6, \\ d_4 \leq d_6 &\iff -\frac{\lambda + 1}{\lambda} \leq \frac{(\lambda + 1)(\lambda - 1)}{4\lambda} \iff -4 \leq \lambda - 1 \iff -3 \leq \lambda, \\ c_4 \leq c_6 &\iff -2f(\lambda) \leq -h(\lambda) \iff -2(\lambda \ln(\lambda) - \lambda + 1) \leq (\lambda^2 - 2\lambda \ln(\lambda) - 1), \\ &\iff 0 \leq \lambda^2 - 2\lambda + 1 = (\lambda - 1)^2. \end{aligned}$$

To determine the existence of a plateau region for a particular case, we must look to the relationship between each of the above parallel lines. This relationship is determined primarily by relationship of the value of phase velocities. To determine this relationship, we must know more about the behavior of  $f$  and  $g$ . From elementary arguments, we have that

$$\begin{aligned} f' &= \ln(\lambda) + 1 - 1 = \ln(\lambda), & f'' &= \frac{1}{\lambda}, \\ g' &= 1 - \frac{1}{\lambda}, & g'' &= \frac{1}{\lambda^2}, \\ h' &= -2\lambda + 2\ln(\lambda) + 2, & h'' &= -2 + \frac{1}{\lambda} = 2\left(\frac{1}{\lambda} - 1\right). \end{aligned}$$

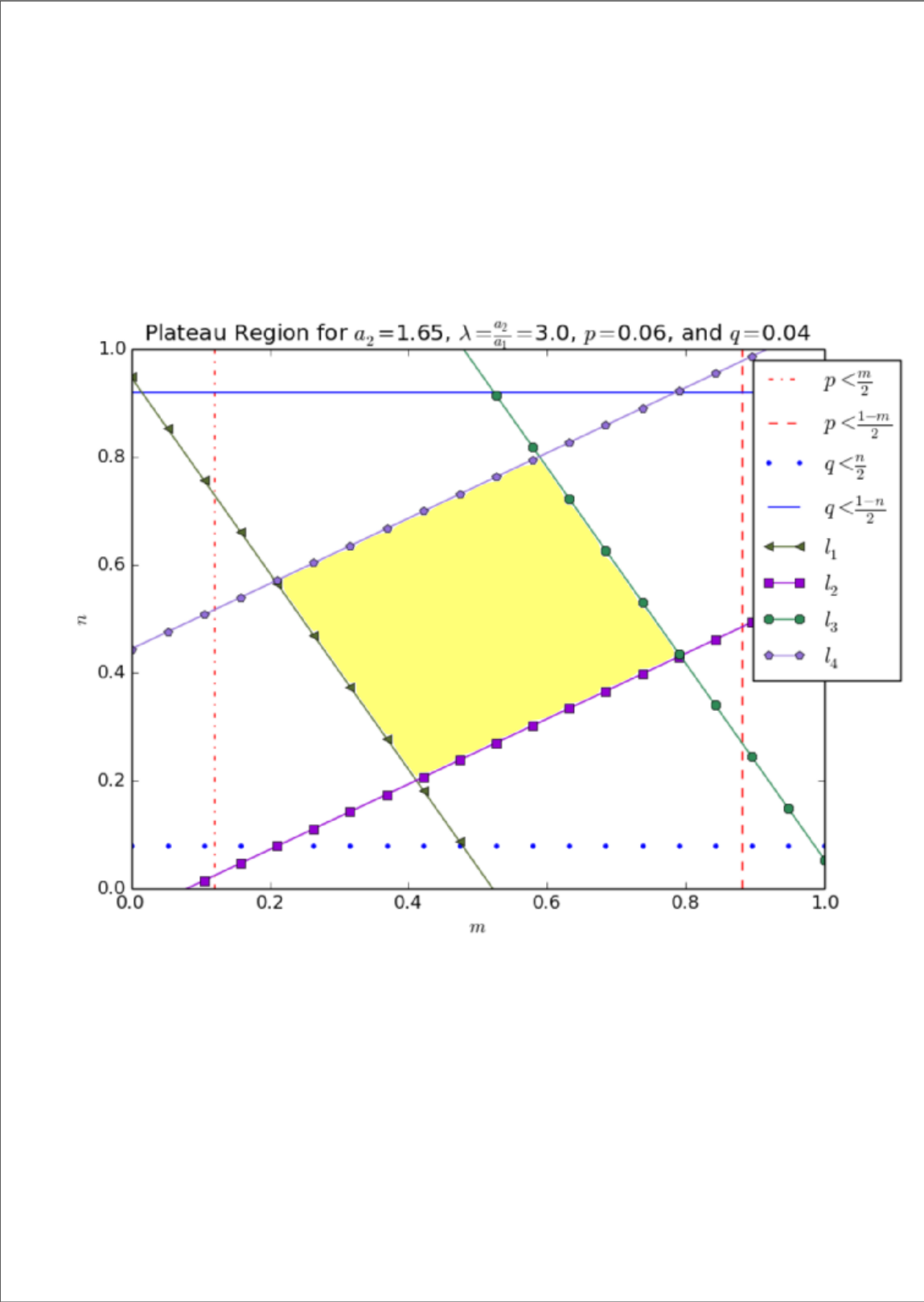


Figure 3.11: Plateau for certain values of material parameters.

$i$	$\frac{\partial l_i}{\partial p}$	$\frac{\partial l_i}{\partial q}$	
1	$c_1 > 0$	$d_1 > 0$	$l_1$ moves up
2	$c_2 > 0$	$d_2 > 0$	$l_2$ moves up
3	$c_3 < 0$	$d_3 < 0$	$l_3$ moves down
4	$c_4 < 0$	$d_4 < 0$	$l_4$ moves down

Table 3.3: Partial derivative of plateau boundary line equation with respect to smoothing parameters  $p$  and  $q$ .

It is clear that  $f, g > 0$  for all  $\lambda > 0$  and that  $h > 0$  for  $0 < \lambda < 1$  and  $h < 0$  for  $1 < \lambda$ . Thus, we have that  $\frac{\partial l_1}{\partial p} = c_1 > 0$   $\frac{\partial l_1}{\partial q} = d_1 > 0$  so line 1 is moving up as we increase either type of smoothing. We can continue this procedure to compile Table 3.3. This table contains all information needed to determine the direction of motion of each line with respect to smoothing parameters  $p$  and  $q$ .

With the motion of the boundary lines in mind, we note that the two conditions for existence of the plateau region are that  $l_1 \leq l_3$  and that  $l_2 \leq l_4$ , in either case this is equivalent to  $0 \leq (b_{i+2} - b_i)m + (c_{i+2} - c_i)p + (d_{i+2} - d_i)q + (e_{i+2} - e_i)$  for  $i = 1, 2$ . These two conditions read as follows

$$0 \leq -\frac{4(\lambda + 1)f(\lambda)}{\tilde{a}_2(\lambda - 1)^2}p - \frac{2(\lambda + 1)}{\lambda}q + \frac{-1 + \tilde{a}_2\lambda - \lambda + \tilde{a}_2}{\tilde{a}_2(\lambda - 1)},$$

$$0 \leq -\frac{4\lambda(\lambda + 1)g(\lambda)}{\tilde{a}_2(\lambda - 1)^2}p - 2(\lambda + 1)q + \frac{\lambda^2 - \tilde{a}_2\lambda + \lambda - \tilde{a}_2}{\tilde{a}_2(\lambda - 1)}.$$

After some algebra, these conditions are equivalent to the following,

$$4\lambda f(\lambda)p + 2\tilde{a}_2(\lambda - 1)^2 q \leq \lambda(\tilde{a}_2 - 1)(\lambda - 1), \quad (3.42)$$

$$4\lambda g(\lambda)p + 2\tilde{a}_2(\lambda - 1)^2 q \leq (\lambda - \tilde{a}_2)(\lambda - 1). \quad (3.43)$$

The left hand side of each inequality is always positive. Thus, by choosing a small enough  $p$  and  $q$ , i.e., as  $p, q \rightarrow 0$  these conditions reduce to

$$0 \leq (\tilde{a}_2 - 1)(\lambda - 1), \quad (3.44)$$

$$0 \leq (\lambda - \tilde{a}_2)(\lambda - 1). \quad (3.45)$$

This means that either

$$1 \leq \tilde{a}_2, \quad \tilde{a}_1 \leq 1, \quad (3.46)$$

or

$$\tilde{a}_2 \leq 1, \quad \leq \tilde{a}_1, \quad (3.47)$$

by substitution of  $\lambda$  into the preceding inequalities (3.42) and (3.43). This condition is the same one derived in paper [27], namely, it guarantees that focusing exists if the greater non-dimensional phase velocity is more than 1 and the smaller non-dimensional phase velocity is less than 1.

After some algebraic manipulations, this becomes the following inequality

$$a_1 \left( \frac{1 + 2(\lambda - 1)q}{1 - \frac{4(\lambda - \ln(\lambda) - 1)}{\lambda - 1}p} \right) \leq \frac{\delta}{\tau} \leq a_2 \left( \frac{1 - \frac{2(\lambda - 1)}{\lambda}q}{1 + \frac{4(\lambda \ln(\lambda) - \lambda + 1)}{\lambda - 1}p} \right). \quad (3.48)$$

This is a direct generalization to the condition derived in reference [27].

Condition (3.48) is guarantees the existence of a  $\mathcal{C}_{z_0}^{p,q}$  plateau zone in the same way that equation (3.46) (or equation (3.47)) does. If it is satisfied than there is a connected set of  $m, n$  values (parallelogram in  $m - n$  space) that produce the focusing effect.

It is possible to obtain the amount of smoothing that will destroy the  $\mathcal{C}_{z_0}^{p,q}$  plateau. This is done as follows we can rewrite (3.42) and (3.43) as

$$q \leq -\frac{4\lambda f(\lambda)}{2\tilde{a}_2(\lambda - 1)^2}p + \frac{\lambda(\tilde{a}_2 - 1)(\lambda - 1)}{2\tilde{a}_2(\lambda - 1)^2}, \quad (3.49)$$

$$q \leq -\frac{4\lambda g(\lambda)}{2\tilde{a}_2(\lambda - 1)^2}p + \frac{(\lambda - \tilde{a}_2)(\lambda - 1)}{2\tilde{a}_2(\lambda - 1)^2}. \quad (3.50)$$

For fixed values of  $a_1$  and  $a_2$  these inequalities represent lines separating two domains in the  $p - q$  plane, a.k.a., the smoothing space. In this space, we are only interested in the upper quadrant which corresponds to positive values of  $p$  and  $q$ . The origin of this space corresponds to a standard sharp checkerboard, i.e., it corresponds to no smoothing. We are interested in determining the exact amount of smoothing that can be added and still allow for a  $\mathcal{C}_{z_0}^{p,q}$  plateau zone. To determine this, we plot inequality (3.49) and equation (3.50) in the  $p - q$  space. There will be a total of 4 cases that are possible. The first case is that the lines do not intersect the first quadrant; this is the trivial case of no plateau zone. The second two cases are that the lines do intersect the first quadrant, however, the intersection point of line (3.49) and line (3.50) is not in the first quadrant. A specific example of this case is given in Figure 3.12 for specific values of material parameters  $\lambda = 2.0$  and  $a_2 = 1.1$ . The last possibility is that the intersection point of line (3.49) and line

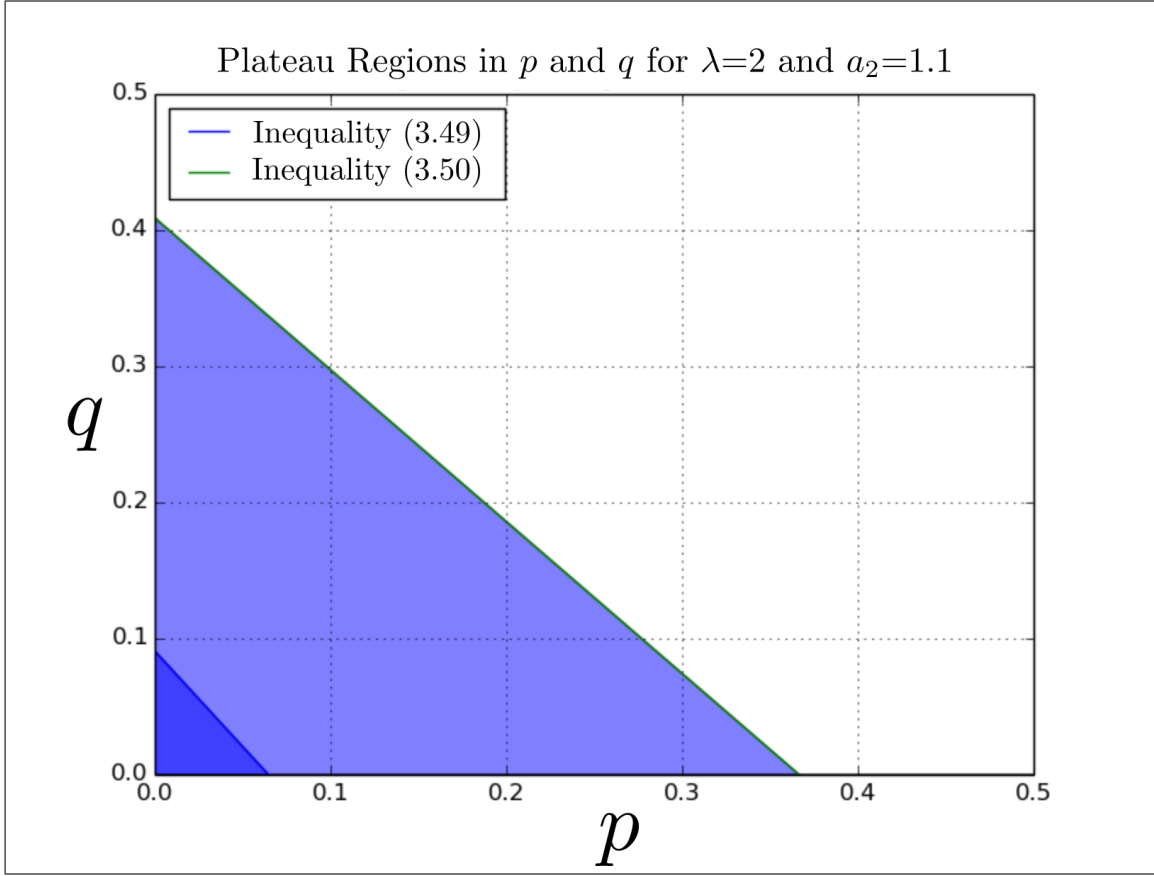


Figure 3.12: Existence region for plateau region in  $p$  and  $q$  space.

(3.50) lies distinctly in the first quadrant. If this point is in the first quadrant, then the boundary of the  $\mathcal{C}^{p,q}$  plateau region is a polygon with 4 sides corresponding to  $p = 0$ ,  $q = 0$ , and the above two inequalities. In this case, the intersection point is the maximum amount of smoothing that can be added and still guarantee a  $\mathcal{C}_{z_0}^{p,q}$  plateau zone.

It is clear that if  $0 < \lambda < 1$  then  $f < g$  and if  $1 < \lambda$  then  $g < f$ . This means that if  $0 < \lambda < 1$  the boundary line of (3.50) is steeper than that of (3.49) and if  $1 < \lambda$  the boundary line of (3.49) is steeper than that of (3.50). To determine the shape of the existence region in the smoothing space, we finally solve for  $p_S$  and  $q_S$ , the point at which the two lines intersect. According to the explanation given above, this point will determine the maximum amount of spatial and temporal smoothing that may be added and still guarantee convergence. This intersection point is given

as follows,

$$p_s = \frac{(\lambda - 1) ((\tilde{a}_2 - 1) \lambda + (\tilde{a}_2 - \lambda))}{4\lambda (\lambda (\ln(\lambda) - 2) + \ln(\lambda) + 2)}, \quad (3.51)$$

$$q_s = \frac{(\tilde{a}_2 (1 - \lambda) + \lambda \ln(\lambda))}{2\tilde{a}_2 (\lambda (\ln(\lambda) - 2) + \ln(\lambda) + 2)}. \quad (3.52)$$

The denominator of each term is nonzero whenever  $\lambda \neq 1$ .

Figure 3.12 shows these inequalities displayed. We can see that they are both satisfied and so the  $\mathcal{C}^{p,q}$  plateau exists. The boundary line goes from around  $(p, q)$  from  $(0, .09)$  to  $(.06, 0)$ . We can explicitly solve for these points. Specifically,  $q = 0$  only in the dark region around  $p = 0, q = 0$ . This is expected from the theory surrounding  $\mathcal{C}_{z_0}^{p,q}$  limit cycles.

We would like to use this information to determine how much smoothing we can add and still reasonably expect  $\mathcal{C}p = 0$  corresponds to  $q = \frac{\lambda(\tilde{a}_2-1)(\lambda-1)}{2*\tilde{a}_2(\lambda-1)^2} = \frac{2}{22} \approx .09$  and  $q = 0$  corresponds to  $p = \frac{\lambda(\tilde{a}_2-1)(\lambda-1)}{4\lambda f(\lambda)} = \frac{2}{80(2\ln(2)-2+1)} \approx .06$ . This means that the region of existence for the  $\mathcal{C}^{p,q}$  plateau zone is a triangle in  $p - q$  space, the boundary of which is a line from  $(0, .06)$  to  $(.09, 0)$ .

Practically, this means that if there is a spatial smoothing thickness from 0% to 6% of the period length, we can then allow, respectively, for temporal smoothing from 9% to 0% and still expect limit cycles of  $\mathcal{C}^{p,q}$  plateau zone to exist. It is important to emphasize that past this amount of smoothing, there may still be limit cycles, however, these do not necessarily follow the specific characteristic path studied in chapter 3 and we must complete more theoretical work must be completed to better understand any plateau zones associated with these new limit cycles.

In Figure 3.13 we see an example where  $\lambda = 0.4$  and  $\tilde{a}_2 = 0.6$  and the smoothing parameters are increased from  $p = 0.015$  and  $q = 0.025$  to  $p = 0.04$  and  $q = 0.06$ . This increase results in shrinking the plateau zone so that the indicated point is not included in the region. To fully confirm this, we examine Figure 3.14. In this figure, we see a comparison of the limit cycle behavior. As we increase the smoothing, we can see that the limit cycle changes from  $\mathcal{C}^{p,q}$  to violating these conditions.

### 3.3 Conclusion

In chapter 3 we have given a specific extension of the perfect “sharp” checkerboard discussed in paper [27] to a new FG checkerboard. Specifically, in section 3.1.1 we show numerically that characteristic focusing persists for the general FG material,

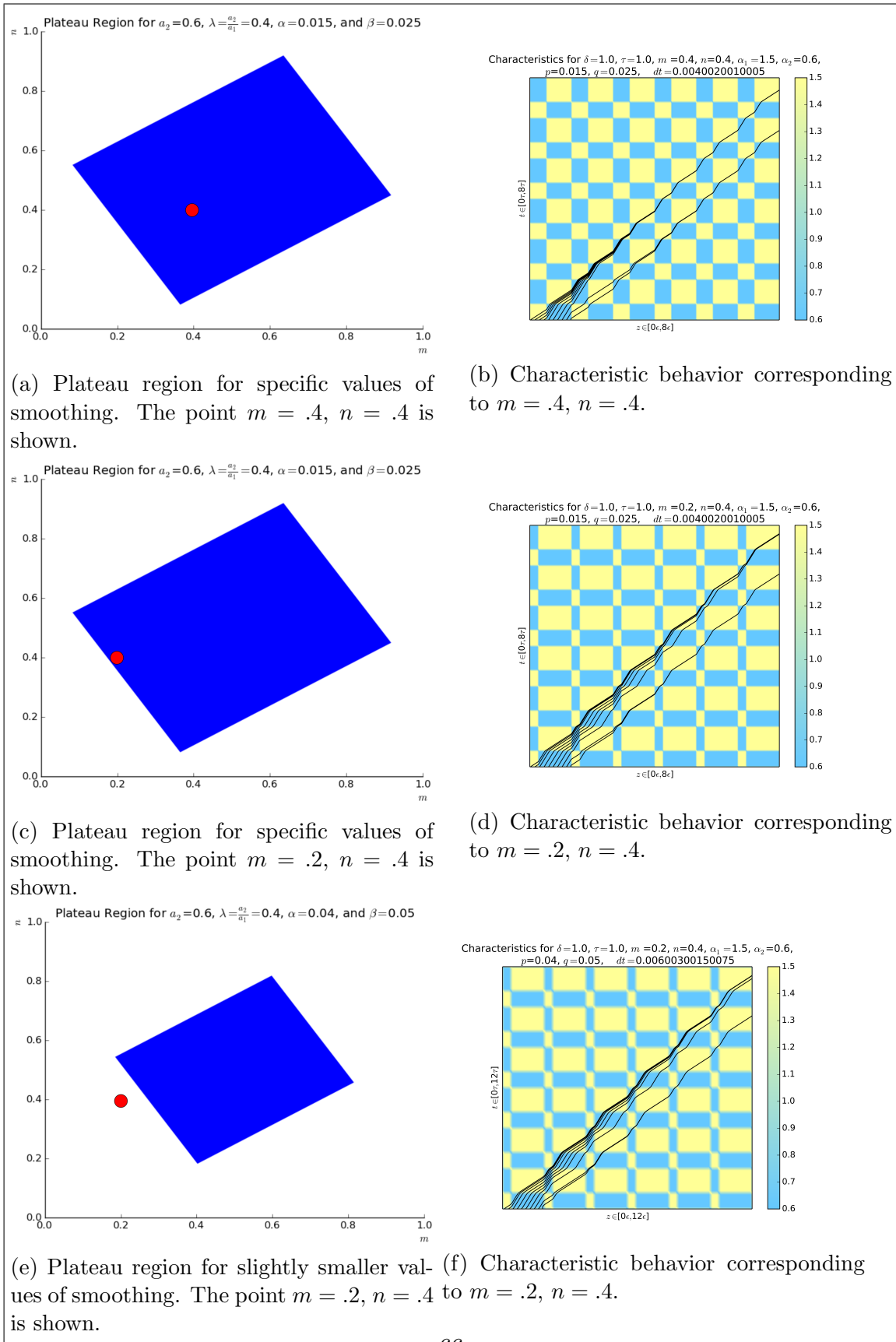


Figure 3.13: Characteristic behavior after smoothing.



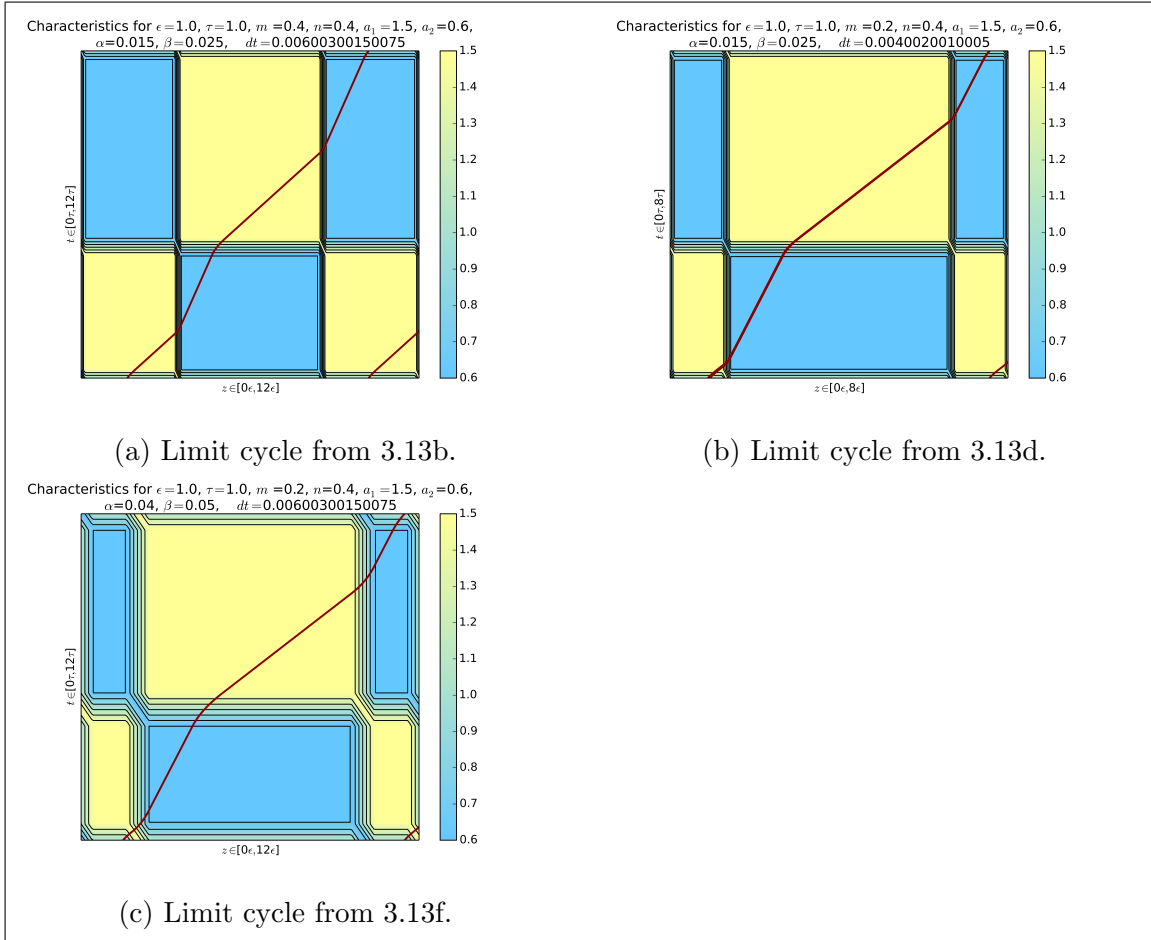


Figure 3.14: Characteristic behavior after 10 periods for each case. Limit cycle behavior is observed, however, only the first two figures show proper  $\mathcal{C}_{z_0}^{p,q}$  behavior.

i.e., when  $a(z, t)$  is allowed to be non-linear. In section 3.2 we furthered this idea by developing existence conditions similar to those governing “sharp”  $\mathcal{C}_{z_0}$  limit cycles and introduce the class of FG  $\mathcal{C}_{z_0}^{p,q}$  limit cycles .

We are also interested in the propagation of waves through a checkerboard that has non-matching wave impedances. This issue presents an interesting theoretical problem that has not been resolved. In chapter 4, we will approach the problem of correctly measuring the energy evolution in a traditional checkerboard, a FG checkerboard, and a checkerboard with mismatched wave impedances.

# Chapter 4

## Energy Accumulation in Checkerboard Generalizations

One of the main important issues in the study of DM is evaluation of the energy of a wave after as it propagates through a dynamic material. Chapter 3 gave an analytical extension to the understanding of the existence region for characteristic convergence in a FG checkerboard. Specifically, it extended the  $\mathcal{C}_{z_0}$  plateau zone to the  $\mathcal{C}_{z_0}^{p,q}$ . In general, it is difficult to analytically solve for wave propagation through these material geometries unless very specific restrictions are placed on the material parameters. Chapter 4 will apply the numerical approach discussed in Section 1.2 to investigate energy accumulation in FG materials and other, more complicated, material geometries that may arise in practical situations.

Recall that the wave  $u(z, t)$  is governed by the following variable coefficient wave equation,

$$\begin{aligned}(\rho u_t)_t - (k u_z)_z &= 0, \quad (z, t) \in [a, b] \times [0, T], \\ u(z, 0) &= u_0(z), \quad u_t(z, 0) = v_0(z),\end{aligned}$$

where  $\rho(z, t)$  and  $k(z, t)$  are material parameters that are specific control functions. Specifically, they are related to the wave velocity  $a(z, t) = \sqrt{k(z, t)/\rho(z, t)}$ , which gives the local speed of a traveling wave and the wave impedance  $\gamma(z, t) = \sqrt{\rho(z, t)k(z, t)}$ , which is a measure of the how much the medium locally resists propagation of the traveling wave, i.e.,  $\rho = \gamma/a$  and  $k = \gamma a$ .

One approach to numerically solve this equation is to introduce a potential function  $\xi$  and convert the preceding second order wave equation to a hyperbolic system of first order equations. The potential function  $\xi$  satisfies the following first order equations

$$u_t - (1/\rho)\xi_z = 0 \tag{4.1}$$

$$\xi_t - k u_z = 0 \tag{4.2}$$

In this section we will approximate the total energy over  $z \in [a, b]$  at time  $T$  of a wave traveling through a checkerboard. The analytical expression for this energy is given by expression

$$E = \int_a^b (\rho(z, T)u_t^2 + k(z, T)u_z^2) dz.$$

When the checkerboard material geometry admits limit cycles, the characteristic convergence implies that the solution evolves to include regions of steep gradients, this is clearly seen in Figure 1.6 and Figure 1.7. This implies that the above expression for energy will grow rapidly because it depends on the temporal and spatial derivative of the wave solution  $u$ .

It is because of this property that we will use adaptive mesh refinement (AMR). This means the computational mesh will be refined in regions where the solution gradient grows past a certain tolerance. This has shown to produce very good results and it will allow us to accurately compute the wave energy. However, the energy's dependence on the temporal derivative is an undesirable property because for a given location on the  $z$ -axis, the cell density will be different between different time steps which creates a potential problem for temporal differencing. To remedy this, we will rewrite the temporal derivatives in terms of the potential formulation. The expression for  $u_t$  in terms of only spatial derivatives only is  $\frac{\xi_z}{\rho}$ , and so, the expression for energy becomes

$$E = \int_a^b \left( \frac{\xi_z^2}{\rho(z, T)} + k(z, T)u_z^2 \right) dz. \quad (4.3)$$

This can be rewritten in terms of wave velocity  $a$  and elastic wave impedance  $\gamma$  as follows,

$$E = \int_a^b a(z, T) \left( \frac{1}{\gamma(z, T)} \xi_z^2 + \gamma(z, T)u_z^2 \right) dz. \quad (4.4)$$

In our numerical scheme, the energy is evaluated by approximating this definite integral. In [27], it is shown that for convergent characteristics in class  $\mathcal{C}_{z_0}$  the energy after every period of the checkerboard should grow by a factor  $(a_2/a_1)^2$ .

## 4.1 Numerical Procedure

In this section we describe the numerical methods implemented in our investigation.

### 4.1.1 Basic Numerical Method

To solve the system of equations (4.1) and (4.2) we use Godunov’s upwind method with limiting on an adaptive mesh. This is a second-order high resolution finite volume method. For the simulations done so far, we have used the minmod limiter. For all of the simulations done in this thesis, we have utilized the open-source package of Python and Fortran routines known as Clawpack [9].

Specifically, we use Clawpack to solve the following system

$$p_t + k_C w_z = 0, \tag{4.5}$$

$$w_t + (1/\rho_C)p_z = 0, \tag{4.6}$$

which is the 1D acoustics equations with bulk modulus  $k_C$  and density  $\rho_C$ . Application to system (4.1) and (4.2) from our DM problem requires specific substitutions to the acoustics equations (4.5) and (4.6) so that they match. Specifically, the substitutions

$$\begin{aligned} k_C = 1/\rho = a/\gamma, & \quad \rho_C = 1/k = 1/(\gamma a), \\ p = u, & \quad w = -\xi, \end{aligned}$$

will successfully recover equations (4.1) and (4.2). Note that the wavespeed is identical for both systems, i.e.,  $a_C = \sqrt{\frac{k_C}{\rho_C}} = \sqrt{\frac{1/\rho}{1/k}} = \sqrt{\frac{k}{\rho}} = a$ . However, the wave impedance for equations (4.5) and (4.6) is the reciprocal of the wave impedance from system (4.1) and (4.2), i.e.,  $\gamma_C = \rho_C a_C = \frac{a}{k} = \frac{1}{\gamma}$ . This is important to consider when looking at propagation through media for which there is a mismatch in wave impedance.

At the beginning of the numerical simulation, the user chooses the type of adaptive mesh refinement (AMR) refinement criteria. The main parameters that must be set are how many levels of mesh refinement will be used and the spatial refinement ratio between differing AMR levels. In what follows, we will use  $L$  to denote the grid level, with  $L = 1$  corresponding to the coarsest grid and  $L = \text{Max.Grid.Levels}$  corresponding to the finest grid. Figure 4.1 shows a plot of three AMR grid levels for a test simulation. As expected, the full computational domain ( $z \in [0, 14]$ ) is covered by the union of all the level 1 patches, however, the level 2 and 3 patches only cover the region where the solution gradient is over a certain threshold.

The most basic method of refinement uses a routine called `flag2refine`. This compares the difference in solution between neighboring cells. If this difference is larger than a specified tolerance, called `flag2refine_tol`, the cell is flagged for refinement, and is refined by a specified amount for the next time step. Another method that can be used is called `flag_richardson`. This uses Richardson extrapolation and approximates the error in solution on two of the finer grids’

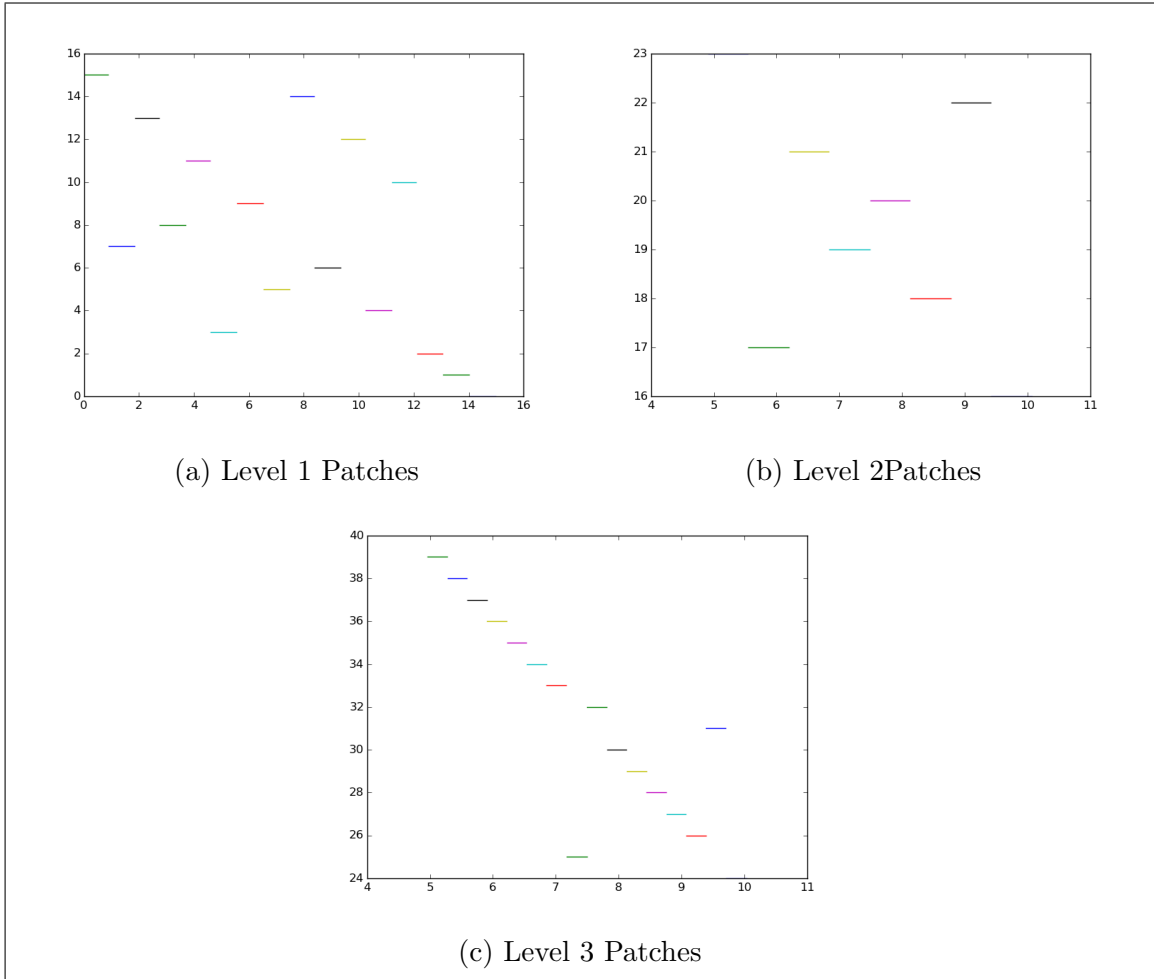


Figure 4.1: Comparison of 3 levels of AMR patches at time .5

time-steps to one timestep on the coarser grid. If this value exceeds the tolerance `flag_richardson_tol`, the cell is flagged for refinement. Regardless, during the course of a simulation, the domain (on each AMR refinement level) must be cut into patches. The union of all of the patches on level 1 give the entire domain. The union of all patches on any higher level will only describe regions where the solution gradient is higher than the previously set tolerance. We have run the simulations in this thesis using the `flag2refine` method with tolerance `flag2refine_tol` =  $10^{-4}$ .

A DM simulation is run using the following procedure:

- Modify the following Fortran files:
  - `qinit.f` – Sets the initial conditions. We will make use of `bump.f`, which is a self made Fortran function that gives a bump polynomial on a given interval. In `qinit.f`,  $q(1, i, j) \rightarrow u_0(x_{ij})$ ,  $q(2, i, j) \rightarrow -\xi_0(x_{ij})$

- `setaux.f` – Sets initial auxiliary conditions ( $\text{aux}_1 = \rho_C = 1/(\gamma a)$ ,  $\text{aux}_2 = a$ ). For the linear implementation, we must choose to make  $a$  a linear function of space and time and  $\gamma$  constant (unless investigating non-matching wave impedances). It is important to remember that because of this,  $\rho$  and  $k$  are not necessarily linear when we choose  $a$  to be.
  - `b4step2.f` – Used to set time dependent material parameters or time dependent boundary conditions.
  - `Functions.f` – Self-made Fortran code used to create sharp and/or functionally graded checkerboard function.
- Run `make` in the working directory. This will create an executable called `xamr` (or possibly something else if the `makefile` is changed).
  - Modify the following python files:
    - `setrun.py` – Use this to set AMR parameters, material parameters, checkerboard parameters, number/type of output times, initial time, final time, endpoints. In this file, parameters are entered for  $\gamma_1, \gamma_2, a_1, a_2$  and are converted to their correct meaning in terms of Clawpack parameters  $\rho_C$  and  $k_C$
    - `setplot.py` - Edit to setup plots using visclaw.
  - Either run “`make .output`” or run the executable `xamr` directly. The latter is what we do for our simulations. This will be important when describing how to parallelize a brute force solution to the optimization problem discussed earlier.

When running a simulation, a directory called `_output` is created. This directory contains a number of files, each of which contains the specific problem information and the solution information at every time step.

These files are the following,

- `fort.q####` Patchwise solution data for  $u$  and  $\xi$  at every timestep
- `fort.a####` Patchwise material property data for  $\gamma$  and  $a$  at every timestep
- `fort.t####` Temporal discretization data

where the `#` sign means that multiple files are produced, each with its own unique identifier number.

After a simulation we use custom Python and Fortran functions to read the `fort.q`, `fort.a`, and `fort.t` files to pull in data, compute finite differences, and calculate the final energy. This is done by approximating the integral in equation (4.3) patchwise on each AMR level. We do not apply an advanced method for integral

approximation, however, this can be easily implemented in the future. Specifically, assume that we wish to approximate equation (4.3) on a given AMR level. There will be a fixed number of patches on the interval from  $a$  to  $b$  and there will be a fixed number of computational cells per patch. A simple quadrature schemes is obtained by using first-order differences to approximate the spatial derivatives  $\xi_z$  and  $u_z$  and then approximating (4.3) with the following sum,

$$\begin{aligned} E &\approx \sum_{k=1}^p \sum_{i=1}^{M_k} \left[ \frac{1}{\rho_i} \left( \frac{\Delta_i \xi}{\Delta_i z} \right)^2 + k_i \left( \frac{\Delta_i u}{\Delta_i z} \right)^2 \right] \Delta_i z, \\ &= \sum_{k=1}^p \sum_{i=1}^{M_k} \left[ \frac{1}{\rho_i} \frac{(\Delta_i \xi)^2}{\Delta_i z} + k_i \frac{(\Delta_i u)^2}{\Delta_i z} \right], \end{aligned} \quad (4.7)$$

where  $p$  is the total number of patches and  $M_k$  is the number of cells on the  $k$ -th patch and  $\Delta_i(\cdot)$  denotes a first order differencing of the quantity to the right of the operator. This will be explained further below.

Alongside many of the energy plots we will plot the energy curve shifted forward to be centered around some positive value of time. The exact formula for this curve is given by the following relationship,

$$E(t) = E(t_i) \left( \left( \frac{a_2}{a_1} \right)^2 \right)^{(t-t_i)}, \quad (4.8)$$

where  $t_i$  is the amount of time we choose to shift forward. The reason for shifting the center the curve around  $t_i$  is that it takes time, sometimes several periods, for the focusing effect to get initiated. Exponential energy growth will not be observed until characteristic focusing takes shape, therefore, it makes sense to translate the expected theoretical energy curve forward to the time at which accumulation has already begun.

Immediately after the `fort.q` (solution) and `fort.a` (material parameter) data is read into Python it is converted from the Clawpack parameters of  $\rho_C$  and  $k_C$  back to parameters  $\rho$  and  $k$ . Specifically,  $q(1, i, j)$  is read in and then assigned to  $q_1$ , this gives  $u$ .  $q(2, i, j)$  is read in and then it's negative is assigned to  $q_2$ , giving  $\xi$ . Lastly,  $\rho_C$  and  $a_C$  are read in. Recall that  $a_C$  is identical to  $a$ , therefore, nothing must be done to convert back to standard parameters. However, remember that take the reciprocal of  $\rho_C$  to obtain the  $k$  used in our problem.

Figure 4.2 is an image of the energy evolution in a checkerboard using an initial grid of 1000 cells that was computed using equation (4.7). The initial grid is subsequently refined a maximum of six times with a refinement ratio of two between distinct AMR levels. This image plots the energy evolution at every AMR level to



show the distinct improvements as we go to finer and finer levels of mesh refinement.

The image clearly shows that the energy evolution at the two highest refinement levels ( $L=5$  and  $L=6$ ) matches the theoretically expected energy growth for at least the first 3 periods. To show this, we plot the theoretically predicted curve given by equation (4.8) alongside the results. Figure 4.3 shows a similar result, the mean difference is that it was run with an initial grid of 3000 cells and then subsequently refined to a maximum of five different AMR levels. For each of these simulations, the refinement ratio between each time was two. For each simulation we plot the energy evolution for each level of the AMR (the  $L$  value in the legend) and we can clearly see that the curve is very accurate at higher AMR levels.

These two simulations show that this method is successfully capturing the desired behavior with energy growth over the first three periods. In both cases, after this time, we quickly lose accuracy in our calculation of energy. This is because the focusing effect sharpens the wave profile so much that it is smaller than the grid resolution. If we investigate different possible AMR schemes, we will show that it is possible to achieve high accuracy up to 4 periods.

### 4.1.2 Brute-Force Optimization

In the following sections, we are interested in studying how wave energy depends on checkerboard structural and material parameters. Specifically, we are interested in solving for parameters  $m$  and  $n$  that produce the maximal energy accumulation. To do this, we consider the use of the numerical method described in Section 4.1.1 for many different values of  $m$  and  $n$ . This poses a potential problem because the number of simulations we have to do grows very fast. Assume we are interested in the wave energy for  $M$  values of  $m$  and  $N$  values of  $n$ , the computation time required for a fine grid is very large. The exact computation time is somewhere between the  $MNT_{min}$  and  $MNT_{max}$ , where  $T_{min}$  and  $T_{max}$  are, respectively, the smallest and largest time required for a simulation. The most straightforward solution is to run these simulations in parallel.

We wrote several routines that sample over a grid of points in  $m - n$  space, run the FVM simulations in parallel, and compute the energy for each value of  $m, n$ . The results presented in the following were obtained by running the algorithms described above on the Turing Cluster, a high-performance computing system acquired through NSF MRI grant DMS-1337943 to Worcester Polytechnic Institute (WPI).

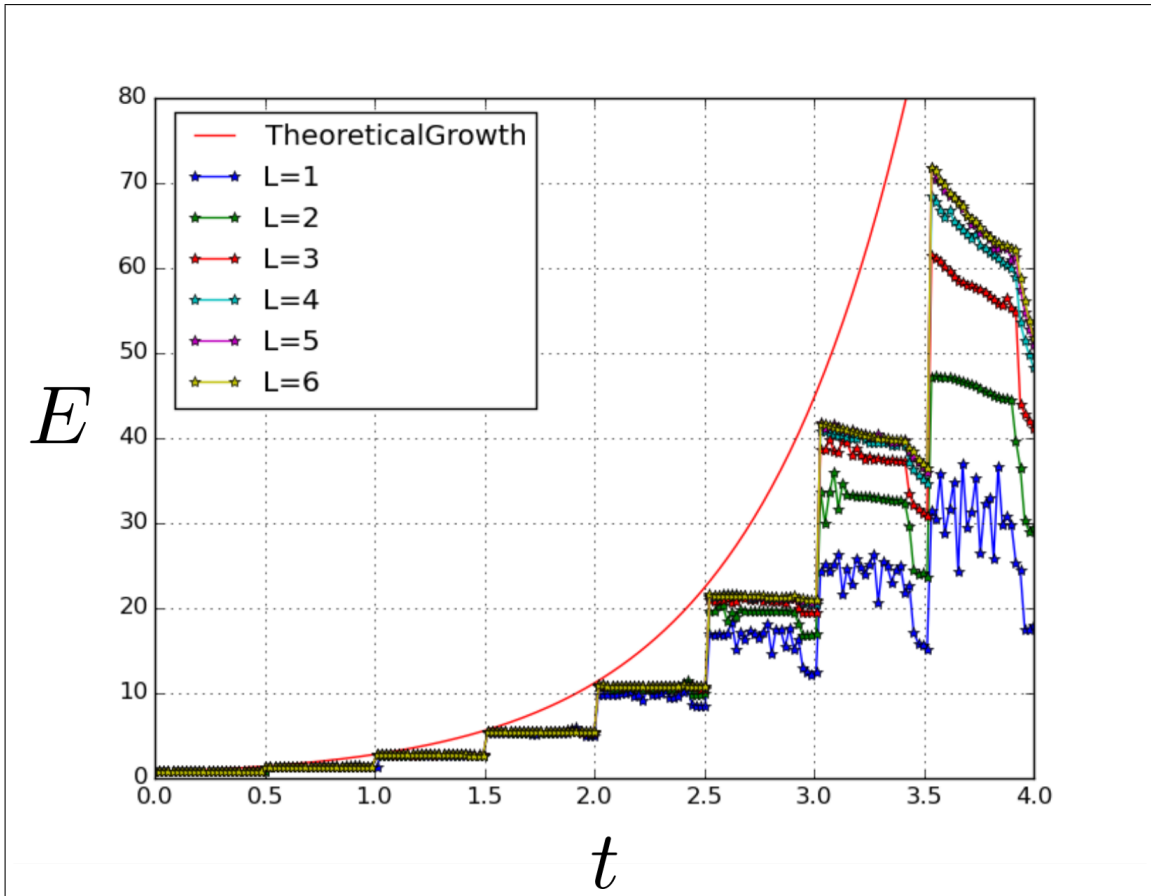


Figure 4.2:  $E(t)$  in a sharp checkerboard for  $a_1 = 0.6$  and  $a_2 = 1.1$  using 6 AMR levels and 1000 initial grid cells .

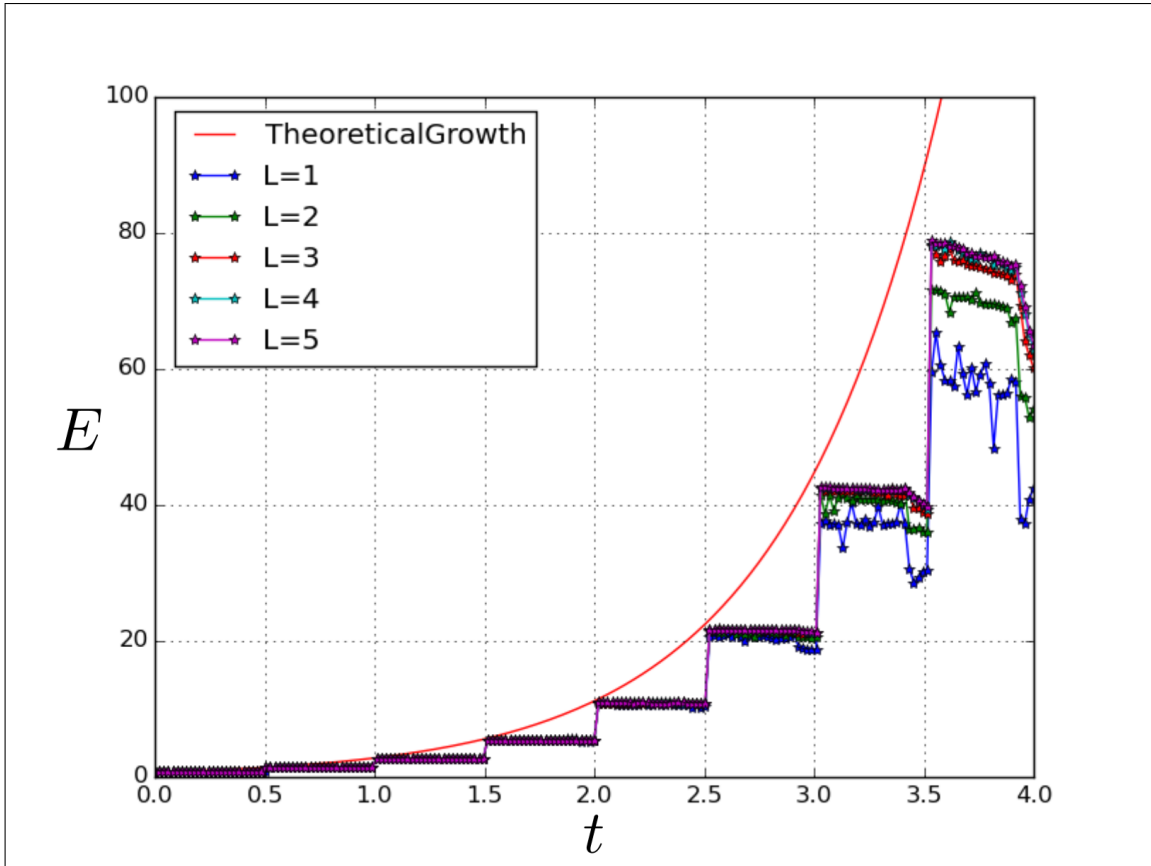


Figure 4.3: Energy evolution in a sharp checkerboard for  $a_1 = .6$  and  $a_2 = 1.1$  using 5 AMR levels and 3000 initial grid cells.

## 4.2 Energy Accumulation in a Checkerboard

In this section, we present a numerical investigation of energy accumulation in various checkerboard structures. First, we examine energy growth in a sharp checkerboard. The plots produced using our code reconfirms previously discovered results from [27] and also confirms similar results for two generalizations of the checkerboard structure: a linear FG checkerboard structure and a sharp checkerboard with mis-matched wave impedances.

### 4.2.1 Energy in a Sharp Checkerboard

Using the method detailed in section 4.1.2 it we calculate the energy as a function of  $m$  and  $n$ . As predicted by the theory from [27], we clearly observe a plateau zone that matches the expected parallelogram shape. In Figure 4.4 we see a plot of the energy ratio between two periods,

$$\frac{E(i\tau)}{E((i-1)\tau)}, \quad (4.9)$$

for  $i$  from 1 to 4.

According to the theory of  $\mathcal{C}_{z_0}$  limit cycles (see reference [27]), when there is characteristic convergence, this ratio should be  $\left(\frac{a_2}{a_1}\right)^2$ . For this image, parameters are such that  $\left(\frac{a_2}{a_1}\right)^2 = 4$ . As time advances, it is clear that this ratio is achieved in the middle of the plateau region. The boundaries appear to be slower at reaching this theoretical value, however, there is a definite tendency to increase to this value within the theoretical plateau region.

It is for this reason that we will investigate more closely the pointwise evolution of energy in the checkerboard. In Figure 4.5, we pick 4 points that we will use as structural parameters to compare the energy evolution: the first point is in the very middle of the checkerboard, the second point is at the northeast boundary line, the third point is at the southeast boundary line (both in and outside the plateau region), these points are respectively given as  $(m, n) \in \{(0.5, 0.5), (0.925, 0.9), (0.5, 0.36), (0.5, 0.37)\}$ . In Figure 4.6, we see different energy plots corresponding to the different values of  $m$  and  $n$ . In each of these cases, there is energy accumulation at each of the distinct jumps. At each jump  $i\tau$  or  $(i+n)\tau$ , the net energy increase is  $\frac{a_2}{a_1}$ . It is clear that the energy growth is slower when we pick a point closer to the boundary when compared to the middle of the checkerboard. This difference is most extreme at the boundary. It is important to consider that even off of the checkerboard, we may still have energy accumulation.

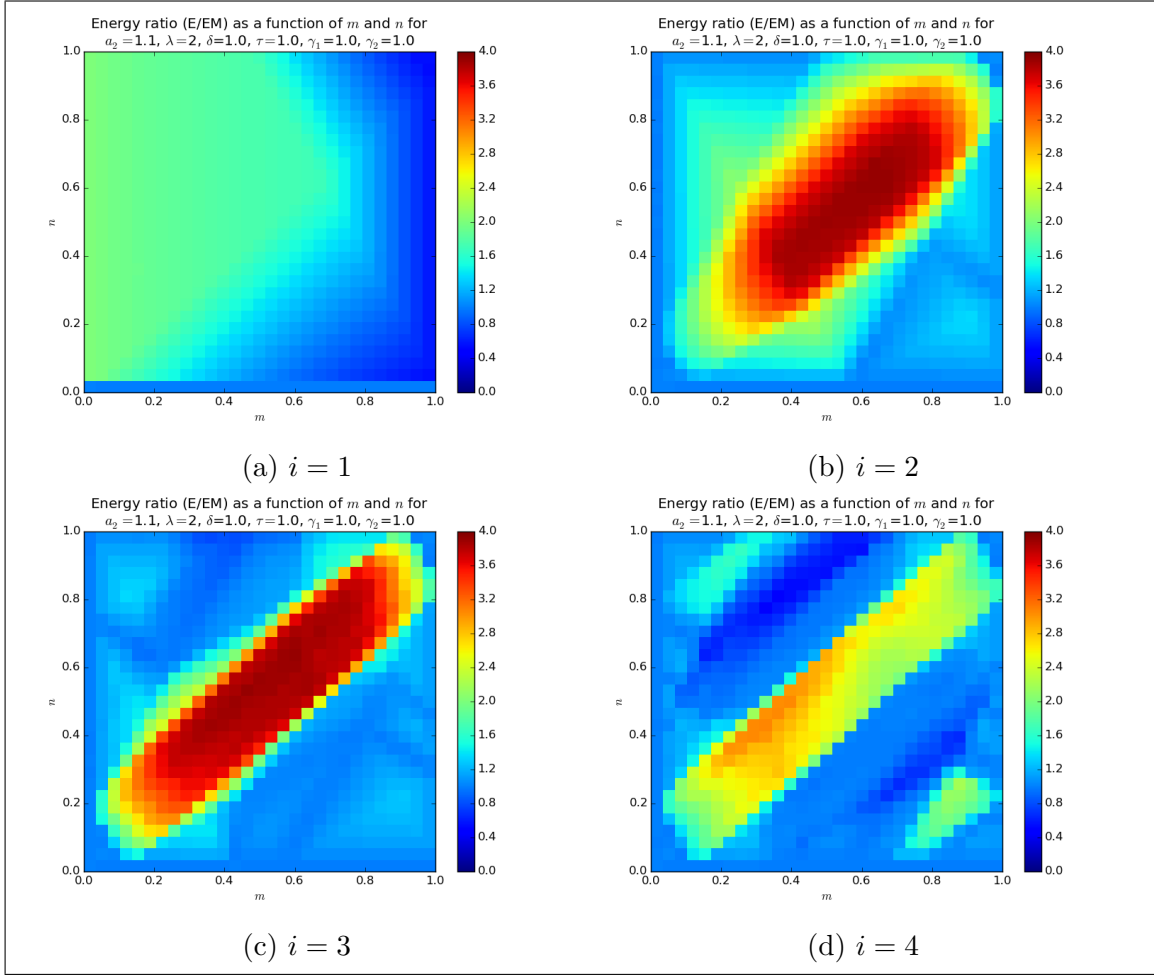


Figure 4.4: Energy ratio between periods as a function of  $m$  and  $n$ .

It is important to contrast this with what happens far off of the plateau. This is summarized in Figure 4.7. In this figure, we see a comparison of 3 cases. The first curve shows the energy evolution in a checkerboard with  $m = \frac{1}{4}$  and  $n = \frac{3}{4}$ . It is clear that there is no energy accumulation. The second curve shows the energy evolution in a static laminate  $m = \frac{1}{2}$  and  $n = 0$ , this is understandably constant. The last curve shows energy evolution in a temporal laminate  $m = 0$  and  $n = \frac{1}{2}$ . This energy is constant in each material section and switches back and forth with each temporal switch.

## 4.2.2 Energy Accumulation in a FG Checkerboard

In this section, we investigate energy accumulation in a linear functionally graded checkerboard. To do this, we consider the energy evolution for various different values of the smoothing parameters  $p$  and  $q$ . In Figure 4.8 we plot the evolution of

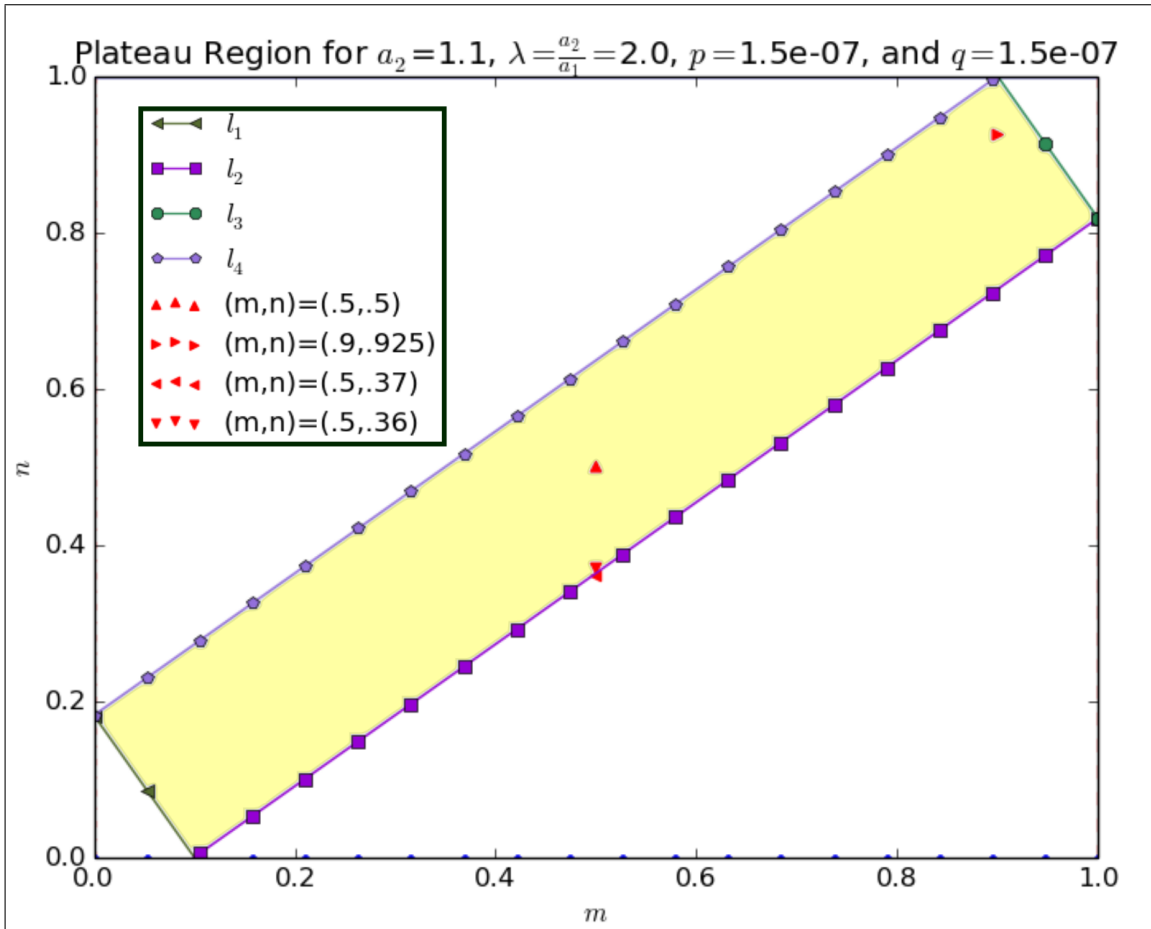


Figure 4.5:  $m$ - $n$  Plateau zone for parameters  $a_1=.55$  and  $a_2=1.1$

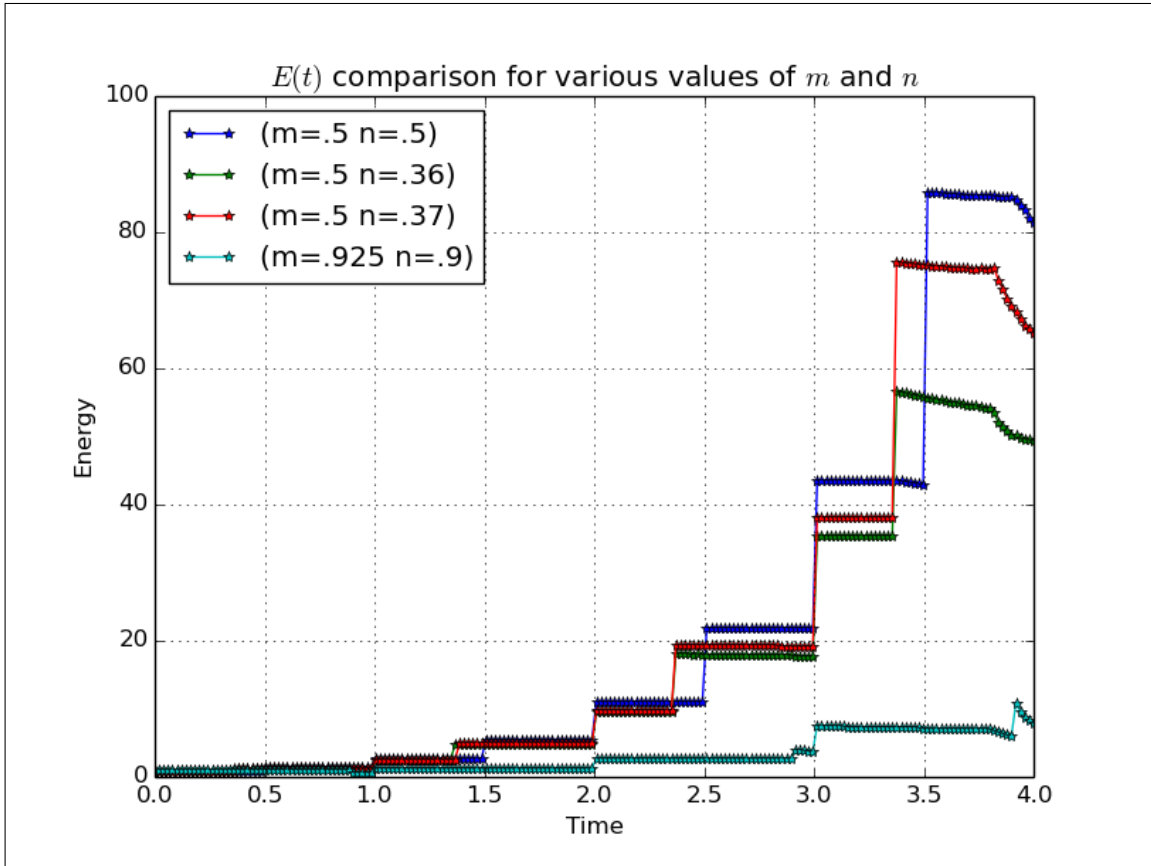


Figure 4.6: Plot showing the evolution of energy growth over time for 4 different combinations of  $m, n$ . In each case, we have apparent energy growth.

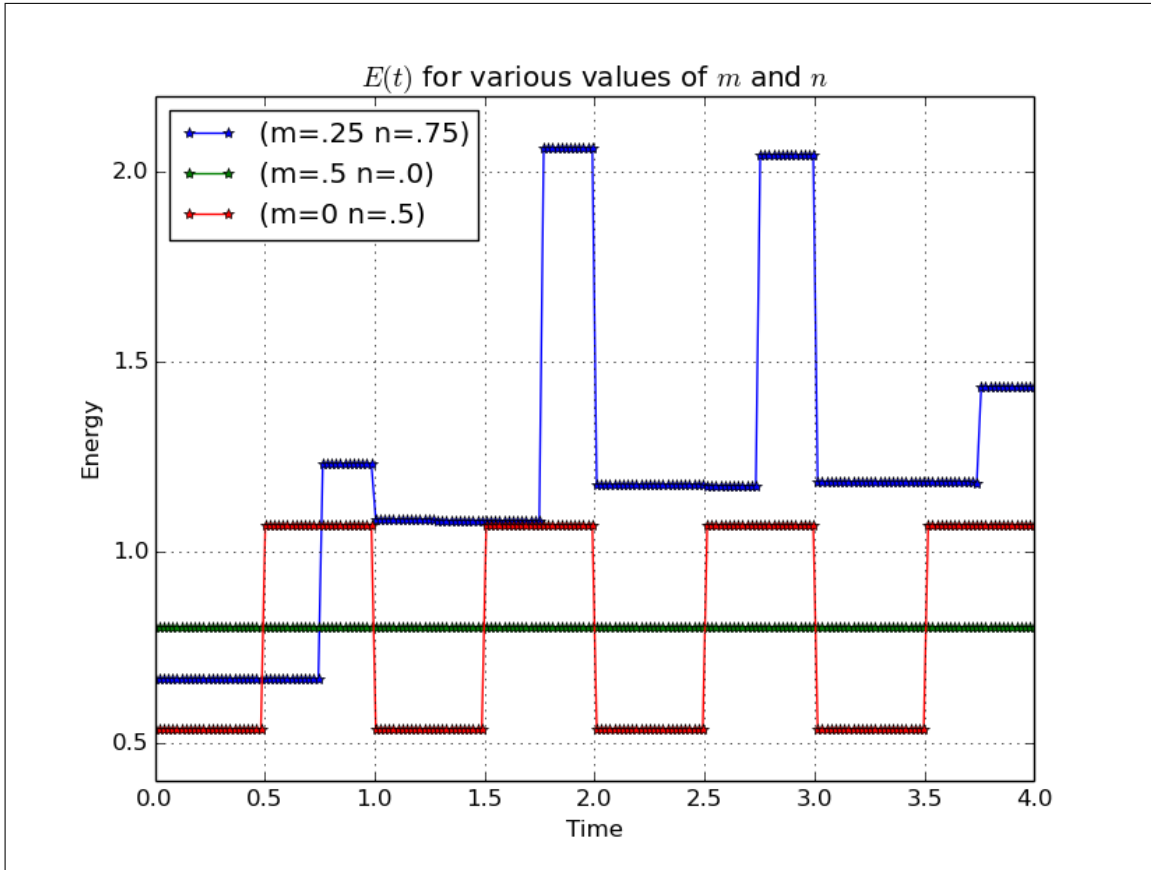


Figure 4.7: Plot showing the evolution of energy growth over time for 3 different combinations of  $m, n$ . In each case, there is no energy growth.



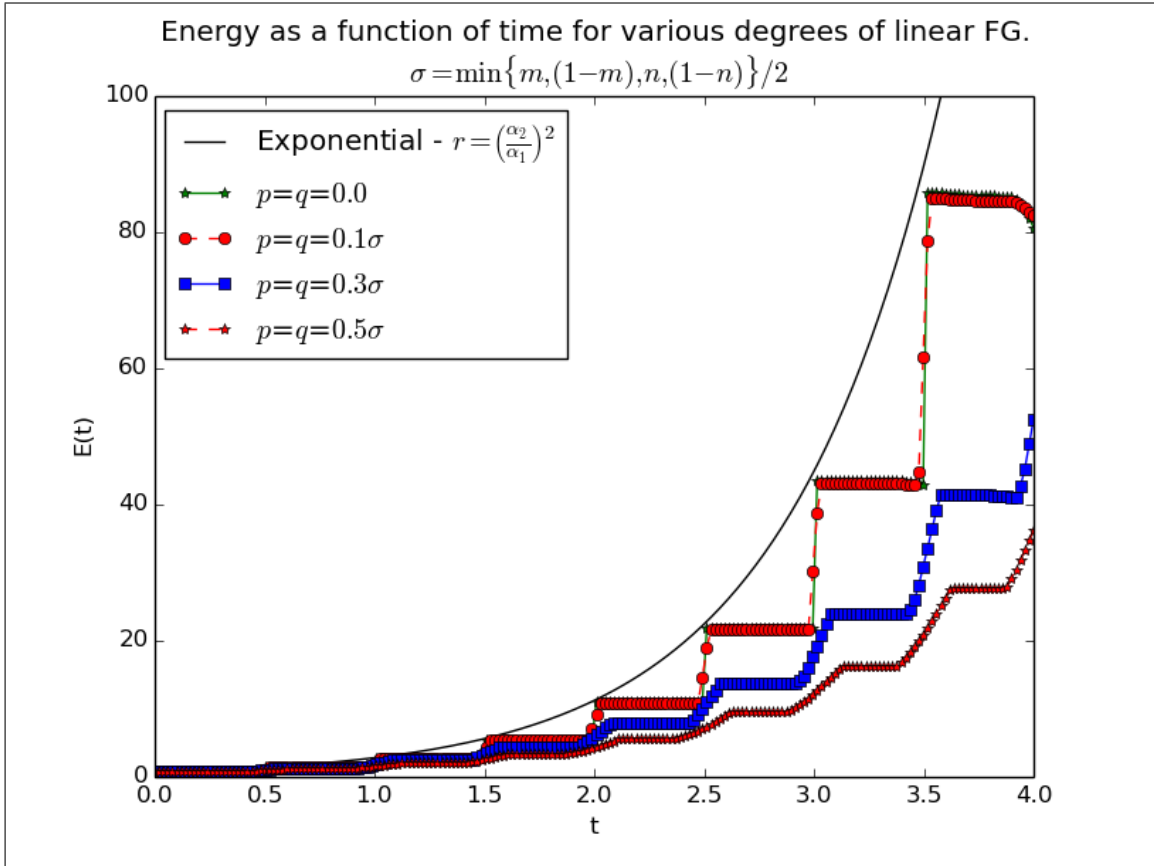


Figure 4.8: Plot showing the evolution of energy growth over time for several different values for the smoothing parameter.

energy over time for several different values of the smoothing parameter. For this image, we choose smoothing parameters to be on the line  $p = q$ , specifically, we choose four different multiples of  $\sigma$ , which is defined as the maximum amount of smoothing that can be allowed, i.e.,  $\sigma = \frac{\min\{m, n, (1-m), (1-n)\}}{2}$ . Specifically, we choose 10%, 30% and 50% of  $\sigma$ .

In each case, there appears to be exponential energy growth and even at a smoothing thickness of  $.1\sigma$  there does not appear to be much difference between the sharp checkerboard and the smooth checkerboard. As the smoothing increases, it seems that there is still exponential growth in the energy. However, it appears that this growth is at a rate less than  $\left(\frac{\alpha_2}{\alpha_1}\right)^2$ . A future research plan is to determine this growth rate using methods similar to those used in [26].

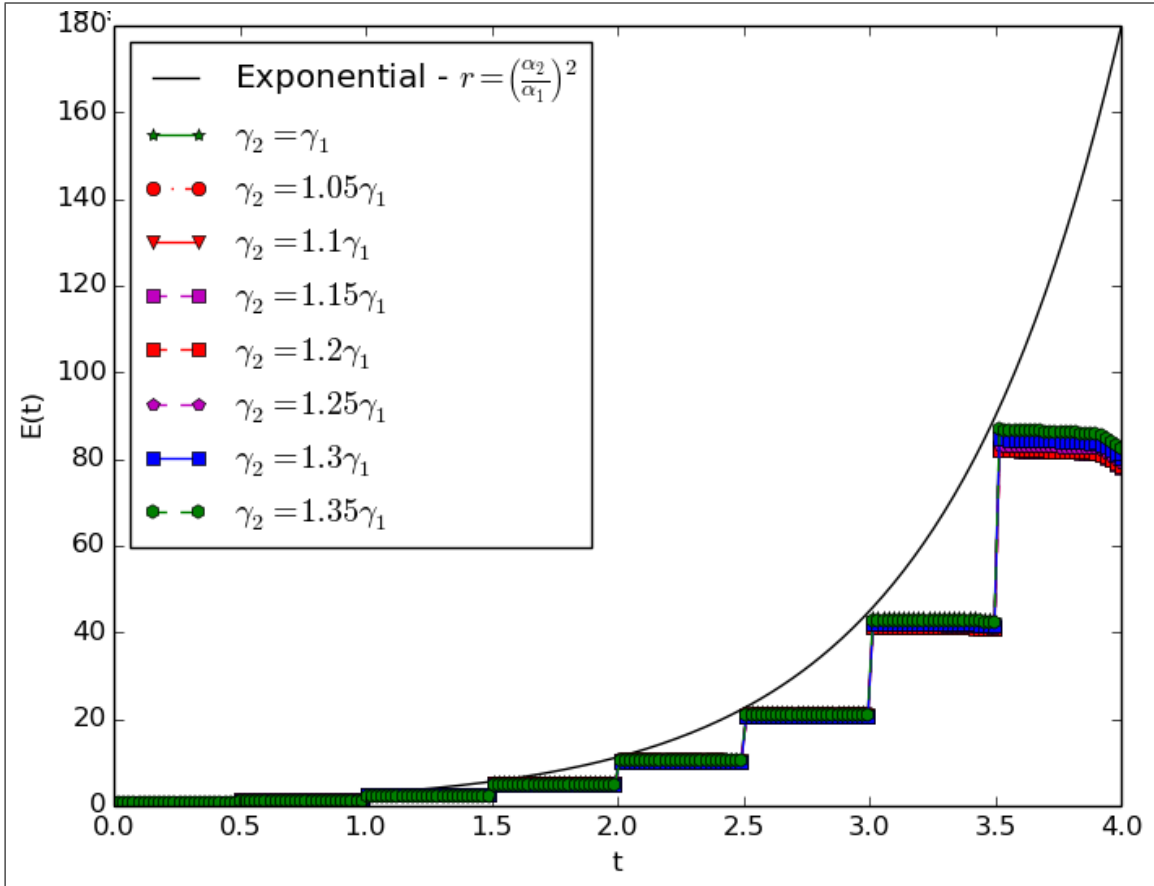


Figure 4.9: Plot showing the evolution of energy growth over time for several different impedance mismatch. In each case, there is exponential energy growth.

### 4.2.3 Energy Accumulation in a Checkerboard with mismatched wave impedances

To show that there is still energy accumulation in the presence of mismatch in wave impedances we plot the evolution of energy over time for several different values of this mismatch. This can be seen in Figure 4.9. In this figure, we compare the energy evolution in time for 8 different elastic wave impedance mismatches. Specifically, we consider up to a 35% difference. In each case, there appears to be exponential energy growth in the total energy.

It is important to emphasize that this figure shows energy accumulation “on the whole” and does not distinguish between the energy carried by the transmitted right-going characteristics or the reflected left-going characteristics. Future theoretical work needs to be done to analyze the effect of impedance mismatch on energy accumulation in right going characteristics compared to left-going characteristics.

#### 4.2.4 Conclusion

In this section we have shown that it is possible to solve for the energy evolution in through a checkerboard structure by implementing of a well-known finite volume method. Using this method, we have shown that the energy accumulation effect remains present with the relaxations of functional grading or mismatch of wave impedances. This is exciting because it shows the robustness of the checkerboard structure when characteristic focusing is in effect. Robustness is a key property that will be useful in physical construction of these materials, because it will certainly be extremely difficult to construct “perfect” structures in actual engineering endeavors.

# Chapter 5

## Conclusion

The preceding chapters have presented several extensions to the concept of dynamic materials. The results presented have been thoroughly investigated and have served to answer several important questions relating to wave propagation in Dynamic Materials. Over the course of this investigation, many new questions have arisen, some of which have been able to be answered in this document, and some of which, will be pursued in future research efforts.

### 5.1 Summary

Allowing for DMs that are more general give future material engineers more flexibility in constructing such materials. We have extended understanding of wave propagation in DMs by showing that the previously discovered effects of characteristic focusing and energy accumulation remain even under more general assumptions.

The results presented in Chapter 2 show that focusing and energy accumulation can occur simultaneously in multiple families or selectively in particular families depending on the material parameters. Practically, this makes it possible to engineer DMs which focus a specific type of elastic wave while leaving the other type untouched. This may be useful in creating devices which detect and amplify  $s$ -waves but leave  $p$ -waves untouched (or vice-versa). Simultaneous focusing implies simultaneous control of multiple families of waves. This idea extends naturally for the more general system of linear conservation laws. Thus, it is conceivable that one could construct a linear checkerboard structure DM that only produces convergence in a selected subset of wave families. The results presented in Chapter 3 show that characteristic focusing occurs for the class of DM termed “Functionally Graded” DM. First, we showed that this focusing can occur in materials with nonlinear function of position and time. Second, we investigated checkerboard focusing in materials with linear change in wave velocity, effectively, extending the class of limit cycles  $\mathcal{C}_{z_0}$  orig-

inally studied in [27] to the class  $\mathcal{C}_{z_0}^{p,q}$ . Practically, this result is very important for many reasons, however, the main reason is that it allows for more opportunities for actively engineering and building these materials. The results presented in Chapter 4 further extend the ideas presented in the preceding chapter and show that energy accumulation is present in the case of Functional Grading and also in the presence of non-matching wave impedances.

I am extremely excited for the future of research in the field of DM. I believe that the results presented in this dissertation will be very helpful in advancing future understanding of wave propagation through DM. I also feel strongly that they will be very useful to future efforts in physically building and engineering such materials.

## 5.2 Future Work

The purpose of this section will be to extend the ideas developed in the previous sections and summarize many new research ideas and applications related to the DM concept. I believe that these ideas will serve as a very good starting point for future research and it is my hope that the ideas presented in this section will be fully developed in future studies.

### 5.2.1 Nonlinear Dynamic Materials

The first idea to be investigated is what we have termed a “non-linear dynamic material”. We define a nonlinear DM as a material that supports nonlinear wave propagation that can be explicitly controlled by a designer in space and time. There are many reasons why considering a non-linear DM is important. Problems from traffic flow, non-linear elasticity, shallow-water flow, etc., can all be interpreted as problems in the field of non-linear dynamic materials. In this regime, material properties should be taken loosely as any system property for which we have a degree of spatial-temporal control. Specifically, we are interested in posing optimization problems related to the optimal control of these properties to minimize/maximize the flux of the solution variable through some boundary or by a certain time.

Specifically, what we mean by a non-linear DM is one that require solutions of systems of  $n$  linear or non-linear conservation laws where one has special spatial temporal control of the flux function, i.e.,

$$\mathbf{q}_{,t} + \mathbf{f}(\mathbf{q}; \mathbf{r}, t)_{,x} + \mathbf{g}(\mathbf{q}; \mathbf{r}, t)_{,y} + \mathbf{h}(\mathbf{q}; \mathbf{r}, t)_{,z} = \psi(\mathbf{q}; \mathbf{r}, \mathbf{t}),$$

along with corresponding boundary and initial conditions satisfied by the quantities. Differential laws of this form are typically derived from integral relationships

expressing conservation of mass, momentum, and energy. Specifically, integration of the system of conservation laws gives

$$\begin{aligned} \int_0^T \int_{\Omega} (\mathbf{q}_t + \operatorname{div}\mathbf{F}) dV dt &= \int_0^T \int_{\Omega} \psi dV dt, \\ \int_0^T \left( \frac{d}{dt} \left( \int_{\Omega} \mathbf{q} dV \right) + \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n} dS \right) dt &= \int_0^T \int_{\Omega} \psi dV dt, \\ \mathbf{Q}(T) - \mathbf{Q}(0) + \int_0^T \int_{\partial\Omega} \mathbf{F} \cdot \mathbf{n} dS dt &= \int_0^T \int_{\Omega} \psi dV dt, \end{aligned}$$

which is to say that the total amount of change in the average value of  $\mathbf{q}$  throughout the domain  $\Omega$  is exactly equal to the amount added/subtracted by source terms  $\psi$  plus/minus the amount that entered/left through the boundary  $\partial\Omega$ .

The main issue in investigating DM is understanding exactly what affect spatial-temporal control of the flux function  $\mathbf{F}$  has on propagating disturbances. It is clear that in the linear regime controlling material properties produces a variety of interesting and non-trivial results, I suspect this will also be true in the non-linear regime. The following section is a discussion of two 1D examples of the above problem.

### 1D Non-linear DM

The natural starting place for this research area is the study of solutions to 1D conservation laws of the form

$$\phi_t + (f(\phi; z, t))_z = 0,$$

where  $f(\phi; z, t)$  is a function of  $\phi$  and is *explicitly dependent* on space and time. Specifically, for many examples we will consider, we will have a flux function of the form  $f(\phi; a(z, t))$  where we term  $a(z, t)$  the control function.

**Ex: Dynamic Burgers Equation** We consider investigation of a variant of the inviscid Burgers equation we have termed the “dynamic Burgers equation”,

$$\phi_t + \left( v(z, t) \frac{\phi^2}{2} \right)_z = 0, \tag{5.1}$$

where  $v(z, t)$  is a function of space and time. We can interpret equation (5.1) as the continuity equation for a line of masses whose velocity is determined simultaneously by how much mass is currently at a point and by a function  $v(z, t)$ . This function can be regarded as either an outside control or possibly as an environmental

factor that is within our control. This is a good equation to start with because it is relatively simple and combines the idea of non-linear wave propagation with the concept of spatial-temporal control. Later on, we will introduce a slightly more complex conservation law which is a better description of traffic flow.

For shock solutions, the Rankine-Hugoniot relationship must be obeyed for any shocks propagating in the medium, i.e.,

$$s [\phi]_{\phi_R}^{\phi_L} = \left[ v(z, t) \frac{\phi^2}{2} \right]_{\phi_R}^{\phi_L},$$

where  $s$  is the local shock speed at  $(z, t)$ .

If  $v(z, t)$  is taken to be a smart dynamic structure, we envision an effect where one can control shocks by prescribing  $v(z, t)$  in a smart manner.

In [8], a version of equation (5.1) was studied with a viscosity term. Specifically, this equation is  $\phi_t + \phi\phi_z = \kappa\phi_{zz}$ . A remarkable approach to solving this non-linear equation was independently discovered in references [10] and [13]. This method is known as the ‘‘Cole-Hopf’’ transformation and has since been applied to linearizing various non-linear PDEs. Many generalizations of this procedure have been developed [14, 6] and we believe that a suitable generalization can be found for the modified dynamic Burgers equation (5.1).

### Traffic Flow

One of the major applications of non-linear DM is to control of traffic flow. Nonlinear traffic flow is an active area of research. Here we present a model that combines traffic flow with dynamic control.

Consider the standard Lighthill-Whitham traffic flow model

$$\rho_t + (f(\rho))_z = 0, \quad z \in [a, b], \quad t \in [0, T),$$

where  $\rho(z, t)$  is the local car density on a one lane road and  $f$  is a suitably chosen function that describes the flux as a function of density. A simple example can be found in reference [16]. In this case,  $f$  is chosen as quadratic function:  $f(\rho) = u_M(1 - \rho)\rho$ , where  $u_M$  is the maximum pointwise velocity of traffic flow (the speed limit). In this scenario,  $u_M$  is a constant, however, I propose that it also makes sense to interpret these quantities as functions of space and time, i.e.,  $u_M(z, t)$ . This alludes to a more general form for the above traffic model, namely, one where  $f$  may depend explicitly on  $z$  or  $t$ ,

$$\rho_t + (f(\rho; z, t))_z = 0, \quad z \in [a, b], \quad t \in [0, T),$$

If we interpret either of these quantities in the above example ( $u_M$  or  $\rho_M$ ) as a control, it becomes possible to construct optimization problems. For example, suppose we control  $u_M$ , the maximum velocity of the traffic flow. Consider determining the  $u_M(x, t)$  which minimizes the flux of cars through certain domains of space-time. Specifically, suppose we wanted to minimize the average density along a specific stretch of road from a point  $c$  to a point  $d$ , i.e.,  $(c, d) \subset (a, b)$  at a given time  $T$ . Mathematically speaking, this is the following problem:

$$\inf_{u_M} \int_c^d \rho(z, T) dz$$

Consider minimizing the above functional subject to the differential constraints imposed by the traffic flow equations. In this effort, we would look into various numerical and/or analytical methods available for functional optimization. In a future research effort, we would further develop these ideas and numerical methods of approaching these and similar problems.

The above model can be extended to different multiple lane models to allow for more complicated traffic analysis/phenomena. It is reasonable to believe that combining these ideas with the idea of spatial-temporal control in traffic flow can serve to enhance understanding of how to optimally control traffic patterns. A potential problem with applying this idea to current traffic situations is that there is necessarily a human driver behind the wheel of the car. This is a potential flaw in the model and one possible way to resolve this issue is the introduction of randomness into the model. However, this may not be as much of an issue as it initially seems to be. A recent research effort is the study of the autonomous or driverless vehicle. A street of autonomous vehicles is a deterministic system and is not subject to random decisions. This is a major application of this theory, i.e., the optimal control of a street of multiple lanes of autonomous vehicles.

The model previously introduced can be expanded to this case in the following manner. Consider two lanes of traffic, with the density of cars in each being given by  $\rho_1$  and  $\rho_2$ , respectively. Assuming traffic in each lane is governed by the continuity equation ensures conservation of mass in each lane; however, to account for possible lane-switching, we must include a source (and sink) term on the right hand side of the equation.

$$\begin{aligned} (\rho_1)_t + (u_{1M}\rho_1(1 - \rho_1))_z &= g_{12}(\rho_1, \rho_2; z, t) \\ (\rho_2)_t + (u_{2M}\rho_2(1 - \rho_2))_z &= -g_{12}(\rho_1, \rho_2; z, t) \end{aligned}$$

where  $g_{12}$  is a function that determines the rate of transfer from lane to lane. A



possible choice is

$$g_{12}(\rho_1, \rho_2) = \begin{cases} 0, & \rho_1 < \rho_S, \rho_2 < \rho_S, \\ \gamma\rho_2, & \rho_1 < \rho_S, \rho_2 \geq \rho_S, \\ -\gamma\rho_1, & \rho_1 \geq \rho_S, \rho_2 < \rho_S, \\ 0, & \rho_1 \geq \rho_S, \rho_2 \geq \rho_S, \end{cases}$$

where  $\gamma$  is a coefficient determining the rate of lane change and  $\rho_S$  is a number in the interval  $(0, \rho_M)$  that determines the density at which vehicles will start switching lanes.

I believe that solving optimization problems for the above system will be important to optimally controlling traffic patterns for streets of smart cars. One can imagine simulating a system of traffic flow equations to optimally control the speed limit on the road to maximize or minimize traffic flow. The results of such a simulation could be very useful in designing, engineering, and maintaining highways or to minimize the frequency of occurrence of traffic jams.

### Nonlinear Elasticity

One limitation of the checkerboard analysis completed so far is the assumption of a linear wave equation. This assumption is valid if the gradient of the solution is small enough, however, the focusing phenomena found when implementing the checkerboard structure will eventually violate this assumption, e.g., it will focus the solution into peaks of extremely high gradient. When solution gradients are high, the fundamental assumptions that are required for linear elasticity break down. This non-linear effect would also be present with the non-linear electrodynamics, only, it would be due to the constitutive relationships.

A more realistic model would either start directly with the full nonlinear equations of elasticity or start with the linear model and take account of the change in governing equations once the gradient of the solution becomes too large in a certain domain. There are many good references for non-linear elasticity [20, 19, 2, 35, 34, 3]. This investigation would make use of reference [1], which gives the mathematical derivation for the vibrations of non-linear strings.

### Asymptotic Analysis of Nonlinear Spatially Inhomogeneous and Spatio-temporally Inhomogeneous DM

One way to investigate a non-linear DM is asymptotic analysis. For example, consider the following PDE:

$$u_t + \left( vu + \epsilon v \frac{u^2}{2} \right)_z = 0, \quad u(z, 0) = \hat{u}(z),$$

where  $v = v(z, t)$  is a prescribed and explicit function. This is the continuity equation perturbed with a small non-linearity. We look for a solution of the form

$$u = u_0 + u_1\epsilon + u_2\epsilon^2 + \dots$$

$$(u_0 + u_1\epsilon + u_2\epsilon^2 + \dots)_t + \left( v(u_0 + u_1\epsilon + u_2\epsilon^2 + \dots) + \epsilon v \frac{(u_0 + u_1\epsilon + u_2\epsilon^2 + \dots)^2}{2} \right)_z = 0.$$

Resulting in the following,

$$(u_0)_t + (vu_0)_z + \left( (u_1)_t + (vu_1)_z + \left( v \frac{(u_0)^2}{2} \right)_z \right) \epsilon + ((u_2)_t + (vu_2)_z + (vu_0u_1)_z) \epsilon^2 + \dots = 0,$$

and the following hierarchical system needs to be solved

$$(u_0)_t + (vu_0)_z = 0 \tag{5.2}$$

$$(u_1)_t + (vu_1)_z = - \left( v \frac{(u_0)^2}{2} \right)_z \tag{5.3}$$

$$(u_2)_t + (vu_2)_z = - (vu_0u_1)_z \tag{5.4}$$

⋮

The solution for  $u_0(z, t)$  can be found by solving (5.2) which is the linear continuity equation governing  $u_0$ . This can be solved exactly for specific choices of  $v(z, t)$ , e.g., a checkerboard or lamination, where  $v(z, t)$  takes on only two values  $v_1$  and  $v_2$ , or it can be solved numerically for more complicated choices of  $v(z, t)$ . Once  $u_0(z, t)$  is available it is possible to solve (5.3) for  $u_1(z, t)$ , and with this solution, we can solve (5.4) using the solutions previously found for  $u_1$  and  $u_0$ . This approach is nice, because it allows us to get a close approximation to the solution of the non-linear conservation law by solving multiple (easier) linear problems.

### 5.2.2 Optimization for Specific Geometries

Another problem to be investigated is optimization of the energy of a wave after it propagates through a checkerboard structure with respect to geometrical or material parameters, e.g., the  $m$  and  $n$  of the checkerboard. The wave  $u(z, t)$  is governed by the following variable coefficient wave equation,

$$(\rho_{mn}u_t)_t - (k_{mn}u_z)_z = 0, \quad t > 0, \\ u(z, 0) = u_0(z), \quad u_t(z, 0) = v_0(z),$$

where  $\rho_{mn}(z, t)$  and  $k_{mn}(z, t)$  are material parameters given by equations (1.9) and (1.10). Figure 1.4 shows a graphical depiction of the material structure.

Given  $\rho_1, \rho_2, k_1, k_2, \delta, \tau$ , and  $T$  we consider the problem of maximizing the final energy  $E$  with respect to  $m$  and  $n$ :

$$\max_{(m,n) \in [0,1] \times [0,1]} E(m, n) = \max_{(m,n) \in [0,1] \times [0,1]} \int_a^b (\rho_{mn}(z, T)u_t^2 + k_{mn}(z, T)u_z^2) dz. \quad (5.5)$$

One question to be answered is whether a maximum exists, and if so, is it unique. This search can be completed over one or multiple periods. The energy is not maximum at the boundaries of the parameter range. The boundary lines of this domain represent either a spatial laminate  $n = 0$  or  $n = 1$  or a temporal laminate  $m = 0$  or  $m = 1$  and the corner points represent a pure material. Energy is *not* accumulated in a spatial lamination, a temporal lamination, or a pure material and thus, the maximum of this function must lie somewhere in the indicated domain, specifically, in the plateau region, where energy is definitely accumulated due to the checkerboard focusing effect. It might be possible to analytically solve for  $E$  explicitly as a function of  $m$  and  $n$  and prove that there is a maximum by using the exact solution over one or two checkerboard periods. Multiple brute force simulations were shown in Chapter 4 showing the region of energy accumulation and confirming the plateau equations derived in reference [27].

# Appendices

# Appendix A

## Code

### A.1 Characteristic Plotter

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 from matplotlib import cm
4 from scipy import ndimage
5 from matplotlib.colors import LightSource
6 from matplotlib import cbook
7
8 import matplotlib as mpl
9 from mpl_toolkits.mplot3d import Axes3D
10 from mayavi import mlab
11
12 #Set the material phase velocities in materials 1 and 2
13 a1=.55;a2=1.1;
14
15 #Set the checkerboard geometrical paramaters
16 tau=1.0;n1=.5;
17 eps=1.0;m1=.5;
18
19 #Set how many spatial-temporal periods the characteristics will be
    plotted over
20 # as well as setting the number of times
21 p_T0=0;p_Tf=10;Nt=1000;
22 p_a =0;p_b =10;Nx=100;
23
24 #Sets the maximum amount of smoothing allowed.
25 Pe=min([m1*eps,(1-m1)*eps,n1*tau,(1-n1)*tau])/2
26
27 #Set the amount of smoothing to be used in FG material
28 alp=(.001)*Pe;bet=(.001)*Pe;
29
30 alpha=alp;beta =bet;
31
32 #Set the boundary of the checkerboard corresponding to the above
```

```

33 T0=p_T0*tau; Tf=p_Tf*tau;
34 a =p_T0*eps; b=p_b*eps;
35
36
37 t_stor=np.linspace(T0,Tf,Nt);
38 X,T = np.mgrid[a:b:Nx*1j, T0:Tf:Nt*1j]; dx=X[1,0]-X[0,0]
39 Xmod,Tmod = np.mgrid[0.0:eps:Nx*1j, 0.0:tau:Nt*1j];
40
41 T0_c=0*tau; Tf_c=Tf;
42 #a_c=alpha+.01;b_c=m1*eps-alpha*eps-.07;
43
44 a_c=-(1-m1)*eps;b_c=eps+.3*m1*eps;
45 N_c=20;
46
47 X_c,T_c = np.mgrid[a_c:b_c:N_c*1j, T0_c:Tf_c:Nt*1j];
48 dt=T_c[0,1]-T_c[0,0];
49
50 ##def LC_Loc1(delta,tau,a1,a2,m,n,alpha,beta):
51 ## lb=a2/a1;
52 ## Orig=(-lb)*delta+a2*tau+(1-lb)*lb*m*delta+(-1+lb)*a2*n*tau;
53 ##
54 ## New= beta*((-3/4)*a2+(1/4)*a1+(1/4)*a2*lb+(1/4)*a2*lb**2)
55 ## return ((Orig+New)/(1.0-lb**2))
56 def LC_Loc1(delta,tau,a1,a2,m,n,alpha,beta):
57 lb=a2/a1;
58 C1=(-lb**2+2*lb*np.log(lb)+1.0)/(lb**2-2*lb+1.0);
59 C2=a2/4*(-1.0+1/lb)
60 C3=lb/(lb+1.0)
61 C4=-a2/(lb+1)
62 C5=(-a2+lb)/(lb**2-1)
63
64 New= C1*alpha+C2*beta+C3*m+C4*m+C5
65 return New
66
67
68 #####Definition of Material Geometry
69 #####
70 ##### Standard Checkerboard
71 #def f_u(x,t,u1,u2):
72 #def M1(x):
73 #return (u1*(np.mod(x,eps)<m1*eps) + u2*(np.mod(x,eps)>=m1*eps)
74 );
75 #def M2(x):
76 #return (u2*(np.mod(x,eps)<m1*eps) + u1*(np.mod(x,eps)>=m1*eps)
77 );
78 #return (M1(x)*(np.mod(t,tau)<n1*tau)+M2(x)*(np.mod(t,tau)>=n1*tau)
79 )
80
81 ##### Functionally Graded Checkerboard (tanh function)
82 #####alpha=.75;
83 #####beta=.75;

```

```

80
81 #alpha=0.4;
82 #beta=0.4;
83
84 #def f_u(x,t,u1,u2):
85     #def smt(x,t):
86         #return np.tanh(np.sin(2*np.pi*x)/alpha)*np.tanh(np.sin(2*np.pi
            *t)/beta)
87     #return (u1+u2)/(2.0)+(u1-u2)/2.0*smt(x,t)
88
89 def l(xi,xil,xi2,y1,y2):
90     m=(y2-y1)/(xi2-xil);
91     return y1+m*(xi-xil)
92
93 def p(xi,eta,xil,xi2,eta1,eta2,y1,y2):
94     mz=(y2 - y1)/(xi2 -xil);
95     mt=(y2 - y1)/(eta2-eta1);
96     return y1+mz*(xi-xil)+mt*(eta-eta1);
97
98 #####One ST inclusion
99
100 #def f_u(x,t,u1,u2):
101     #Wx=eps/10;
102     #Wt=tau/5;
103     #xTi=np.mod(x,eps);
104     #tTi=np.mod(t,tau);
105     #z1=m1*eps-Wx;
106     #z2=m1*eps+Wx;
107     #t1=n1*tau-Wt;
108     #t2=n1*tau+Wt;
109
110     ##def M1(x):
111         ##return u1*(xTi < z1) + l(x,z1,z2,u1,u2)*((z1<=xTi)*(xTi < z2)
            )+u2*(z2<= xTi)
112     ##def M2(x,t):
113         ##return (l(t,t1,t2,u1,u2)*(xTi < z1) +
            ##p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= xTi)*(xTi < z2)*(xTi-z1 <((
            z2-z1)/(t1-t2))*(tTi-t2)))+
114         ##p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= xTi)*(xTi < z2)*(xTi-z1 >((
            z2-z1)/(t1-t2))*(tTi-t2)))+
115         ##l(t,t1,t2,u2,u1)*(z2 <= xTi))
116     ##def M3(x):
117         ##return u2*(xTi < z1) + l(x,z2,z1,u1,u2)*((z1<=xTi)*(xTi < z2)
            )+u1*(z2<= xTi);
118
119
120     #def M1(x):
121         #return u1*(x < z1) + l(x,z1,z2,u1,u2)*((z1<=x)*(x < z2))+u2*(
            z2<= x)
122     #def M2(x,t):
123         #return (l(t,t1,t2,u1,u2)*(x < z1) +
            #p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 <((z2-z1)
124

```

```

125 /((t1-t2))*(t-t2)))+
      #p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 >=((z2-z1)
126 /((t1-t2))*(t-t2)))+
      #l(t,t1,t2,u2,u1)*(z2 <= x))
127 #def M3(x):
128 #return u2*(x < z1) + l(x,z2,z1,u1,u2)*((z1<=x)*(x < z2))+u1*(
z2<= x);
129
130 #def Fin(x,t):
131 #return M1(x)*(t <t1)+M2(x,t)*((t1 <= t)*(t <t2))+M3(x)*(t2<= t)
132
133 #return Fin(xTi,tTi)
134
135
136 #####Linear FG Checkerboard
137 def py(x,z1,z2,t,t1,t2,u1,u2):
138 #return (p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 <((z2-z1)
)/(t1-t2))*(t-t2)))+
139 p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 >=((z2-z1)/(t1-
t2))*(t-t2))))
140
141 Wx=alpha
142 Wt=beta;
143 def f_u(x,t,u1,u2):
144 xTi=np.mod(x,eps);
145 tTi=np.mod(t,tau);
146
147 #def M1(x):
148 #return u1*(xTi < z1) + l(x,z1,z2,u1,u2)*((z1<=xTi)*(xTi < z2))
+u2*(z2<= xTi)
149 #def M2(x,t):
150 #return (l(t,t1,t2,u1,u2)*(xTi < z1) +
151 #p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= xTi)*(xTi < z2)*(xTi-z1 <((
z2-z1)/(t1-t2))*(tTi-t2)))+
152 #p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= xTi)*(xTi < z2)*(xTi-z1 >=((
z2-z1)/(t1-t2))*(tTi-t2)))+
153 #l(t,t1,t2,u2,u1)*(z2 <= xTi))
154 #def M3(x):
155 #return u2*(xTi < z1) + l(x,z2,z1,u1,u2)*((z1<=xTi)*(xTi < z2))
+u1*(z2<= xTi);
156
157 def M0(x,t):
158 #return (py(x,-Wx,Wx,t,-Wt,Wt,u1,u2)*(x<Wx)+
159 l(t,-Wt,Wt,u2,u1)*(x>=Wx)*(x<m1*eps-Wx)+
160 py(x,m1*eps-Wx,m1*eps+Wx,t,-Wt,Wt,u2,u1)*(x>=m1*eps-Wx)
*(x<m1*eps+Wx)+
161 l(t,-Wt,Wt,u1,u2)*(m1*eps+Wx<=x)*(x<eps-Wx)+
162 py(x,eps-Wx,eps+Wx,t,-Wt,Wt,u1,u2)*(x>eps-Wx))
163
164 def M1(x,t,u1,u2):
165 z1=m1*eps-Wx;

```



```

166     z2=m1*eps+Wx;
167     return ( l(x,-Wx,Wx,u2,u1)*(x<Wx)+
168             u1*(Wx<=x)*(x < z1) +
169             l(x,z1,z2,u1,u2)*((z1<=x)*(x < z2))+
170             u2*(z2<= x)*(x<eps-Wx)+
171             l(x,eps-Wx,eps+Wx,u2,u1)*(eps-Wx<=x))
172
173     def M2(x,t):
174         t1=n1*tau-Wt;
175         t2=n1*tau+Wt;
176         z1=m1*eps-Wx;
177         z2=m1*eps+Wx;
178         return (py(x,-Wx,Wx,t,t1,t2,u2,u1)*(x<Wx)+
179                l(t,t1,t2,u1,u2)*(Wx<=x)*(x < z1) +
180                p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 <((z2-z1)/(
181 t1-t2)))*(t-t2)))+
181                p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 >((z2-z1)/(
182 t1-t2)))*(t-t2)))+
182                l(t,t1,t2,u2,u1)*(z2 <= x)*(x<eps-Wx)+
183                py(x,eps-Wx,eps+Wx,t,n1*tau-Wt,n1*tau+Wt,u2,u1)*(eps-Wx<=x))
184 #     def M3(x,t):
185 #         z1=m1*eps-Wx;
186 #         z2=m1*eps+Wx;
187 #         return l(x,-Wx,Wx,u1,u2)*(x<Wx)+u2*(Wx<=x)*(x < z1) + l(x,z2,
188 z1,u1,u2)*((z1<=x)*(x < z2))+u1*(z2<= x)*(x<eps-Wx)+l(x,eps-Wx,eps+
189 Wx,u1,u2)*(eps-Wx<=x);
188     def M3(x,t):
189         return M1(x,t,u2,u1)
190
191     def M4(x,t):
192         return (py(x,-Wx,Wx,t,tau-Wt,tau+Wt,u1,u2)*(x<Wx)+
193                l(t,tau-Wt,tau+Wt,u2,u1)*(x>=Wx)*(x<m1*eps-Wx)+
194                py(x,m1*eps-Wx,m1*eps+Wx,t,tau-Wt,tau+Wt,u2,u1)*(x>=m1*
195 eps-Wx)*(x<m1*eps+Wx)+
195                l(t,tau-Wt,tau+Wt,u1,u2)*(m1*eps+Wx<=x)*(x<eps-Wx)+py(x,eps-Wx,eps+
196 Wx,t,tau-Wt,tau+Wt,u1,u2)*(x>=eps-Wx))
196
197     def Fin(x,t):
198         t1=n1*tau-Wt;
199         t2=n1*tau+Wt;
200     return (M0(x,t)*(t<Wt)+
201            M1(x,t,u1,u2)*(Wt<=t)*(t < t1)+
202            M2(x,t)*((t1 <= t)*(t < t2))+
203            M3(x,t)*(t2<= t)*(t<tau-Wt)+
204            M4(x,t)*(tau-Wt<=t))
205
206     return Fin(xTi,tTi)
207
208 #def f_u(x,t,u1,u2):
209 #    #Wx=eps/10;
210 #    #Wt=tau/10;

```

```

211 #xTi=np.mod(x, eps);
212 #tTi=np.mod(t, tau);
213 #z1=m1*eps-Wx;
214 #z2=m1*eps+Wx;
215 #t1=n1*tau-Wt;
216 #t2=n1*tau+Wt;
217
218 #def pyr(x,y,x1,x2,y1,y2,u1,u2):
219     #m=(y2-y1)/(x1-x)
220     #return (p(x,y,x1,x2,y1,y2,u1,u2)*((x-x1)<m*(y-y1))+
221     #p(x,y,x2,x1,y2,y1,u1,u2)*((x-x1)>=m*(y-y1)))
222
223
224
225 #def M1(x):
226     #return l(x,Wx,-Wx,u1,u2)*(x<Wx)+u1*((Wx<=x)*(x < z1)) + l(x,z1
227     ,z2,u1,u2)*((z1<=x)*(x < z2))+u2*((z2<= x)*(x<eps-Wx))+l(x,eps-Wx,
228     eps+Wx,u2,u1)*(eps-Wx<=x)
229     #def M2(x,t):
230     #return (p(x,t,-Wx,Wx,t1,t2,u2,u1)*((x<Wx)*(x+Wx< ((2*Wx)/(t1-
231     t2))*(t-t2))) +
232     #p(x,t,Wx,-Wx,t2,t1,u2,u1)*((x<Wx)*(x+Wx>=((2*Wx)/(t1-t2))*(t-t2)))
233     +
234     #l(t,t1,t2,u1,u2)*((Wx<=x)*(x < z1)) +
235     #p(x,t,z1,z2,t1,t2,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 <((z2-z1)
236     /(t1-t2))*(t-t2)))+
237     #p(x,t,z2,z1,t2,t1,u1,u2)*((z1 <= x)*(x < z2)*(x-z1 >=((z2-z1)
238     /(t1-t2))*(t-t2)))+
239     #l(t,t1,t2,u2,u1)*((z2<=x)*(x < eps-Wx)) +
240     #pyr(x,t,eps-Wx,eps+Wx,t1,t2,u1,u2)*(eps-Wx<=x))
241     #def M3(x):
242     #return l(x,Wx,-Wx,u2,u1)*(x<Wx)+u2*((Wx<x)*(x < z1)) + l(x,z2,
243     z1,u1,u2)*((z1<=x)*(x < z2))+u1*((z2<= x)*(x<eps-Wx))+l(x,eps-Wx,
244     eps+Wx,u1,u2)*(eps-Wx<=x);
245
246
247     #def Fin(x,t):
248     #return M1(x)*(t <t1)+M2(x,t)*((t1 <= t)*(t <t2))+M3(x)*(t2<= t)
249
250     #return Fin(xTi,tTi)
251
252 ##### Solution of Characteristic Equation
253 C_c=np.zeros(X_c.shape);
254
255 ##### Rk4 Method
256
257 N_unstable=2;
258 X_unstable=np.zeros((N_unstable,X_c.shape[-1]));
259 T_unstable=np.zeros(X_unstable.shape);
260
261 N_stable=2;
262 X_stable=np.zeros((N_stable,X_c.shape[-1]));

```

```

254 T_stable=np.zeros(X_stable.shape);
255
256 T_unstable[:,0]=T0+T_unstable[:,0];
257 Unstable_init=m1*eps+LC_Loc1(eps,tau,a2,a1,1-m1,n1,alpha,beta);
258 X_unstable[:,0]=np.array([Unstable_init,Unstable_init+eps]);
259
260 T_stable[:,0]=T0+T_stable[:,0];
261 Stable_init=LC_Loc1(eps,tau,a1,a2,m1,n1,alpha,beta);
262 X_stable[:,0]=np.array([Stable_init,Stable_init+eps]);
263
264 C_avg=np.zeros(X_c[:,0].shape)
265
266 Np=np.int((5.0/6)*Nt);
267
268 for n in range(Nt-1):
269 #C_c[:,n]=f_u(X_c[:,n],T_c[:,n],a1,a2);
270 k1=f_u(X_c[:,n],T_c[:,n],a1,a2);
271 k2=f_u(X_c[:,n]+k1*dt/2,T_c[:,n]+dt/2,a1,a2);
272 k3=f_u(X_c[:,n]+k2*dt/2,T_c[:,n]+dt/2,a1,a2);
273 k4=f_u(X_c[:,n]+k3*dt,T_c[:,n]+dt,a1,a2);
274 X_c[:,n+1]=X_c[:,n]+(dt/6)*(k1+2*k2+2*k3+k4);
275
276 k1_u=f_u(X_unstable[:,n],T_unstable[:,n],a1,a2);
277 k2_u=f_u(X_unstable[:,n]+k1_u*dt/2,T_unstable[:,n]+dt/2,a1,a2);
278 k3_u=f_u(X_unstable[:,n]+k2_u*dt/2,T_unstable[:,n]+dt/2,a1,a2);
279 k4_u=f_u(X_unstable[:,n]+k3_u*dt,T_unstable[:,n]+dt,a1,a2);
280 X_unstable[:,n+1]=X_unstable[:,n]+(dt/6)*(k1_u+2*k2_u+2*k3_u+k4_u);
T_unstable[:,n+1]=T_unstable[:,n]+dt;
281
282 k1_s=f_u(X_stable[:,n],T_stable[:,n],a1,a2);
283 k2_s=f_u(X_stable[:,n]+k1_s*dt/2,T_stable[:,n]+dt/2,a1,a2);
284 k3_s=f_u(X_stable[:,n]+k2_s*dt/2,T_stable[:,n]+dt/2,a1,a2);
285 k4_s=f_u(X_stable[:,n]+k3_s*dt,T_stable[:,n]+dt,a1,a2);
286 X_stable[:,n+1]=X_stable[:,n]+(dt/6)*(k1_s+2*k2_s+2*k3_s+k4_s);
T_stable[:,n+1]=T_stable[:,n]+dt;
287
288 if t_stor[n]>=t_stor[Np]:
289 C_avg=C_avg+f_u(X_c[:,n],T_c[:,n],a1,a2)
290
291 C_avg=C_avg/np.size(t_stor[Np:-1])
292
293 ##### diff1=np.abs(X_c[:,0][1]-X_c[:,0][0])
294 ##### diff2=np.abs(X_c[:, -1][1]-X_c[:, -1][0])
295
296 ##### TotalDiff=diff2-diff1;
297
298 Np=np.int((2.0/3)*C_c.shape[1]);
299
300 StorAvg=np.zeros(C_c.shape[0]);
301
302 #for n in range(C_c.shape[0]):

```

```

303 # StorAvg[n]=np.average(C_c[4,Np:-1])
304
305 CharNumber=0;
306 Xcmmod=np.mod(X_c,eps);Tcmmod=np.mod(T_c,tau);
307
308 ## Create Custom ColorMap Function
309 def make_cmap(colors, position=None, bit=False):
310     '''
311     make_cmap takes a list of tuples which contain RGB values. The RGB
312     values may either be in 8-bit [0 to 255] (in which bit must be set
313     to
314     True when called) or arithmetic [0 to 1] (default). make_cmap
315     returns
316     a cmap with equally spaced colors.
317     Arrange your tuples so that the first color is the lowest value for
318     the
319     colorbar and the last is the highest.
320     position contains values from 0 to 1 to dictate the location of
321     each color.
322     '''
323     import matplotlib as mpl
324     import numpy as np
325     bit_rgb = np.linspace(0,1,256)
326     if position == None:
327         position = np.linspace(0,1,len(colors))
328     else:
329         if len(position) != len(colors):
330             sys.exit("position length must be the same as colors")
331         elif position[0] != 0 or position[-1] != 1:
332             sys.exit("position must start with 0 and end with 1")
333     if bit:
334         for i in range(len(colors)):
335             colors[i] = (bit_rgb[colors[i][0]],
336                         bit_rgb[colors[i][1]],
337                         bit_rgb[colors[i][2]])
338     cdict = {'red':[], 'green':[], 'blue':[]}
339     for pos, color in zip(position, colors):
340         cdict['red'].append((pos, color[0], color[0]))
341         cdict['green'].append((pos, color[1], color[1]))
342         cdict['blue'].append((pos, color[2], color[2]))
343
344     cmap = mpl.colors.LinearSegmentedColormap('my_colormap',cdict,256)
345     return cmap
346
347 RedCharCmap=make_cmap([(220,220,220),(140,0,26)],bit=True)
348
349 colors = [(100,200,255),(255,255,150)]
350 MyCmap=make_cmap(colors,bit=True)
351
352 ##### Figure-1 #####

```

```

350 plt.figure(1)
351 gr=(0.863,0.863,0.863)
352
353 plt.pcolormesh(X,T,f_u(X,T,a1,a2),cmap=MyCmap)
354 plt.colorbar()
355
356 n_Contours=20;
357
358 #plt.contour(X,T,f_u(X,T,a1,a2),contours=n_Contours,colors='k')
359
360
361 plt.xlim([0.0,1.0*eps+m1*eps])
362 plt.ylim([0.0,1.0*tau])
363
364
365 plt.title('Characteristics for '+r'\delta =$'+str(eps) + \
366 r', \$\tau =$'+str(tau) + ', $m$ =' + str(m1) + ', $n$=' +str(n1)
367 +\
368 r', \$\alpha_1$=' + str(a1) + r', \$\alpha_2$=' + str(a2)+", \n"+
369 r'$p$=' +str(alpha)+r', $q$=' +str(beta)+r', $dt$=' +str(dt))
370 plt.ylabel('$t$ in $['+str(p-T0)+r'\tau, $'+str(p-Tf)+r'\tau$]')
371 plt.xlabel('$z$ in $['+str(p-a )+r'\epsilon, $'+str(p-b) +r'\epsilon$]')
372 )
373 frame1=plt.gca()
374
375 frame1.axes.get_xaxis().set_ticks([])
376 frame1.axes.get_yaxis().set_ticks([])
377
378 plt.plot(np.rot90(X_c),np.rot90(T_c),'k-',linewidth=2.0,color=gr);
379
380 plt.savefig("Checkerboard.png",dpi=1000)
381
382 plt.clf()
383
384 plt.figure(3)
385
386 plt.pcolormesh(X,T,f_u(X,T,a1,a2),cmap=MyCmap)
387 plt.colorbar()
388
389 n_Contours=1;
390
391 #plt.contour(X,T,f_u(X,T,a1,a2),colors='k',linestyle='dotted')
392
393 plt.xlim([0.0,b])
394 plt.ylim([0.0,Tf])
395
396
397 plt.title('Characteristics for '+r'\delta =$'+str(eps) + \
398 r', \$\tau =$'+str(tau) + r', $m$ =' + str(m1) + r', $n$=' +str(n1)

```

```

399     ) +\
400     r', $\alpha_1=$' + str(a1) + r', $\alpha_2$=' + str(a2)+", \n"+
401     r'$p$='+str(alpha)+r', $q$='+str(beta)+r', $dt$='+str(dt))
402 plt.ylabel('$t$ in $'+str(p_T0)+r'$\tau$, $'+str(p_Tf)+r'$\tau$')
403 plt.xlabel('$z$ in $'+str(p_a )+r'$\epsilon$, $'+str(p_b) +r'$\epsilon$')
404 )
405 frame1=plt.gca()
406
407 frame1.axes.get_xaxis().set_ticks([])
408 frame1.axes.get_yaxis().set_ticks([])
409
410 #Char=plt.plot(np.rot90(X_c),np.rot90(T_c),colormap=RedCharCmap,
411               linewidth=1.0);
412
413 p_R=Nt/p_Tf;n_R=p_Tf-2;
414
415 gr=(0.863,0.863,0.863)
416 re=(0.941,0.0,0.102)
417
418 Char=plt.plot(np.rot90(X_c[:,0:n_R*p_R]),np.rot90(T_c[:,0:n_R*p_R]),
419              color=gr,linewidth=1.0);
420
421 Char=plt.plot(np.rot90(X_c[:,n_R*p_R:Nt-1]),np.rot90(T_c[:,n_R*p_R:Nt-1]),
422              color=re,linewidth=1.0);
423 #plt.scatter(np.rot90(X_c),np.rot90(T_c))
424
425 plt.savefig("Checkerboard_Full.png",dpi=1000)
426
427 plt.clf()
428
429 plt.figure(4)
430
431 colors = [(100,200,255),(255,255,150)]
432 MyCmap=make_cmap(colors,bit=True)
433
434 plt.pcolormesh(X,T,f_u(X,T,a1,a2),cmap=MyCmap)
435 plt.colorbar()
436
437 n_Contours=2;
438
439 #plt.contour(X,T,f_u(X,T,a1,a2),contours=n_Contours,colors='k')
440
441 plt.xlim([9*eps,10*eps])
442 plt.ylim([9*tau,10*tau])
443
444 plt.title('Characteristics for '+r'$\delta$='+str(eps) + \
445          r', $\tau$='+str(tau) + ', $m$='+str(m1) + ', $n$='+str(n1)

```

```

445     +\
446     r', $\alpha_1=$' + str(a1) + r', $\alpha_2$=' + str(a2)+"", \n"+
447     r'$p$='+str(alpha)+r', $q$='+str(beta)+r', $dt$='+str(dt))
448 plt.ylabel('$t$ in $['+str(4)+r'$\tau$, $'+str(5)+r'$\tau$]')
449 plt.xlabel('$z$ in $['+str(4)+r'$\delta$, $'+str(5)+r'$\epsilon$]')
450
451 frame1=plt.gca()
452
453 frame1.axes.get_xaxis().set_ticks([])
454 frame1.axes.get_yaxis().set_ticks([])
455
456 plt.plot(np.rot90(X_c),np.rot90(T_c),'-',color=re,linewidth=2.0);
457
458 plt.savefig("Checkerboard-LC.png",dpi=1000)
459
460 plt.clf()
461
462
463
464
465
466 plt.figure(4)
467
468 colors = [(100,200,255),(255,255,150)]
469 MyCmap=make_cmap(colors,bit=True)
470
471 plt.pcolormesh(X,T,f_u(X,T,a1,a2),cmap=MyCmap)
472 plt.colorbar()
473
474 n_Contours=2;
475
476 #plt.contour(X,T,f_u(X,T,a1,a2),contours=n_Contours,colors='k')
477
478 plt.xlim([0.0*eps,3.0*eps+m1*eps])
479 plt.ylim([0.0*tau,3.0*tau])
480
481 plt.title('Checkerboard for '+r'$\delta$ =' + str(eps) + \
482     r', $\tau$ =' + str(tau) + ', $m$ =' + str(m1) + ', $n$ =' + str(n1)
483     +\
484     r', $\alpha_1=$' + str(a1) + r', $\alpha_2$=' + str(a2)+"", \n"+
485     r'$p$='+str(alpha)+r', $q$='+str(beta))
486 plt.ylabel('$t$ in $['+str(0)+r'$\tau$, $'+str(3)+r'$\tau$]')
487 plt.xlabel('$z$ in $['+str(0)+r'$\delta$, $'+str(3)+r'$\delta$]')
488
489 frame1=plt.gca()
490
491 frame1.axes.get_xaxis().set_ticks([])
492 frame1.axes.get_yaxis().set_ticks([])
493

```

```

494 #plt . plot (np . rot90 (X_c) ,np . rot90 (T_c) ,'-',color='darkred',linewidth
      =2.0);
495
496 plt . savefig ("Checkerboard_NoChar . png" ,dpi=1000)
497
498 plt . clf ()
499
500 ##### Figure -2 #####
501
502 ###X_unstable_mod=np . mod (X_unstable , eps) ;T_unstable_mod=np . mod (
      T_unstable , tau) ;
503
504 ###plt . figure (2)
505
506
507 ###plt . pcolormesh (Xmod,Tmod,f_u (Xmod,Tmod,a1 , a2) ,cmap=MyCmap)
508 ###plt . colorbar ()
509
510 ###n_Contours=20;
511
512 ###plt . contour (Xmod,Tmod,f_u (Xmod,Tmod,a1 , a2) ,contours=n_Contours)
513
514
515 #####plt . xlim ([0.0 , eps])
516 #####plt . ylim ([0.0 , tau])
517 #####plt . colorbar ()
518
519 #####plt . title ('Characteristics for '+r'\epsilon ='+str(eps) + '\
520     #####r', '\tau ='+str(tau) + ', m$='+str(m1) + ', n$='+str(
      n1) +\
521     ###', $a_1$='+str(a1) + ', $a_2$='+str(a2)+", \n"+
522     #####r '\alpha$='+str(alpha)+r', '\beta$='+str(beta))
523
524 #####plt . ylabel ('$t$ in $['+str(p_T0)+r'\tau, '$'+str(p_Tf)+r'\tau$
      ]')
525 #####plt . xlabel ('$z$ in $['+str(p_a )+r'\epsilon, '$'+str(p_b) +r'\
      epsilon$]')
526
527 ###frame1=plt . gca ()
528
529 ###frame1 . axes . get_xaxis () . set_ticks ([])
530 ###frame1 . axes . get_yaxis () . set_ticks ([])
531
532
533 #####frame1 . axes . get_xaxis () . set_visible (False)
534 #####frame1 . axes . get_yaxis () . set_visible (False)
535
536 #####XcRot=np . rot90 (Xcmod [CharNumber ,:] ) ;
537 #####TcRot=np . rot90 (Tcmod [CharNumber ,:] ) ;
538
539 ###plt . plot (Xcmod [CharNumber ,:] ,Tcmod [CharNumber ,:] , 'b*',linewidth=1.0)

```



```

540 ;
    #####plt . plot (X_unstable_mod [0 ,:] , T_unstable_mod [0 ,:] , 'g*' , linewidth =3.0)
    ;
541
542
543
544 #####plt . plot (XcRot , TcRot , 'b*' , linewidth =1.0) ;
545
546 ##### plt . plot (np . rot90 (X_unstable) , np . rot90 (T_unstable) , 'k--' ,
    linewidth =3.0) ;
547 #####plt . plot (np . rot90 (X_stable) , np . rot90 (T_stable) , 'k-' , linewidth =3.0) ;
548
549 #####plt . savefig (" Checkerboard . png" , dpi =1000)
550
551 #####plt . savefig ('2 Characteristics for '+r '$\epsilon ='+str (eps) + '\
552     #####r ' , '$\tau ='+ str (tau) + ' , $m$ =' + str (m1) + ' , $n$=' +str (
    n1) +\
553     #####' , $a_1$=' + str (a1) + ' , $a_2$=' + str (a2) \
554     #####+r "$\alpha$"+ str (alpha)+r "$\beta$"+str (beta) \
555     #####+ , p_T0="+str (p_T0)+"P_Tf"+str (p_Tf) \
556     #####+".png" , dpi =100)
557
558 #####plt . savefig ('3 Characteristics for '+r '$\epsilon ='+str (eps) +
    \
559     #####r ' , '$\tau ='+ str (tau) + ' , $m$ =' + str (m1) + ' , $n$=' +
    str (n1) +\
560     ##### , $a_1$=' + str (a1) + ' , $a_2$=' + str (a2) \
561     #####+r "$\alpha$"+ str (alpha)+r "$\beta$"+str (beta) \
562     #####+ , p_T0="+str (p_T0)+"P_Tf"+str (p_Tf) \
563     #####+".png" , dpi =100)
564
565 ##### Figure -4 #####
566
567 #####plt . figure (4)
568
569
570 #####X_Int=np . mod (X_c , eps) ; T_Int=np . mod (T_c , tau) ;
571
572 #####Tol = .01 ;
573
574 #####X_0=np . where (X_Int == 0.0) ;
575 #####T_0=np . where (T_Int == 0.0) ;
576
577 #####Xabs1=np . where (np . abs (X_Int-eps)<Tol) ;
578 #####Xabs2=np . where (np . abs (X_Int-m1*eps)<Tol) ;
579 #####Tabs1=np . where (np . abs (T_Int-tau)<Tol) ;
580 #####Tabs2=np . where (np . abs (T_Int-n1*tau)<Tol) ;
581
582
583
584 #####plt . pcolormesh (X , T , f_u (X , T , a1 , a2) , cmap =MyCmap)

```

```

585 #####plt.colorbar()
586
587 #####n_Contours=20;
588
589 #####plt.contour(X,T,f_u(X,T,a1,a2),contours=n_Contours)
590
591
592 #####plt.xlim([a,b])
593 #####plt.ylim([T0,Tf])
594 #####plt.colorbar()
595
596 #####plt.title('Characteristics for '+r'\epsilon ='+str(eps) + \
597     #####r', '\tau ='+str(tau) + ', m$ ='+str(m1) + ', n$='+str(
598     ###', $a_1$'+str(a1) + ', $a_2$'+str(a2)+", \n"+
599     #####r'\alpha$'+str(alpha)+r', '\beta$'+str(beta))
600
601 #####plt.ylabel('$t$ in $'+str(p-T0)+r'\tau, $'+str(p-Tf)+r'\tau$
602     ]')
603 #####plt.xlabel('$z$ in $'+str(p-a)+r'\epsilon, $'+str(p-b)+r'\epsilon$
604     ]')
605
606 #####frame1=plt.gca()
607 #####frame1.axes.get_xaxis().set_ticks([])
608 #####frame1.axes.get_yaxis().set_ticks([])
609
610 #####frame1.axes.get_xaxis().set_visible(False)
611 #####frame1.axes.get_yaxis().set_visible(False)
612
613 #####XcRot=np.rot90(Xcmod[CharNumber,:]);
614 #####TcRot=np.rot90(Tcmod[CharNumber,:]);
615
616
617 #####Characteristics
618 #####plt.plot(Xcmod[CharNumber,:],Tcmod[CharNumber,:], 'b*',linewidth
619     =1.0);
620
621 #####plt.plot(np.rot90(X_c),np.rot90(T_c), 'b-',linewidth=1.0);
622
623 #####plt.plot(X_c[X_0],T_c[X_0], 'r*')
624 #####plt.plot(X_c[T_0],T_c[T_0], 'r*')
625
626 #####plt.plot(X_c[Xabs1],T_c[Xabs1], 'k*')
627 #####plt.plot(X_c[Xabs2],T_c[Xabs2], 'k*')
628 #####plt.plot(X_c[Tabs1],T_c[Tabs1], 'g*')
629 #####plt.plot(X_c[Tabs2],T_c[Tabs2], 'g*')
630
631 #####plt.plot(XcRot,TcRot, 'b*',linewidth=1.0);

```

```

632 ##### plt.plot(np.rot90(X_unstable),np.rot90(T_unstable),'k--',
        linewidth=3.0);
633 #####plt.plot(np.rot90(X_stable),np.rot90(T_stable),'k-',linewidth=3.0);
634
635 #####plt.savefig("Checkerboard.png",dpi=1000)
636
637 #####plt.savefig('4 Characteristics for '+r'\epsilon =' +str(eps) + \
638     #####r', '\tau =' + str(tau) + ', $m$ =' + str(m1) + ', $n$=' +str(
        n1) +\
639     #####', $a_1$=' + str(a1) + ', $a_2$=' + str(a2) \
640     #####+r'\alpha'+ str(alpha)+r'\beta'+str(beta) \
641     #####+', p_T0="+str(p_T0)+"P_Tf"+str(p_Tf) \
642     #####+'.png',dpi=100)
643
644
645 #####plt.show()
646
647
648
649 #####Torus1
650
651
652 #####R=1.0;r=0.5;
653
654
655 #####NN=100;
656 #####Theta,Phi=np.mgrid[0.0:eps:NN*1j, 0.0:tau:NN*1j];
657
658
659 #####X_Torus =(R+r*np.cos(2*np.pi*Phi/tau))*(np.cos(2*np.pi*Theta/eps))
660 #####Y_Torus =(R+r*np.cos(2*np.pi*Phi/tau))*(np.sin(2*np.pi*Theta/eps))
661 #####Z_Torus = r*np.sin(2*np.pi*Phi/tau)
662
663
664 #####Xc_Torus =(R+r*np.cos(2*np.pi*Tmod/tau))*(np.cos(2*np.pi*Xcmod/
        eps))
665 #####Yc_Torus =(R+r*np.cos(2*np.pi*Tmod/tau))*(np.sin(2*np.pi*Xcmod/
        eps))
666 #####Zc_Torus = r*np.sin(2*np.pi*Tmod/tau)
667
668
669
670 #####mpl.rcParams['legend.fontsize'] = 10
671
672 #####fig3 = plt.figure()
673 #####ax3 = fig3.gca(projection='3d')
674
675
676 #####ax3.plot(Xc_Torus[CharNumber,:], Yc_Torus[CharNumber,:], Zc_Torus[
        CharNumber,:], alpha=.4)
677

```

```

678
679
680 #####G=f_u (Theta ,Phi ,a1 ,a2)
681
682 #####N=(G-G.min ( ) )/(G.max ( )-G.min ( ) )
683
684
685
686 #####ax3.plot_surface (X_Torus , Y_Torus , Z_Torus , cmap=MyCmap, rstride
        =1, cstride=1, facecolors=cm.Pastell (N) ,
687                               #####linewidth=0, antialiased=False ,alpha=.6)
688
689
690
691 #####Torus Plots Using Mayavi
692
693 R=1.0;r=0.5;
694
695 NN=1000;
696 Theta ,Phi=np.mgrid [0.0:eps:NN*1j , 0.0:tau:NN*1j ];
697
698
699 X_Torus =(R+r*np.cos (2*np.pi*Phi/tau ))*(np.cos (2*np.pi*Theta/eps ))
700 Y_Torus =(R+r*np.cos (2*np.pi*Phi/tau ))*(np.sin (2*np.pi*Theta/eps ))
701 Z_Torus = r*np.sin (2*np.pi*Phi/tau )
702
703
704 Xc_Torus =(R+r*np.cos (2*np.pi*Tcmo d/tau ))*(np.cos (2*np.pi*Xcmo d/eps ))
705 Yc_Torus =(R+r*np.cos (2*np.pi*Tcmo d/tau ))*(np.sin (2*np.pi*Xcmo d/eps ))
706 Zc_Torus = r*np.sin (2*np.pi*Tcmo d/tau )
707
708 mpl.rcParams [ 'legend.fontsize ' ] = 10
709
710 G=f_u (Theta ,Phi ,a1 ,a2)
711
712 N=(G-G.min ( ) )/(G.max ( )-G.min ( ) )
713
714 mlab.figure (fgcolor=(0, 0, 0) , bgcolor=(1, 1, 1))
715
716 torus=mlab.mesh (X_Torus , Y_Torus , Z_Torus , scalars=N, colormap='jet ')
717
718 # Retrieve the LUT of the surf object.
719 lut = torus.module_manager.scalar_lut_manager.lut.table.to_array
720 lut=np.array (MyCmap (np.arange (256) ) *255, dtype=int )
721 torus.module_manager.scalar_lut_manager.lut.table = lut
722
723 #char1=mlab.plot3d (Xc_Torus [CharNumber ,:] , Yc_Torus [CharNumber ,:] ,
        Zc_Torus [CharNumber ,:] , color=(0.5,0.5,0.5) , line_width=0.1,
        tube_radius=.02)
724 char1=mlab.plot3d (Xc_Torus [CharNumber ,:] , Yc_Torus [CharNumber ,:] ,
        Zc_Torus [CharNumber ,:] , np.log (t_stor+1) , line_width=0.1, tube_radius

```

```

    =.02)
725 lut_char1=char1.module_manager.scalar_lut_manager.lut.table.to_array()
726
727 lut_char1=np.array(RedCharCmap(np.arange(256))*255,dtype=int)
728 char1.module_manager.scalar_lut_manager.lut.table = lut_char1
729
730
731 jp=1;
732 char2=mlab.plot3d(Xc_Torus[CharNumber+jp,:],Yc_Torus[CharNumber+jp,:],
    Zc_Torus[CharNumber+jp,:],np.log(t_stor+1),line_width=0.1,
    tube_radius=.02)
733 char2.module_manager.scalar_lut_manager.lut.table = lut_char1
734
735 jp=2;
736 char3=mlab.plot3d(Xc_Torus[CharNumber+jp,:],Yc_Torus[CharNumber+jp,:],
    Zc_Torus[CharNumber+jp,:],np.log(t_stor+1),line_width=0.1,
    tube_radius=.02)
737 char3.module_manager.scalar_lut_manager.lut.table = lut_char1
738
739 jp=3;
740 char4=mlab.plot3d(Xc_Torus[CharNumber+jp,:],Yc_Torus[CharNumber+jp,:],
    Zc_Torus[CharNumber+jp,:],np.log(t_stor+1),line_width=0.1,
    tube_radius=.02)
741 char4.module_manager.scalar_lut_manager.lut.table = lut_char1
742
743 jp=4;
744 char5=mlab.plot3d(Xc_Torus[CharNumber+jp,:],Yc_Torus[CharNumber+jp,:],
    Zc_Torus[CharNumber+jp,:],np.log(t_stor+1),line_width=0.1,
    tube_radius=.02)
745 char5.module_manager.scalar_lut_manager.lut.table = lut_char1
746
747 # Nice view from the front
748 #mlab.view(.0, - 5.0, 4)
749 mlab.show()

```

## A.2 Inequality Plotter

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sympy.assumptions.assume import global_assumptions
5 from sympy import *
6
7 m,n,alpha,beta,a1,a2,l,delta,tau=sp.symbols('m n alpha beta a1 a2
    lambda delta tau')
8
9 w_1,w_2,w_3=sp.symbols('w_1 w_2 w_3')
10
11 t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9=sp.symbols('t_1 t_2 t_3 t_4 t_5 t_6
    t_7 t_8 t_9')
12

```

```

13 x_1 , x_2 , x_3 , x_4 , x_5 , x_6 , x_7 , x_8 , x_9=sp . symbols ( 'x_1 x_2 x_3 x_4 x_5 x_6
    x_7 x_8 x_9 ' )
14
15 t1 , t2 , z1 , z2 , t , xS , tS=sp . symbols ( 't1 t2 z1 z2 t xS tS ' )
16
17 p , q=sp . symbols ( 'p , q ' )
18
19 sp . init _ printing ( ) # use _ unicode = True )
20
21 # A2 = .6 ; A1 = 3.1 ;
22 # A1 = .55 ;
23 # A2 = 3 * A1 ;
24
25 A1 = .55 ;
26 # A2 = 3 * A1 ;
27 A2 = 1.1
28
29 # A2 = .55 ;
30 # A1 = 3 * A2 ;
31
32
33 eps = 1.0 ; tau = 1.0 ; pm = 0.5 ; qn = 0.3 ;
34
35
36 Pe = min ( [ pm * eps , ( 1 - pm ) * eps , qn * tau , ( 1 - qn ) * tau ] ) / 2
37
38 alp = (.000001) * Pe ;
39 bet = (.000001) * Pe ;
40
41
42
43
44
45 Beta = bet ; Alph = alp ;
46
47 # A1 = .6 ; A2 = 2.1 ;
48
49 L = A2 / A1 ;
50
51 Delta = eps ; Tau = tau ;
52
53
54 #
55
56 z = sp . Function ( ' z ' ) ;
57
58 solution_T = dsolve ( sp . Derivative ( z ( t ) , t ) - ( a1 + ( a2 - a1 ) / ( t2 - t1 ) * ( t - t1 ) ) , z ( t
    ) )
59 solution_S = dsolve ( sp . Derivative ( z ( t ) , t ) - ( a1 + ( a2 - a1 ) / ( z2 - z1 ) * ( z ( t ) - z1 ) )
    , z ( t ) )
60

```

```

61
62 C1=sp.Symbol('C1')
63 A=sp.Symbol('A')
64
65 IC_T=sp.solve(xS-solution_T.rhs.subs(C1,A).subs(t,tS),A)[0]
66 IC_S=sp.solve(xS-solution_S.rhs.subs(C1,A).subs(t,tS),A)[0]
67
68
69
70
71 numer_T,denom_T=sp.simplify(solution_T.rhs.subs(C1,IC_T)).
    as_numer_denom()
72
73 numer_S,denom_S=sp.simplify(solution_S.rhs.subs(C1,IC_S)).
    as_numer_denom()
74
75
76
77 numer_T_Check=sp.simplify(numer_T.subs({t2:q,tS:0,xS:w_1,t1:-q}).subs({
    t:q}))
78 denom_T_Check=sp.simplify(denom_T.subs({t2:q,tS:0,xS:w_1,t1:-q}).subs({
    t:q}))
79
80
81
82 a1T,a2T=sp.symbols('a1T,a2T')
83 numer_S_R=numer_S.subs({a1:a2T,a2:a1T}).subs({a2T:a2,a1T:a1})
84 denom_S_R=denom_S.subs({a1:a2T,a2:a1T}).subs({a2T:a2,a1T:a1})
85
86
87
88
89
90
91 numer_S_R_Check=sp.simplify(numer_S_R.subs({z1:m*delta-p,z2:m*delta+p,
    xS:m*delta-p,tS:t2}))
92 denom_S_R_Check=sp.simplify(denom_S_R.subs({z1:m*delta-p,z2:m*delta+p,
    xS:m*delta-p,tS:t2}))
93
94
95 solve_t3=sp.solve(m*delta+p-numer_S_R_Check/denom_S_R_Check,t)[0]
96
97
98
99 #solution_T2=dsolve(sp.Derivative(z(t),t)-(a2+(a1-a2)/(t2-t1)*(t-t1)),z
    (t))
100 #solution_S2 =dsolve(sp.Derivative(z(t),t)-(a2+(a1-a2)/(z2-z1)*(z(t)-z1
    )),z(t))
101
102 #IC_T2=sp.solve(xS-solution_T2.rhs.subs(C1,A).subs(t,tS),A)[0]
103 #IC_S2=sp.solve(xS-solution_S2.rhs.subs(C1,A).subs(t,tS),A)[0]

```

```

104
105 #numer_T2,denom_T2=sp.simplify(solution_T2.rhs.subs(C1,IC_T2)).
    as_numer_denom()
106 #numer_S2,denom_S2=sp.simplify(solution_S2.rhs.subs(C1,IC_S2)).
    as_numer_denom()
107
108
109 #numer_T2_Check=sp.simplify( numer_T2.subs({t2:tau,tS:0,xS:w_1,t1:-tau})
    .subs({t:tau})/(-2*tau))
110
111
112
113
114
115
116
117
118
119
120
121
122 #Original Expressions
123
124 #C_alpha, C_beta
125 C0_beta =beta -(sp.Min(n,1-n)/2)
126 C0_alpha=alpha -(sp.Min(m,1-m)/2)
127
128 #C_0
129 C0_1=-w_1
130 C0_2=w_1-x_1
131
132
133 #C1, x_1
134 x_1_E=w_1+(1*a1/4+3*a2/4)*beta;
135 x_1_E=x_1_E.subs(a1,a2/1)
136
137 C1_1=w_1-x_1
138 C1_2=x_1-(m-alpha)
139
140
141 #C2, t_2
142 t_2_E=beta+(1/a2)*(m-alpha-x_1)
143
144 C2_1=beta-t_2
145 C2_2=t_2-t_3
146
147
148 #C3, t_3
149 t_3_E=t_2+(2*alpha/(a2-a1))*sp.ln(a2/a1)
150 t_3_E=t_3_E.subs(a1,a2/1)
151

```



```

152 C3_1=t_2-t_3
153 C3_2=t_3-(n-beta)
154
155
156 #C4, x_4
157 x_4_E=m+alpha+a1*(n-beta-t_3)
158 x_4_E=x_4_E.subs(a1,a2/1)
159
160 C4_1=m-x_4
161 C4_2=x_4-x_5
162
163
164 #C5, x_5
165 x_5_E=x_4+beta*(a1+a2)
166 x_5_E=x_5_E.subs(a1,a2/1)
167
168 C5_1=x_4-x_5
169 C5_2=x_5-(1-alpha)
170
171
172 #C6, x_6
173 t_6_E=(n+beta)+(1/a2)*(1-alpha-x_5)
174
175 C6_1=(n+beta)-t_6
176 C6_2=t_6-t_7
177
178
179 #C7, t_7
180 t_7_E=t_6+(2*alpha/(a2-a1))*sp.log(a2/a1)
181 t_7_E=t_7_E.subs(a1,a2/1)
182
183
184 C7_1=t_6-t_7
185 C7_2=t_7-(1-beta)
186
187
188 #C8, x_8
189 x_8_E=1+alpha+a1*((1-beta)-t_7)
190 x_8_E=x_8_E.subs(a1,a2/1)
191
192 C8_1=1+alpha-x_8
193 C8_2=x_8-x_9
194
195 #C9, x_9
196 x_9_E=x_8+(1*a2/4+3*a1/4)*beta
197 x_9_E=x_9_E.subs(a1,a2/1)
198
199 C9_1=x_8-x_9
200 C9_2=x_9-(1+m)
201
202

```

```

203
204
205 ##Substitutions for w1
206
207 x_9_S=x_9_E.subs(x_8,x_8_E)
208 x_9_S=x_9_S.subs(t_7,t_7_E)
209 x_9_S=x_9_S.subs(t_6,t_6_E)
210 x_9_S=x_9_S.subs(x_5,x_5_E)
211 x_9_S=x_9_S.subs(x_4,x_4_E)
212 x_9_S=x_9_S.subs(t_3,t_3_E)
213 x_9_S=x_9_S.subs(t_2,t_2_E)
214 x_9_S=x_9_S.subs(x_1,x_1_E)
215
216 w_3=x_9_S-1;
217
218
219 w3_N,w3_D=sp.simplify(w_3,[m,n,alpha,beta]).as_numer_denom()
220
221 w3_N_Coef=sp.collect(sp.expand(w3_N),[alpha,beta,m,n,w_1],evaluate=
    False)
222
223 w3_Coef={}
224
225 w3_Coef[0]=[w_1,alpha,beta,m,n,sympify(1)]
226 w3_Coef[1]=[sp.simplify(w3_N_Coef[w_1]/w3_D),sp.simplify(w3_N_Coef[
    alpha]/w3_D),\
227             sp.simplify(w3_N_Coef[beta]/w3_D),\
228             sp.simplify(w3_N_Coef[m]/w3_D),
229             sp.simplify(w3_N_Coef[n]/w3_D),
230             sp.simplify(w3_N_Coef[sympify(1)]/w3_D)]
231
232
233
234
235
236 d,e=sp.solve(w_1-w_3,w_1)[0].as_numer_denom()
237
238 e=sp.factor(e)
239
240 h=sp.collect(d,[m,n,alpha,beta],evaluate=False)
241
242 LimitCycleCoef={}
243
244 LimitCycleCoef[0]=[alpha,beta,m,n,sympify(1)]
245 LimitCycleCoef[1]=[sp.simplify(h[alpha]/e),\
246                   sp.simplify(h[beta]/e),\
247                   sp.simplify(h[m]/e),
248                   sp.simplify(h[n]/e),
249                   sp.simplify(h[sympify(1)]/e)]
250
251

```

```

252
253
254 #Substitutions for C
255
256 ##### C9
257
258 ###C9_1
259
260 C9_1_S=C9_1.subs(x_9,x_9_E)
261 C9_1_S=C9_1_S.subs(x_8,x_8_E)
262 C9_1_S=C9_1_S.subs(t_7,t_7_E)
263 C9_1_S=C9_1_S.subs(t_6,t_6_E)
264 C9_1_S=C9_1_S.subs(x_5,x_5_E)
265 C9_1_S=C9_1_S.subs(x_4,x_4_E)
266 C9_1_S=C9_1_S.subs(t_3,t_3_E)
267 C9_1_S=C9_1_S.subs(t_2,t_2_E)
268 C9_1_S=C9_1_S.subs(x_1,x_1_E)
269
270 ###w_1
271
272 C9_1_S=C9_1_S.subs(w_1,d/e)
273
274 C9_1_Num,C9_1_Den=C9_1_S.as_numer_denom()
275 C9_1_Num_S=sp.collect(sp.expand(C9_1_Num),[m,n,alpha,beta],evaluate=
    False)
276
277 C9_1_Num_S[beta]=sp.factor(C9_1_Num_S[beta])
278
279
280 ###C9_2
281
282 C9_2_S=C9_2.subs(x_9,x_9_E)
283 C9_2_S=C9_2_S.subs(x_8,x_8_E)
284 C9_2_S=C9_2_S.subs(t_7,t_7_E)
285 C9_2_S=C9_2_S.subs(t_6,t_6_E)
286 C9_2_S=C9_2_S.subs(x_5,x_5_E)
287 C9_2_S=C9_2_S.subs(x_4,x_4_E)
288 C9_2_S=C9_2_S.subs(t_3,t_3_E)
289 C9_2_S=C9_2_S.subs(t_2,t_2_E)
290 C9_2_S=C9_2_S.subs(x_1,x_1_E)
291
292 ###w_1
293
294 C9_2_S=C9_2_S.subs(w_1,d/e)
295
296 C9_2_Num,C9_2_Den=C9_2_S.as_numer_denom()
297 C9_2_Num_S=sp.collect(sp.expand(C9_2_Num),[m,n,alpha,beta],evaluate=
    False)
298
299 C9_2_Num_S[alpha]=sp.factor(C9_2_Num_S[alpha])
300 C9_2_Num_S[beta]=sp.factor(C9_2_Num_S[beta])

```

```

301 C9_2_Num_S[m          ]=sp.factor(C9_2_Num_S[m          ])
302 C9_2_Num_S[n          ]=sp.factor(C9_2_Num_S[n          ])
303 C9_2_Num_S[sympify(1)]=sp.factor(C9_2_Num_S[sympify(1)])
304
305
306
307
308 ##### C8
309
310 ###C8_1
311
312 C8_1_S=C8_1.subs(x_8,x_8_E)
313 C8_1_S=C8_1_S.subs(t_7,t_7_E)
314 C8_1_S=C8_1_S.subs(t_6,t_6_E)
315 C8_1_S=C8_1_S.subs(x_5,x_5_E)
316 C8_1_S=C8_1_S.subs(x_4,x_4_E)
317 C8_1_S=C8_1_S.subs(t_3,t_3_E)
318 C8_1_S=C8_1_S.subs(t_2,t_2_E)
319 C8_1_S=C8_1_S.subs(x_1,x_1_E)
320
321 ##w_1
322
323 C8_1_S=C8_1_S.subs(w_1,d/e)
324
325 C8_1_Num,C8_1_Den=C8_1_S.as_numer_denom()
326 C8_1_Num_S=sp.collect(sp.expand(C8_1_Num),[m,n,alpha,beta],evaluate=
    False)
327
328 C8_1_Num_S[alpha      ]=sp.factor(C8_1_Num_S[alpha      ])
329 C8_1_Num_S[beta       ]=sp.factor(C8_1_Num_S[beta       ])
330 C8_1_Num_S[m          ]=sp.factor(C8_1_Num_S[m          ])
331 C8_1_Num_S[n          ]=sp.factor(C8_1_Num_S[n          ])
332 C8_1_Num_S[sympify(1)]=sp.factor(C8_1_Num_S[sympify(1)])
333
334
335 ###C8_2
336
337 C8_2_S=C8_2.subs(x_9,x_9_E)
338 C8_2_S=C8_2_S.subs(x_8,x_8_E)
339 C8_2_S=C8_2_S.subs(t_7,t_7_E)
340 C8_2_S=C8_2_S.subs(t_6,t_6_E)
341 C8_2_S=C8_2_S.subs(x_5,x_5_E)
342 C8_2_S=C8_2_S.subs(x_4,x_4_E)
343 C8_2_S=C8_2_S.subs(t_3,t_3_E)
344 C8_2_S=C8_2_S.subs(t_2,t_2_E)
345 C8_2_S=C8_2_S.subs(x_1,x_1_E)
346
347 ##w_1
348
349 C8_2_S=C8_2_S.subs(w_1,d/e)
350

```

```

351 C8_2_Num, C8_2_Den=C8_2_S.as_numer_denom()
352 C8_2_Num_S=sp.collect(sp.expand(C8_2_Num), [m,n,alpha,beta], evaluate=
    False)
353
354
355 C8_2_Num_S[beta      ]=sp.factor(C8_2_Num_S[beta      ])
356
357
358
359
360
361
362 ##### C7
363 ###C7_1
364
365 C7_1_S=C7_1.subs(t_7, t_7_E)
366 C7_1_S=C7_1_S.subs(t_6, t_6_E)
367 C7_1_S=C7_1_S.subs(x_5, x_5_E)
368 C7_1_S=C7_1_S.subs(x_4, x_4_E)
369 C7_1_S=C7_1_S.subs(t_3, t_3_E)
370 C7_1_S=C7_1_S.subs(t_2, t_2_E)
371 C7_1_S=C7_1_S.subs(x_1, x_1_E)
372
373 ##w_1
374
375 C7_1_S=C7_1_S.subs(w_1, d/e)
376
377 C7_1_Num, C7_1_Den=C7_1_S.as_numer_denom()
378 C7_1_Num_S=sp.collect(sp.expand(C7_1_Num), [m,n,alpha,beta], evaluate=
    False)
379
380 C7_1_Num_S[alpha      ]=sp.factor(C7_1_Num_S[alpha      ])
381
382
383 ###C7_2
384
385 C7_2_S=C7_2.subs(t_7, t_7_E)
386 C7_2_S=C7_2_S.subs(t_6, t_6_E)
387 C7_2_S=C7_2_S.subs(x_5, x_5_E)
388 C7_2_S=C7_2_S.subs(x_4, x_4_E)
389 C7_2_S=C7_2_S.subs(t_3, t_3_E)
390 C7_2_S=C7_2_S.subs(t_2, t_2_E)
391 C7_2_S=C7_2_S.subs(x_1, x_1_E)
392
393 ##w_1
394
395 C7_2_S=C7_2_S.subs(w_1, d/e)
396
397 C7_2_Num, C7_2_Den=C7_2_S.as_numer_denom()
398 C7_2_Num_S=sp.collect(sp.expand(C7_2_Num), [m,n,alpha,beta], evaluate=
    False)

```

```

399
400
401 C7_2_Num_S [alpha      ]=sp.factor(C7_2_Num_S [alpha      ])
402 C7_2_Num_S [beta       ]=sp.factor(C7_2_Num_S [beta       ])
403 C7_2_Num_S [m          ]=sp.factor(C7_2_Num_S [m          ])
404 C7_2_Num_S [n          ]=sp.factor(C7_2_Num_S [n          ])
405 C7_2_Num_S [sympify(1)]=sp.factor(C7_2_Num_S [sympify(1)])
406
407
408
409
410
411
412 ##### C6
413 ###C6_1
414
415 C6_1_S=C6_1.subs(t_7,t_7_E)
416 C6_1_S=C6_1_S.subs(t_6,t_6_E)
417 C6_1_S=C6_1_S.subs(x_5,x_5_E)
418 C6_1_S=C6_1_S.subs(x_4,x_4_E)
419 C6_1_S=C6_1_S.subs(t_3,t_3_E)
420 C6_1_S=C6_1_S.subs(t_2,t_2_E)
421 C6_1_S=C6_1_S.subs(x_1,x_1_E)
422
423 ##w_1
424
425 C6_1_S=C6_1_S.subs(w_1,d/e)
426
427 C6_1_Num,C6_1_Den=C6_1_S.as_numer_denom()
428 C6_1_Num_S=sp.collect(sp.expand(C6_1_Num),[m,n,alpha,beta],evaluate=
      False)
429
430 C6_1_Num_S [alpha      ]=sp.factor(C6_1_Num_S [alpha      ])
431 C6_1_Num_S [beta       ]=sp.factor(C6_1_Num_S [beta       ])
432 C6_1_Num_S [m          ]=sp.factor(C6_1_Num_S [m          ])
433 C6_1_Num_S [n          ]=sp.factor(C6_1_Num_S [n          ])
434 C6_1_Num_S [sympify(1)]=sp.factor(C6_1_Num_S [sympify(1)])
435
436
437 ###C6_2
438
439 C6_2_S=C6_2.subs(t_7,t_7_E)
440 C6_2_S=C6_2_S.subs(t_6,t_6_E)
441 C6_2_S=C6_2_S.subs(x_5,x_5_E)
442 C6_2_S=C6_2_S.subs(x_4,x_4_E)
443 C6_2_S=C6_2_S.subs(t_3,t_3_E)
444 C6_2_S=C6_2_S.subs(t_2,t_2_E)
445 C6_2_S=C6_2_S.subs(x_1,x_1_E)
446
447 ##w_1
448

```

```

449 C6_2_S=C6_2_S.subs(w_1,d/e)
450
451 C6_2_Num,C6_2_Den=C6_2_S.as_numer_denom()
452 C6_2_Num_S=sp.collect(sp.expand(C6_2_Num),[m,n,alpha,beta],evaluate=
    False)
453
454
455 C6_2_Num_S[alpha      ]=sp.factor(C6_2_Num_S[alpha      ])
456
457
458
459
460
461
462 ##### C5
463 ###C5_1
464
465 C5_1_S=C5_1.subs(x_5,x_5_E)
466 C5_1_S=C5_1_S.subs(x_4,x_4_E)
467 C5_1_S=C5_1_S.subs(t_3,t_3_E)
468 C5_1_S=C5_1_S.subs(t_2,t_2_E)
469 C5_1_S=C5_1_S.subs(x_1,x_1_E)
470
471 ##w_1
472
473 C5_1_S=C5_1_S.subs(w_1,d/e)
474
475 C5_1_Num,C5_1_Den=C5_1_S.as_numer_denom()
476 C5_1_Num_S=sp.collect(sp.expand(C5_1_Num),[m,n,alpha,beta],evaluate=
    False)
477
478 C5_1_Num_S[beta      ]=sp.factor(C5_1_Num_S[beta      ])
479
480
481
482 ###C5_2
483
484 C5_2_S=C5_2.subs(x_5,x_5_E)
485 C5_2_S=C5_2_S.subs(x_4,x_4_E)
486 C5_2_S=C5_2_S.subs(t_3,t_3_E)
487 C5_2_S=C5_2_S.subs(t_2,t_2_E)
488 C5_2_S=C5_2_S.subs(x_1,x_1_E)
489
490 ##w_1
491
492 C5_2_S=C5_2_S.subs(w_1,d/e)
493
494 C5_2_Num,C5_2_Den=C5_2_S.as_numer_denom()
495 C5_2_Num_S=sp.collect(sp.expand(C5_2_Num),[m,n,alpha,beta],evaluate=
    False)
496

```

```

497
498 C5_2_Num_S [alpha      ]=sp.factor(C5_2_Num_S [alpha      ])
499 C5_2_Num_S [beta       ]=sp.factor(C5_2_Num_S [beta       ])
500 C5_2_Num_S [m          ]=sp.factor(C5_2_Num_S [m          ])
501 C5_2_Num_S [n          ]=sp.factor(C5_2_Num_S [n          ])
502 C5_2_Num_S [sympify(1)]=sp.factor(C5_2_Num_S [sympify(1)])
503
504
505
506
507
508
509
510 ##### C4
511 ###C4.1
512
513 C4_1_S=C4_1.subs(x_5,x_5_E)
514 C4_1_S=C4_1_S.subs(x_4,x_4_E)
515 C4_1_S=C4_1_S.subs(t_3,t_3_E)
516 C4_1_S=C4_1_S.subs(t_2,t_2_E)
517 C4_1_S=C4_1_S.subs(x_1,x_1_E)
518
519 ##w_1
520
521 C4_1_S=C4_1_S.subs(w_1,d/e)
522
523 C4_1_Num,C4_1_Den=C4_1_S.as_numer_denom()
524 C4_1_Num_S=sp.collect(sp.expand(C4_1_Num),[m,n,alpha,beta],evaluate=
      False)
525
526 C4_1_Num_S [alpha      ]=sp.factor(C4_1_Num_S [alpha      ])
527 C4_1_Num_S [beta       ]=sp.factor(C4_1_Num_S [beta       ])
528 C4_1_Num_S [m          ]=sp.factor(C4_1_Num_S [m          ])
529 C4_1_Num_S [n          ]=sp.factor(C4_1_Num_S [n          ])
530 C4_1_Num_S [sympify(1)]=sp.factor(C4_1_Num_S [sympify(1)])
531
532
533
534
535 ###C4.2
536
537 C4_2_S=C4_2.subs(x_5,x_5_E)
538 C4_2_S=C4_2_S.subs(x_4,x_4_E)
539 C4_2_S=C4_2_S.subs(t_3,t_3_E)
540 C4_2_S=C4_2_S.subs(t_2,t_2_E)
541 C4_2_S=C4_2_S.subs(x_1,x_1_E)
542
543 ##w_1
544
545 C4_2_S=C4_2_S.subs(w_1,d/e)
546

```



```

547 C4_2_Num, C4_2_Den=C4_2_S.as_numer_denom()
548 C4_2_Num_S=sp.collect(sp.expand(C4_2_Num), [m,n,alpha,beta], evaluate=
      False)
549
550 C4_2_Num_S[beta      ]=sp.factor(C4_2_Num_S[beta      ])
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566 ##### C3
567 ###C3_1
568
569 C3_1_S=C3_1.subs(t_3, t_3_E)
570 C3_1_S=C3_1_S.subs(t_2, t_2_E)
571 C3_1_S=C3_1_S.subs(x_1, x_1_E)
572
573 ##w_1
574
575 C3_1_S=C3_1_S.subs(w_1, d/e)
576
577 C3_1_Num, C3_1_Den=C3_1_S.as_numer_denom()
578 C3_1_Num_S=sp.collect(sp.expand(C3_1_Num), [m,n,alpha,beta], evaluate=
      False)
579
580
581 C3_1_Num_S[alpha      ]=sp.factor(C3_1_Num_S[alpha      ])
582
583
584
585 ###C3_2
586
587 C3_2_S=C3_2.subs(t_3, t_3_E)
588 C3_2_S=C3_2_S.subs(t_2, t_2_E)
589 C3_2_S=C3_2_S.subs(x_1, x_1_E)
590
591 ##w_1
592
593 C3_2_S=C3_2_S.subs(w_1, d/e)
594
595 C3_2_Num, C3_2_Den=C3_2_S.as_numer_denom()

```

```

596 C3_2_Num_S=sp.collect(sp.expand(C3_2_Num),[m,n,alpha,beta],evaluate=
      False)
597
598
599 C3_2_Num_S[alpha]=sp.factor(C3_2_Num_S[alpha])
600 C3_2_Num_S[beta]=sp.factor(C3_2_Num_S[beta])
601 C3_2_Num_S[m]=sp.factor(C3_2_Num_S[m])
602 C3_2_Num_S[n]=sp.factor(C3_2_Num_S[n])
603 C3_2_Num_S[sympify(1)]=sp.factor(C3_2_Num_S[sympify(1)])
604
605
606
607
608
609
610
611
612
613
614
615
616 ##### C2
617 ###C2_1
618
619 C2_1_S=C2_1.subs(t_3,t_3_E)
620 C2_1_S=C2_1_S.subs(t_2,t_2_E)
621 C2_1_S=C2_1_S.subs(x_1,x_1_E)
622
623 ##w_1
624
625 C2_1_S=C2_1_S.subs(w_1,d/e)
626
627 C2_1_Num,C2_1_Den=C2_1_S.as_numer_denom()
628 C2_1_Num_S=sp.collect(sp.expand(C2_1_Num),[m,n,alpha,beta],evaluate=
      False)
629
630
631 C2_1_Num_S[alpha]=sp.factor(C2_1_Num_S[alpha])
632 C2_1_Num_S[beta]=sp.factor(C2_1_Num_S[beta])
633 C2_1_Num_S[m]=sp.factor(C2_1_Num_S[m])
634 C2_1_Num_S[n]=sp.factor(C2_1_Num_S[n])
635 C2_1_Num_S[sympify(1)]=sp.factor(C2_1_Num_S[sympify(1)])
636
637
638
639 ###C2_2
640
641 C2_2_S=C2_2.subs(t_3,t_3_E)
642 C2_2_S=C2_2_S.subs(t_2,t_2_E)
643 C2_2_S=C2_2_S.subs(x_1,x_1_E)
644

```

```

645 ##w_1
646
647 C2_2_S=C2_2_S.subs(w_1,d/e)
648
649 C2_2_Num,C2_2_Den=C2_2_S.as_numer_denom()
650 C2_2_Num_S=sp.collect(sp.expand(C2_2_Num),[m,n,alpha,beta],evaluate=
    False)
651
652
653 C2_2_Num_S[alpha      ]=sp.factor(C2_2_Num_S[alpha      ])
654
655
656
657
658
659 ##### C1
660 ###C1_1
661 C1_1_S=C1_1.subs(x_1,x_1_E)
662
663 ##w_1
664
665 C1_1_S=C1_1_S.subs(w_1,d/e)
666
667 C1_1_Num,C1_1_Den=C1_1_S.as_numer_denom()
668 C1_1_Num_S=sp.collect(sp.expand(C1_1_Num),[m,n,alpha,beta],evaluate=
    False)
669
670 C1_1_Num_S[beta      ]=sp.factor(C1_1_Num_S[beta      ])
671
672
673 ###C1_2
674 C1_2_S=C1_2.subs(x_1,x_1_E)
675
676 ##w_1
677 C1_2_S=C1_2_S.subs(w_1,d/e)
678
679 C1_2_Num,C1_2_Den=C1_2_S.as_numer_denom()
680 C1_2_Num_S=sp.collect(sp.expand(C1_2_Num),[m,n,alpha,beta],evaluate=
    False)
681
682
683 C1_2_Num_S[alpha      ]=sp.factor(C1_2_Num_S[alpha      ])
684 C1_2_Num_S[beta      ]=sp.factor(C1_2_Num_S[beta      ])
685 C1_2_Num_S[m          ]=sp.factor(C1_2_Num_S[m          ])
686 C1_2_Num_S[n          ]=sp.factor(C1_2_Num_S[n          ])
687 C1_2_Num_S[sympify(1)]=sp.factor(C1_2_Num_S[sympify(1)])
688
689
690
691
692 ##### C0

```

```

693 ###C0_1
694 C0_1_S=C0_1.subs(x_1,x_1_E)
695
696 ##w_1
697
698 C0_1_S=C0_1_S.subs(w_1,d/e)
699
700 C0_1_Num,C0_1_Den=C0_1_S.as_numer_denom()
701 C0_1_Num_S=sp.collect(sp.expand(C0_1_Num),[m,n,alpha,beta],evaluate=
    False)
702
703 C0_1_Num_S[alpha]=sp.factor(C0_1_Num_S[alpha])
704 C0_1_Num_S[beta]=sp.factor(C0_1_Num_S[beta])
705 C0_1_Num_S[m]=sp.factor(C0_1_Num_S[m])
706 C0_1_Num_S[n]=sp.factor(C0_1_Num_S[n])
707 C0_1_Num_S[sympify(1)]=sp.factor(C0_1_Num_S[sympify(1)])
708
709 ###C0_2
710 C0_2_S=C0_2.subs(x_1,x_1_E)
711
712 ##w_1
713 C0_2_S=C0_2_S.subs(w_1,d/e)
714
715 C0_2_Num,C0_2_Den=C0_2_S.as_numer_denom()
716 C0_2_Num_S=sp.collect(sp.expand(C0_2_Num),[m,n,alpha,beta],evaluate=
    False)
717
718 C0_2_Num_S[beta]=sp.factor(C0_2_Num_S[beta])
719
720
721
722
723
724
725
726
727 Coef={}
728
729 Coef[0]=['m','n','alpha','beta','1']
730
731 #No m,n, or 1 term in C1_1_S
732 Coef[1]=[0,0,\
733         0,sp.simplify(C1_1_Num_S[beta]/C1_1_Den),\
734         0]
735
736 Coef[2]=[sp.simplify(C1_2_Num_S[m]/C1_2_Den),sp.simplify(C1_2_Num_S[
    n
    ]/C1_2_Den),\
737         sp.simplify(C1_2_Num_S[alpha]/C1_2_Den),sp.simplify(C1_2_Num_S
    [beta]/C1_2_Den),\
738         sp.simplify(C1_2_Num_S[sympify(1)]/C1_2_Den)]
739

```

```

740 Coef[3]=[sp.simplify(C2_1_Num_S[m      ]/C2_1_Den),sp.simplify(
      C2_1_Num_S[n      ]/C2_1_Den),\
741      sp.simplify(C2_1_Num_S[alpha      ]/C2_1_Den),sp.simplify(
      C2_1_Num_S[beta      ]/C2_1_Den),\
742      sp.simplify(C2_1_Num_S[sympify(1)]/C2_1_Den)]
743
744 Coef[4]=[0,0,\
745      0,sp.simplify(C2_2_Num_S[alpha]/C2_2_Den),\
746      0]
747
748 Coef[5]=[0,0,\
749      sp.simplify(C3_1_Num_S[alpha]/C3_1_Den),0,\
750      0]
751
752
753 Coef[6]=[sp.simplify(C3_2_Num_S[m      ]/C3_2_Den),sp.simplify(C3_2_Num_S
      [n      ]/C3_2_Den),\
754      sp.simplify(C3_2_Num_S[alpha]/C3_2_Den),sp.simplify(C3_2_Num_S
      [beta      ]/C3_2_Den),\
755      sp.simplify(C3_2_Num_S[sympify(1)]/C3_2_Den) ]
756
757
758
759 Coef[7]=[sp.simplify(C4_1_Num_S[m      ]/C4_1_Den),sp.simplify(C4_1_Num_S
      [n      ]/C4_1_Den),\
760      sp.simplify(C4_1_Num_S[alpha]/C4_1_Den),sp.simplify(C4_1_Num_S
      [beta      ]/C4_1_Den),\
761      sp.simplify(C4_1_Num_S[beta ]/C4_1_Den) ]
762
763
764 Coef[8]=[0,0,\
765      0,sp.simplify(C4_2_Num_S[beta      ]/C4_2_Den),\
766      0]
767
768
769 Coef[7]=[0,0,\
770      0,sp.simplify(C5_1_Num_S[beta      ]/C5_1_Den),\
771      0]
772
773
774 Coef[8]=[sp.simplify(C5_2_Num_S[m      ]/C5_2_Den),sp.simplify(
      C5_2_Num_S[n      ]/C5_2_Den),\
775      sp.simplify(C5_2_Num_S[alpha      ]/C5_2_Den),sp.simplify(
      C5_2_Num_S[beta      ]/C5_2_Den),\
776      sp.simplify(C5_2_Num_S[sympify(1)]/C5_2_Den) ]
777
778
779 Coef[9]=[sp.simplify(C6_1_Num_S[m      ]/C6_1_Den),sp.simplify(
      C6_1_Num_S[n      ]/C6_1_Den),\
780      sp.simplify(C6_1_Num_S[alpha      ]/C6_1_Den),sp.simplify(
      C6_1_Num_S[beta      ]/C6_1_Den),\

```

```

781     sp . simplify ( C6_1_Num_S [sympify (1)] / C6_1_Den ) ]
782
783 Coef [10] = [0, 0, \
784             sp . simplify ( C6_2_Num_S [alpha
785             ] / C6_2_Den ) , 0, \
786             0]
787 Coef [11] = [0, 0, \
788             sp . simplify ( C7_1_Num_S [alpha
789             ] / C7_1_Den ) , 0, \
790             0]
791 Coef [12] = [sp . simplify ( C7_2_Num_S [m
792             ] / C7_2_Den ) , sp . simplify (
793             C7_2_Num_S [n
794             ] / C7_2_Den ) , \
795             sp . simplify ( C7_2_Num_S [alpha] / C7_2_Den ) , sp . simplify (
796             C7_2_Num_S [beta
797             ] / C7_2_Den ) , \
798             sp . simplify ( C7_2_Num_S [sympify (1)] / C7_2_Den ) ]
799
800 Coef [13] = [sp . simplify ( C8_1_Num_S [m
801             ] / C8_1_Den ) , sp . simplify (
802             C8_1_Num_S [n
803             ] / C8_1_Den ) , \
804             sp . simplify ( C8_1_Num_S [alpha] / C8_1_Den ) , sp . simplify (
805             C8_1_Num_S [beta
806             ] / C8_1_Den ) , \
807             sp . simplify ( C8_1_Num_S [sympify (1)] / C8_1_Den ) ]
808
809 Coef [14] = [0, 0, \
810             0, sp . simplify ( C8_2_Num_S [beta
811             ] / C8_2_Den ) , \
812             0]
813 Coef [15] = [0, 0, \
814             0, sp . simplify ( C9_1_Num_S [beta
815             ] / C9_1_Den ) , \
816             0]
817 Coef [16] = [sp . simplify ( C9_2_Num_S [m
818             ] / C9_2_Den ) , sp . simplify (
819             C9_2_Num_S [n
820             ] / C9_2_Den ) , \
821             sp . simplify ( C9_2_Num_S [alpha] / C9_2_Den ) , sp . simplify (
822             C9_2_Num_S [beta
823             ] / C9_2_Den ) , \
            sp . simplify ( C9_2_Num_S [sympify (1)] / C9_2_Den ) ]
824
825 Coef [17] = [sp . simplify ( C0_1_Num_S [m
826             ] / C0_1_Den ) , sp . simplify (
827             C0_1_Num_S [n
828             ] / C0_1_Den ) , \
829             sp . simplify ( C0_1_Num_S [alpha] / C0_1_Den ) , sp . simplify (
830             C0_1_Num_S [beta
831             ] / C0_1_Den ) , \
832             sp . simplify ( C0_1_Num_S [sympify (1)] / C0_1_Den ) ]
833
834 Coef [18] = [0, 0, \
835             0, sp . simplify ( C0_2_Num_S [beta
836             ] / C0_2_Den ) , \
837             0]

```

```

824 L_Coef={}; Lines={}; N_Lines={}; R_Lines={};
825
826 L_Coef[0]=[ 'i+1', 'm', 'alpha', 'beta', '1' ]
827 Lines[0]= 'exact lines '
828 N_Lines[0]= 'numerical lines '
829
830 k=1
831
832
833
834
835
836 for i in range(18):
837     if Coef[i+1][1] !=0:
838         L_Coef[k]=[i+1,sp.simplify(-Coef[i+1][0]/Coef[i+1][1]),\
839             sp.simplify(-Coef[i+1][2]/Coef[i+1][1]),\
840             sp.simplify(-Coef[i+1][3]/Coef[i+1][1]),\
841             sp.simplify(-Coef[i+1][4]/Coef[i+1][1])]
842
843         k=k+1
844
845         Lines[i+1]=Coef[i+1][0]*m +Coef[i+1][1]*n\
846             +Coef[i+1][2]*alpha+Coef[i+1][3]*beta +Coef[i+1][4]
847         R_Lines[i+1]=sp.factor(Lines[i+1].subs({beta:0,alpha:0}))
848         N_Lines[i+1]=Lines[i+1].subs({a2:A2,l:L,alpha:Alph,beta:Beta})
849
850 nmLines=[]
851
852 LineNumbers=[2,3,6,8,9,12,13,16,17]
853
854 C_2_8_Coef={}; C_2_8_Coef[0]={str(2)+'_'+str(8)}
855 C_16_8_Coef={}; C_16_8_Coef[0]={str(16)+'_'+str(8)}
856 C_6_12_Coef={}; C_6_12_Coef[0]={str(6)+'_'+str(12)}
857 C_6_17_Coef={}; C_6_17_Coef[0]={str(6)+'_'+str(17)}
858 for i in range(4):
859     C_2_8_Coef[i+1]=sp.factor(L_Coef[2][i+1]-L_Coef[8][i+1])
860     C_16_8_Coef[i+1]=sp.factor(L_Coef[16][i+1]-L_Coef[8][i+1])
861     C_6_12_Coef[i+1]=sp.factor(L_Coef[6][i+1]-L_Coef[12][i+1])
862     C_6_17_Coef[i+1]=sp.factor(L_Coef[6][i+1]-L_Coef[17][i+1])
863
864
865
866 AllIneq=sp.And(N_Lines[2]<0.0, N_Lines[3]<0.0, N_Lines[6]<0.0, N_Lines
867     [8]<0.0, N_Lines[9]<0.0,\
868     N_Lines[12]<0.0, N_Lines[13]<0.0, N_Lines[16]<0.0,
869     N_Lines[17]<0.0,\
870     Beta<n/2, Beta<(1-n)/2, Alph<(m/2), Alph<(1-m)/2 )
871
872

```

```

873
874
875
876 l2=sp.lambdify(m,sp.solve(N_Lines[2],n)[0], "numpy")
877 l3=sp.lambdify(m,sp.solve(N_Lines[3],n)[0], "numpy")
878
879 l6=sp.lambdify(m,sp.solve(N_Lines[6],n)[0], "numpy")
880
881 l8=sp.lambdify(m,sp.solve(N_Lines[8],n)[0], "numpy")
882 l9=sp.lambdify(m,sp.solve(N_Lines[9],n)[0], "numpy")
883
884 l12=sp.lambdify(m,sp.solve(N_Lines[12],n)[0], "numpy")
885 l13=sp.lambdify(m,sp.solve(N_Lines[13],n)[0], "numpy")
886
887 l16=sp.lambdify(m,sp.solve(N_Lines[16],n)[0], "numpy")
888
889 l17=sp.lambdify(m,sp.solve(N_Lines[17],n)[0], "numpy")
890
891
892
893 m_grid=np.linspace(0,1,20);
894 n_grid=np.linspace(0,1,20);
895
896
897
898 p_Ineq=sp.plot_implicit(AllIneq,(m,0.0,1.0),(n,0.0,1.0),depth=2,xlabel=
      r '$m$',ylabel=r '$n$'\
899   ,title=r 'Plateau Region for $\alpha_2$='+str(A2)+r ', $\lambda=\frac{\alpha_2}{\alpha_1}$'+str(A2/A1)\
      +r ', $p$'+str(Alph)+r ', and $q$'+str(Beta) )
900
901
902
903
904 p0_alpha_1=plt.plot(2*Alph*np.ones(n_grid.shape),n_grid,'-r');
905 p0_alpha_2=plt.plot(1-2*Alph*np.ones(n_grid.shape),n_grid,'-r');
906
907 p0_beta_1=plt.plot(m_grid,2*Beta*np.ones(m_grid.shape),'.b');
908 p0_beta_2=plt.plot(m_grid,1-2*Beta*np.ones(m_grid.shape),'-b');
909
910 p2=plt.plot(m_grid,l2(m_grid),'<',color='darkolivegreen');
911 #p3=plt.plot(m_grid,l3(m_grid),'-o',color='Red');
912
913 p6=plt.plot(m_grid,l6(m_grid),'-s',color='darkviolet');
914
915 p8=plt.plot(m_grid,l8(m_grid),'-8',color='seagreen');
916 #p9=plt.plot(m_grid,l9(m_grid),'-o',color='Orange');
917
918 p12=plt.plot(m_grid,l12(m_grid),'-p',color='mediumpurple');
919 #p13=plt.plot(m_grid,l13(m_grid),'g-o');
920
921 #p16=plt.plot(m_grid,l16(m_grid),'m-D');

```



```

922
923 #p17=plt . plot (m_grid , l17(m_grid) , 'k-d' ) ;
924
925
926 plt . xlabel (r '$m$' )
927 plt . ylabel (r '$n$' )
928
929
930 plt . title (r 'Plateau Region for $a_2$=' + str (A2) + r ', $\lambda = \frac{a_2}{a_1}$' + str (A2/A1) \
931             + r ', $p$=' + str (Alpha) + r ', and $q$=' + str (Beta))
932
933 plt . xlim ([0 , 1])
934 plt . ylim ([0 , 1])
935
936 plt . scatter ([.5] , [.5] , color='r' , marker='^' )
937 plt . scatter ([.9] , [.925] , color='r' , marker='>' )
938 plt . scatter ([.5] , [.36] , color='r' , marker='<' )
939 plt . scatter ([.5] , [.37] , color='r' , marker='v' )
940
941 #plt . legend ([r '$p < \frac{m}{2}$' , r '$p < \frac{1-m}{2}$' , \
942             #r '$q < \frac{n}{2}$' , r '$q < \frac{1-n}{2}$' , \
943             #r '$1_1$' , r '$1_2$' , r '$1_3$' , r '$1_4$' , '(m,n) = (.5 , .5)' , '(m,n) = (.9 , .925)' , '(m,n) = (.5 , .37)' , '(m,n) = (.5 , .36)' ] , \
944             #loc='upper left' )
945
946 plt . show ()
947
948
949 sp . lambdify (m , sp . solve (N_Lines [17] , n) [0] , "numpy" )
950
951
952 F_2_8 = C_2_8_Coef [2] * alpha + C_2_8_Coef [3] * beta + C_2_8_Coef [4]
953 F_2_8_Num , F_2_8_Den = F_2_8 . as_numer_denom ()
954 F_2_8_Num = sp . factor (F_2_8_Num)
955
956 F_16_8 = C_16_8_Coef [2] * alpha + C_16_8_Coef [3] * beta + C_16_8_Coef [4]
957 F_16_8_Num , F_16_8_Den = F_16_8 . as_numer_denom ()
958 F_16_8_Num = sp . factor (F_16_8_Num)
959
960 F_6_12 = C_6_12_Coef [2] * alpha + C_6_12_Coef [3] * beta + C_6_12_Coef [4]
961 F_6_12_Num , F_6_12_Den = F_6_12 . as_numer_denom ()
962 F_6_12_Num = sp . factor (F_6_12_Num)
963
964 F_6_17 = C_6_17_Coef [2] * alpha + C_6_17_Coef [3] * beta + C_6_17_Coef [4]
965 F_6_17_Num , F_6_17_Den = F_6_17 . as_numer_denom ()
966 F_6_17_Num = sp . factor (F_6_17_Num)
967
968
969 f_2_8 = sp . lambdify (1 , (F_2_8_Num / ((1-1) * a2 * (1+1))) . subs ({ alpha : Alpha , beta : Beta , a2 : A2} ) , "numpy" )

```

```

970 f_16_8=sp.lambdify(1,(F_16_8_Num/((1-1)*a2*(1+1))).subs({alpha:Alph,
    beta:Beta,a2:A2}),"numpy")
971 f_6_12=sp.lambdify(1,(F_6_12_Num/((1-1)*a2*(1+1))).subs({alpha:Alph,
    beta:Beta,a2:A2}),"numpy")
972 f_6_17=sp.lambdify(1,(F_6_17_Num/((1-1)*a2*(1+1))).subs({alpha:Alph,
    beta:Beta,a2:A2}),"numpy")
973
974 Lambda=np.linspace(0.0001,2,2000)
975
976 p_Line=plt.plot(Lambda,np.zeros(Lambda.shape),'Red')
977 p_2_8=plt.plot(Lambda,f_2_8(Lambda),'-',color='Blue')
978 p_16_8=plt.plot(Lambda,f_16_8(Lambda),'-',color='Green')
979 p_6_12=plt.plot(Lambda,f_6_12(Lambda),'-',color='Purple')
980 p_6_17=plt.plot(Lambda,f_6_17(Lambda),'-',color='Aqua')
981
982 plt.plot(L,f_2_8(L),'o',color="Blue")
983 plt.plot(L,f_16_8(L),'o',color="Green")
984 plt.plot(L,f_6_12(L),'o',color="Purple")
985 plt.plot(L,f_6_17(L),'o',color="Aqua")
986
987 plt.xlabel(r'\lambda$')
988 plt.ylabel(r'Constraint')
989 plt.legend(['y=0',r'$1_2-1_8<0$',r'$1_{16}-1_8<0$',r'$1_6-1_{12}<0$',r'
    $1_6-1_{17}<0$'])
990
991 plt.title(r'Constraints Guaranteeing Existence of Plateau Region for
    $a_2$='+str(A2)+r', $\lambda=\frac{a_2}{a_1}=$'+str(A2/A1)\
    +r', $\alpha=$'+str(Alph)+r', and $\beta=$'+str(Beta))
992
993
994
995
996
997
998 plt.xlim(0,1)
999 plt.ylim(0,1)
1000
1001 plt.show()
1002
1003 #C_2_8=sp.simplify(Lines[2]-Lines[8])
1004 #C_2_8_Num,C_2_8_Den=C_2_8.as_numer_denom()
1005
1006
1007 #C_16_8=sp.simplify(Lines[16]-Lines[8])
1008 #C_16_8_Num,C_16_8_Den=C_16_8.as_numer_denom()
1009
1010 #C_16_8_Num=sp.collect(C_16_8_Num,[alpha,beta,m,n])
1011 #C_16_8_Num_Hold=sp.collect(C_16_8_Num,[alpha,beta,m,n],evaluate=False)
1012
1013
1014
1015 #C_6_12=sp.simplify(Lines[6]-Lines[12])

```

```

1016 #C_6_12_Num , C_6_12_Den=C_6_12 . as_numer_denom ()
1017
1018 #C_6_17=sp . simplify ( Lines [6] - Lines [17])
1019 #C_6_17_Num , C_6_17_Den=C_6_17 . as_numer_denom ()
1020
1021
1022 #C_6_17_Num=sp . collect ( C_6_17_Num , [ alpha , beta , m , n])
1023 #C_6_17_Num_Hold=sp . collect ( C_6_17_Num , [ alpha , beta , m , n] , evaluate=False)
1024
1025 Cond1=sp . simplify (( L_Coef [8] [2] - L_Coef [2] [2] ) *p+(L_Coef [8] [3] - L_Coef
    [2] [3] ) *q+(L_Coef [8] [4] - L_Coef [2] [4] ) ) . subs ( { a2 : a2 / beta } )
1026 Cond2=sp . simplify (( L_Coef [12] [2] - L_Coef [6] [2] ) *p+(L_Coef [12] [3] - L_Coef
    [6] [3] ) *q+(L_Coef [12] [4] - L_Coef [6] [4] ) ) . subs ( { a2 : a2 / beta } )
1027
1028
1029 nC1 , dC1=Cond1 . as_numer_denom () ;
1030 nC2 , dC2=Cond2 . as_numer_denom () ;
1031
1032 nC1=sp . collect ( sp . expand ( nC1 / beta ) , { beta , q } )
1033 nC2=sp . collect ( sp . expand ( nC2 / beta ) , { beta , q } )
1034
1035
1036 CB1=sp . simplify ( sp . collect ( nC1 . coeff ( beta ) , p ) )
1037 CB2=sp . simplify ( sp . collect ( nC2 . coeff ( beta ) , p ) )
1038
1039 Cq1=sp . simplify ( sp . collect ( nC1 . coeff ( q ) , 1 ) )
1040 Cq2=sp . simplify ( sp . collect ( nC2 . coeff ( q ) , 1 ) )
1041
1042
1043
1044
1045
1046 n_13 , d_13=sp . solve ( sp . simplify (( L_Coef [8] [2] - L_Coef [2] [2] ) *p+(L_Coef
    [8] [3] - L_Coef [2] [3] ) *q+(L_Coef [8] [4] - L_Coef [2] [4] ) ) . subs ( { a2 : a2 /
    beta } ) , beta ) [0] . as_numer_denom ()
1047 n_24 , d_24=sp . solve ( sp . simplify (( L_Coef [12] [2] - L_Coef [6] [2] ) *p+(L_Coef
    [12] [3] - L_Coef [6] [3] ) *q+(L_Coef [12] [4] - L_Coef [6] [4] ) ) . subs ( { a2 : a2 /
    beta } ) , beta ) [0] . as_numer_denom ()
1048
1049
1050 n_13=sp . collect ( n_13 , q )
1051 d_13=sp . collect ( d_13 , p )
1052
1053 n_13=sp . collect ( n_13 , q )
1054 d_13=sp . collect ( d_13 , p )

```

## A.3 Clawpack Code

### A.3.1 Functions.f

```

2      double precision function p(xi , eta , xi1 , xi2 , eta1 , eta2 , y1 , y2)
3  C
4  ! This function evaluates the plane of slope mz and mt at point (xi , eta
5  )
6      implicit double precision (a-h,o-z)
7
8      zm=(y2 - y1)/(xi2 -xi1 )
9      tm=(y2 - y1)/(eta2-eta1)
10
11     p=y1+zm*(xi-xi1)+tm*(eta-eta1)
12     return
13     end function p
14
15 C
16 C
17 double precision function hl(xi , xi1 , xi2 , y1 , y2)
18 C
19 ! This function evaluates the line of slope m and y intercept (0,y1-m*
20 xi1) at point xi
21     implicit double precision (a-h,o-z)
22
23     zm=(y2 - y1)/(xi2 - xi1)
24
25     hl=y1+zm*(xi - xi1)
26     return
27     end function hl
28
29 C
30 C
31 double precision function py(x , z1 , z2 , t , t1 , t2 , u1 , u2)
32 C
33     implicit double precision (a-h,o-z)
34
35     zm=(z2-z1)/(t1-t2);
36
37     if ((z1 .le .x) .and. (x .lt .z2) .and.
38 &      ((x-z1) .lt .zm*(t-t2)) ) then
39         py1=p(x , t , z1 , z2 , t1 , t2 , u1 , u2);
40     else
41         py1=0.0d0;
42     endif
43
44     if ((z1 .le .x) .and.(x .lt .z2) .and.
45 &      ((x-z1) .ge .zm*(t-t2))) then
46         py2=p(x , t , z2 , z1 , t2 , t1 , u1 , u2)
47     else
48         py2=0.0d0;
49     endif
50
51     py=py1+py2
52     return
53     end function py

```

```

51 C
52 double precision function hm0(x, t, alp, bet, eps, tau, pm, qn, u1, u2)
53 C
54 implicit double precision (a-h, o-z)
55
56 if (x.lt.alp) then
57     f1=py(x,-alp,alp,t,-bet,bet,u1,u2)
58 else
59     f1=0.0d0
60 endif
61
62 if ((x.ge.alp).and.(x.lt.pm*eps-alp)) then
63     f2=h1(t,-bet,bet,u2,u1)
64 else
65     f2=0.0d0
66 endif
67
68 if ((x.ge.pm*eps-alp).and.(x.lt.pm*eps+alp)) then
69     f3=py(x,pm*eps-alp,pm*eps+alp,t,-bet,bet,u2,u1)
70 else
71     f3=0.0d0
72 endif
73
74 if ((pm*eps+alp.le.x).and.(x.lt.eps-alp)) then
75     f4=h1(t,-bet,bet,u1,u2)
76 else
77     f4=0.0d0
78 endif
79
80 if (x.gt.eps-alp) then
81     f5=py(x,eps-alp,eps+alp,t,-bet,bet,u1,u2)
82 else
83     f5=0.0d0
84 endif
85
86 hm0=f1+f2+f3+f4+f5
87 return
88 end function hm0
89
90 C
91 double precision function hm1(x, t, alp, bet, eps, tau, pm, qn, u1, u2)
92 C
93 implicit double precision (a-h, o-z)
94 z1=pm*eps-alp
95 z2=pm*eps+alp
96
97 if (x.lt.alp) then
98     f1=h1(x,-alp,alp,u2,u1)
99 else
100     f1=0.0d0
101 endif

```

```

102
103     if ((alp . le . x) . and . (x . lt . z1)) then
104         f2=u1
105     else
106         f2=0.0d0
107     endif
108
109     if ((z1 . le . x) . and . (x . lt . z2)) then
110         f3=h1(x , z1 , z2 , u1 , u2)
111     else
112         f3=0.0d0
113     endif
114
115     if ((z2 . le . x) . and . (x . lt . eps-alp)) then
116         f4=u2
117     else
118         f4=0.0d0
119     endif
120
121     if (eps-alp . le . x) then
122         f5=h1(x , eps-alp , eps+alp , u2 , u1)
123     else
124         f5=0.0d0
125     endif
126
127     hm1=f1+f2+f3+f4+f5
128     return
129     end function hm1
130
131 C
132     =====
133     double precision function hm2(x , t , alp , bet , eps , tau , pm , qn , u1 , u2)
134 C
135     =====
136     implicit double precision (a-h , o-z)
137
138     t1=qn*tau-bet
139     t2=qn*tau+bet
140     z1=pm*eps-alp
141     z2=pm*eps+alp
142
143     zm=(z2-z1)/(t1-t2)
144
145
146     if (x . lt . alp) then
147         f1=py(x , -alp , alp , t , qn*tau-bet , qn*tau+bet , u2 , u1)
148     else
149         f1=0.0d0
150     endif
151
152     if ((alp . le . x) . and . (x . lt . z1)) then

```

```

153         f2=h1(t,t1,t2,u1,u2)
154     else
155         f2=0.0d0
156     endif
157
158     if ((z1.le.x) .and. (x.lt.z2) .and.
159 & ((x-z1).lt.zm*(t-t2)) ) then
160         f3=p(x,t,z1,z2,t1,t2,u1,u2)
161     else
162         f3=0.0d0
163     endif
164
165     if ((z1.le.x) .and. (x.lt.z2) .and.
166 & ((x-z1).ge.zm*(t-t2)) ) then
167         f4=p(x,t,z2,z1,t2,t1,u1,u2)
168     else
169         f4=0.0d0
170     endif
171
172     if ((z2.le.x) .and. (x.lt.eps-alp)) then
173         f5=h1(t,t1,t2,u2,u1)
174     else
175         f5=0.0d0;
176     endif
177
178     if (eps-alp.le.x) then
179         f6=py(x,eps-alp,eps+alp,t,qn*tau-bet,qn*tau+bet,u2,u1)
180     else
181         f6=0.0d0
182     endif
183
184     hm2=f1+f2+f3+f4+f5+f6
185     return
186
187     end function hm2
188
189 C =====
190 double precision function hm3(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
191 C =====
192 implicit double precision (a-h,o-z)
193 z1=pm*eps-alp
194 z2=pm*eps+alp
195
196 hm3=hm1(x,t,alp,bet,eps,tau,pm,qn,u2,u1)
197 return
198 end function hm3
199
200 C =====
201 double precision function hm4(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
202 C =====
203 implicit double precision (a-h,o-z)

```

```

204
205     if (x.lt.alp) then
206         f1=py(x,-alp,alp,t,tau-bet,tau+bet,u1,u2)
207     else
208         f1=0.0d0
209     endif
210
211     if ((alp.le.x).and.(x.lt.pm*eps-alp)) then
212         f2=h1(t,tau-bet,tau+bet,u2,u1)
213     else
214         f2=0.0d0
215     endif
216
217     if ((pm*eps-alp.le.x).and.(x.lt.pm*eps+alp)) then
218         f3=py(x,pm*eps-alp,pm*eps+alp,t,tau-bet,tau+bet,u2,u1)
219     else
220         f3=0.0d0
221     endif
222
223     if ((pm*eps+alp.le.x).and.(x.lt.eps-alp)) then
224         f4=h1(t,tau-bet,tau+bet,u1,u2)
225     else
226         f4=0.0d0
227     endif
228
229     if (eps-alp.le.x) then
230         f5=py(x,eps-alp,eps+alp,t,tau-bet,tau+bet,u1,u2)
231     else
232         f5=0.0d0
233     endif
234
235     hm4=f1+f2+f3+f4+f5
236     return
237     end function hm4
238
239 C =====
240 double precision function fl_CB(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
241 C =====
242 implicit double precision (a-h,o-z)
243
244 t1=qn*tau-bet
245 t2=qn*tau+bet
246
247 if (t.lt.bet) then
248     f1=hm0(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
249 else
250     f1=0.0d0
251 endif
252
253 if ((bet.le.t).and.(t.lt.t1)) then
254     f2=hml(x,t,alp,bet,eps,tau,pm,qn,u1,u2)

```



```

255     else
256         f2=0.0d0
257     endif
258
259     if ((t1.le.t).and.(t.lt.t2)) then
260         f3=hm2(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
261     else
262         f3=0.0d0
263     endif
264
265     if ((t2.le.t).and.(t.lt.tau-bet)) then
266         f4=hm3(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
267     else
268         f4=0.0d0
269     endif
270
271     if (tau-bet.le.t) then
272         f5=hm4(x,t,alp,bet,eps,tau,pm,qn,u1,u2)
273     else
274         f5=0.0d0
275     endif
276
277     fl_CB=f1+f2+f3+f4+f5
278     return
279     end function fl_CB
280
281 C
282     double precision function Sh_CB(x,y,t,u1,u2)
283 C
284     implicit double precision (a-h,o-z)
285
286     common /cparam/ gamma1,v1,gamma2,v2
287     common /cboard/ pm,qn,eps,tau,alp,bet
288
289
290
291     !!!     x=mod(real(x),eps)
292     !!!     t=mod(real(t),tau)
293
294
295     if (t.ge.0.0d0) then
296         if ((x.lt.pm*eps).and.(t.lt.qn*tau)) then
297             Sh_CB=u1
298         elseif ((x.ge.pm*eps).and.(t.lt.qn*tau)) then
299             Sh_CB=u2
300         elseif ((x.lt.pm*eps).and.(t.ge.qn*tau)) then
301             Sh_CB=u2
302         else
303             Sh_CB=u1
304         endif
305     else

```

```

306         Sh_CB=u1
307     endif
308
309     return
310 end function Sh_CB
311
312
313
314
315 C
316 double precision function f_u(x,y,t,u1,u2)
317 C
318 implicit double precision (a-h,o-z)
319
320 common /cparam/ gamma1,v1,gamma2,v2
321 common /cboard/ pm,qn,eps,tau,alp,bet
322
323 xM=mod(real(x),eps)
324 tM=mod(real(t),tau)
325
326 f_u=Sh_CB(xM,y,tM,u1,u2)
327 ! f_u=fl_CB(xM,tM,alp,bet,eps,tau,pm,qn,u1,u2)
328
329 return
330 end function f_u
331
332
333
334
335 !!!!!      subroutine Velocity(X,Y,V,t,Mx,My)
336 !!!!!      implicit double precision (a-h,o-z)
337 !!!!!      INTEGER Mx,My
338 !!!!!      REAL*8 t
339 !!!!!      REAL*8 X(Mx,My),Y(Mx,My)
340 !!!!!      REAL*8 V(Mx,My)
341 !!!!!
342 !!!!!      integer i,j
343
344 !!!!!
345 !!!!! cf2py intent(in) Mx,My
346 !!!!! cf2py intent(in) t
347 !!!!! cf2py intent(in) X(Mx,My),Y(Mx,My)
348 !!!!! cf2py intent(inout) V(Mx,My)
349
350
351 !!!!!      common /cparam/ rho1,bulk1,gamma1,v1,rho2,bulk2,gamma2,v2
352 !!!!!      common /cboard/ pm,qn,eps,tau,alp,bet
353 !!!!!
354 !!!!!      gamma=1.0d0;
355 !!!!!      gamma1=gamma;
356 !!!!!      gamma2=gamma;

```

```

357
358 !!!!!      v1=1.10d0;v2=1.10d0;
359
360 !!!!!      rho1=gamma1/v1;rho2=gamma2/v2
361
362 !!!!!      bulk1=gamma1*v1;bulk2=gamma2*v2
363
364 !!!!!      eps=.50d0;tau=.50d0;pm=.50d0;qn=.50d0
365
366 !!!!!      alp=.050d0;bet=.050d0;
367 !!!!!
368
369 !!!!!      do 10 i = 1, Mx
370 !!!!!          do 20 j = 1, My
371 !!!!!              V(i,j)=f_u(X(i,j),Y(i,j),t,v1,v2)
372 !!!!!          20   enddo
373 !!!!!      10   enddo
374 !!!!!
375
376 !!!!!      return
377 !!!!!      end

```

### A.3.2 setrun.py

```

1 """
2 Module to set up run time parameters for Clawpack.
3
4 The values set in the function setrun are then written out to data
5 files
6 that will be read in by the Fortran code.
7 """
8
9 import os
10 import numpy as np
11
12
13 gamma1=1.0;
14 gamma2=gamma1;
15
16 v1=1.1;
17 v2=0.55;
18
19
20 bulk1_A=gamma1*v1 #This is our (Lurie,Weekes) k1
21 bulk2_A=gamma2*v2 #This is our (Lurie,Weekes) k2
22
23 rho1_A=gamma1/v1 #This is our (Lurie,Weekes) rho2
24 rho2_A=gamma2/v2 #This is our (Lurie,Weekes) rho2
25
26
27

```

```

28
29 #bulk1=bulk1_A
30 #bulk2=bulk2_A
31
32 #rho1=rho1_A
33 #rho2=rho2_A
34
35 bulk1=1/rho1_A #This sets correct dependence of clawpack k1 on our rho1
36 bulk2=1/rho2_A #This sets correct dependence of clawpack k2 on our rho2
37 rho1=1/bulk1_A #This sets correct dependence of clawpack rho1 on our k1
38 rho2=1/bulk2_A #This sets correct dependence of clawpack rho1 on our k2
39
40
41
42 eps = 1.0; tau = 1.0; pm = 0.5; qn = 0.5;
43
44
45 Pe = min([pm*eps, (1-pm)*eps, qn*tau, (1-qn)*tau]) / 2
46
47 alp = (.001) * Pe;
48 bet = (.001) * Pe;
49
50
51 t_0 = 0.0; t_F = 4 * tau;
52
53
54 #-----
55 def setrun(claw_pkg='amrclaw'):
56 #-----
57
58     """
59     Define the parameters used for running Clawpack.
60
61     INPUT:
62         claw_pkg expected to be "amrclaw" for this setrun.
63
64     OUTPUT:
65         rundata - object of class ClawRunData
66
67     """
68
69     from clawpack.clawutil import data
70
71
72     assert claw_pkg.lower() == 'amrclaw', "Expected claw_pkg = '
73     amrclaw'"
74
75     num_dim = 2
76     rundata = data.ClawRunData(claw_pkg, num_dim)
77
78     #-----

```

```

78 # Problem-specific parameters to be written to setprob.data:
79 #-----
80
81 probdata = rundata.new_UserData(name='probdata',fname='setprob.data
82 ')
83
84 probdata.add_param('rho1', rho1, 'density of medium1')
85 probdata.add_param('bulk1', bulk1, 'bulk modulus of medium1')
86 probdata.add_param('gammal', gammal, 'impedance of medium 1')
87 probdata.add_param('v1', v1, 'wave speed of medium1')
88
89 probdata.add_param('rho2', rho2, 'density of medium')
90 probdata.add_param('bulk2', bulk2, 'bulk modulus')
91 probdata.add_param('gamma2', gamma2, 'impedance of medium 2')
92 probdata.add_param('v2', v2, 'wave speed of medium2')
93
94 probdata.add_param('pm', pm, 'spatial volume fraction')
95 probdata.add_param('qn', qn, 'temporal volume fraction')
96 probdata.add_param('eps', eps, 'spatial period')
97 probdata.add_param('tau', tau, 'temporal period')
98
99 probdata.add_param('alp', alp, 'spatial smoothing')
100 probdata.add_param('bet', bet, 'temporal smoothing')
101
102 #-----
103 # Standard Clawpack parameters to be written to claw.data:
104 # (or to amrclaw.data for AMR)
105 #-----
106
107 clawdata = rundata.clawdata # initialized when rundata
108 instantiated
109
110 # Set single grid parameters first.
111 # See below for AMR parameters.
112
113
114 #-----
115 # Spatial domain:
116 #-----
117
118 # Number of space dimensions:
119 clawdata.num_dim = num_dim
120
121 # Lower and upper edge of computational domain:
122 clawdata.lower[0] = 0.000000e+00 # xlower
123 clawdata.upper[0] = 15.000000e+00 # xupper
124 clawdata.lower[1] = 0.000000e+00 # ylower
125 clawdata.upper[1] = 1.000000e+00 # yupper
126

```

```

127 # Number of grid cells:
128 clawdata.num_cells[0] = 5000      # mx
129 clawdata.num_cells[1] = 4        # my
130
131
132 # -----
133 # Size of system:
134 # -----
135
136 # Number of equations in the system:
137 clawdata.num_eqn = 3
138
139 # Number of auxiliary variables in the aux array (initialized in
140 setaux)
141 clawdata.num_aux = 2
142
143 # Index of aux array corresponding to capacity function, if there
144 is one:
145 clawdata.capa_index = 0
146
147 # -----
148 # Initial time:
149 # -----
150
151 clawdata.t0 = t_0
152
153 # Restart from checkpoint file of a previous run?
154 # Note: If restarting, you must also change the Makefile to set:
155 #   RESTART = True
156 # If restarting, t0 above should be from original run, and the
157 # restart_file 'fort.chkNNNNN' specified below should be in
158 # the OUTDIR indicated in Makefile.
159
160 clawdata.restart = False           # True to restart from prior
161 results                             # File to use for restart
162 clawdata.restart_file = 'fort.chk00006'
163
164 # -----
165 # Output times:
166 # -----
167
168 # Specify at what times the results should be written to fort.q
169 files.
170 # Note that the time integration stops after the final output time.
171
172 clawdata.output_style = 2

```

```

173     if clawdata.output_style==1:
174         # Output ntimes frames at equally spaced times up to tfinal:
175         # Can specify num_output_times = 0 for no output
176         N_t=50;
177         clawdata.num_output_times = N_t
178         clawdata.tfinal = t_F
179         clawdata.output_t0 = True # output at initial (or restart)
time?
180
181     elif clawdata.output_style == 2:
182         Nt=200
183         clawdata.output_times=np.linspace(t_0,t_F,Nt)
184
185     elif clawdata.output_style == 3:
186         # Output every step_interval timesteps over total_steps
timesteps:
187         clawdata.output_step_interval = 2
188         clawdata.total_steps = 4
189         clawdata.output_t0 = True # output at initial (or restart)
time?
190
191
192     clawdata.output_format = 'ascii'      # 'ascii', 'binary', 'netcdf
,
193
194     clawdata.output_q_components = 'all'  # could be list such as [
True,True]
195     clawdata.output_aux_components = 'all' # could be list
196     clawdata.output_aux_onlyonce = False  # output aux arrays only at
t0
197
198
199     # -----
200     # Verbosity of messages to screen during integration:
201     # -----
202
203     # The current t, dt, and cfl will be printed every time step
204     # at AMR levels <= verbosity. Set verbosity = 0 for no printing.
205     # (E.g. verbosity == 2 means print only on levels 1 and 2.)
206     clawdata.verbosity = 2
207
208
209
210     # -----
211     # Time stepping:
212     # -----
213
214     # if dt_variable==True: variable time steps used based on
cfl_desired ,
215     # if dt_variable==False: fixed time steps dt = dt_initial always
used.

```

```

216  clawdata.dt_variable = True
217
218  # Initial time step for variable dt.
219  # (If dt_variable==0 then dt=dt_initial for all steps)
220  clawdata.dt_initial = 2.00000e-06
221
222  # Max time step to be allowed if variable dt used:
223  clawdata.dt_max = 1.000000e+99
224
225  # Desired Courant number if variable dt used
226  clawdata.cfl_desired = 0.95
227  # max Courant number to allow without retaking step with a smaller
228  dt:
229  clawdata.cfl_max = 1.000000
230
231  # Maximum number of time steps to allow between output times:
232  clawdata.steps_max = 100000 #Original value 50000
233
234  # -----
235  # Method to be used:
236  # -----
237
238  # Order of accuracy:  1 => Godunov,  2 => Lax-Wendroff plus
239  limiters
240  clawdata.order = 2
241
242  # Use dimensional splitting?
243  clawdata.dimensional_split = 'unsplit'
244
245  # For unsplit method, transverse_waves can be
246  # 0 or 'none'      => donor cell (only normal solver used)
247  # 1 or 'increment' => corner transport of waves
248  # 2 or 'all'      => corner transport of 2nd order corrections
249  too
250  clawdata.transverse_waves = 2
251
252  # Number of waves in the Riemann solution:
253  clawdata.num_waves = 2
254
255  # List of limiters to use for each wave family:
256  # Required: len(limiter) == num_waves
257  # Some options:
258  # 0 or 'none'      => no limiter (Lax-Wendroff)
259  # 1 or 'minmod'    => minmod
260  # 2 or 'superbee' => superbee
261  # 3 or 'vanleer'  => van Leer
262  # 4 or 'mc'       => MC limiter
263  # clawdata.limiter = [0,0]
264  clawdata.limiter = ['minmod', 'minmod']

```



```

264
265
266
267   clawdata.use_fwaves = False    # True ==> use f-wave version of
algorithms
268
269   # Source terms splitting:
270   #   src_split == 0 or 'none'    ==> no source term (src routine
never called)
271   #   src_split == 1 or 'godunov' ==> Godunov (1st order) splitting
used,
272   #   src_split == 2 or 'strang'  ==> Strang (2nd order) splitting
used, not recommended.
273   clawdata.source_split = 0
274
275
276   # -----
277   # Boundary conditions:
278   # -----
279
280   # Number of ghost cells (usually 2)
281   clawdata.num_ghost = 2
282
283   # Choice of BCs at xlower and xupper:
284   #   0 or 'user'    => user specified (must modify bcNamr.f to use
this option)
285   #   1 or 'extrap'  => extrapolation (non-reflecting outflow)
286   #   2 or 'periodic' => periodic (must specify this at both
boundaries)
287   #   3 or 'wall'    => solid wall for systems where q(2) is normal
velocity
288
289   clawdata.bc_lower[0] = 'periodic' # at xlower
290   clawdata.bc_upper[0] = 'periodic' # at xupper
291
292   clawdata.bc_lower[1] = 'extrap'   # at ylower
293   clawdata.bc_upper[1] = 'extrap'   # at yupper
294
295
296   # -----
297   # Gauges:
298   # -----
299   rundata.gaugedata.gauges = []
300   # for gauges append lines of the form [gaugeno, x, y, t1, t2]
301   rundata.gaugedata.gauges.append([0, 0.0, 0.0, 0., 10.])
302   rundata.gaugedata.gauges.append([1, 0.7, 0.0, 0., 10.])
303   rundata.gaugedata.gauges.append([2, 0.7/np.sqrt(2.), 0.7/np.sqrt
(2.), 0., 10.])
304
305   # -----
306   # Checkpointing:

```

```

307 # -----
308
309 # Specify when checkpoint files should be created that can be
310 # used to restart a computation.
311
312 clawdata.checkpt_style = 0
313
314 if clawdata.checkpt_style == 0:
315     # Do not checkpoint at all
316     pass
317
318 elif clawdata.checkpt_style == 1:
319     # Checkpoint only at tfinal.
320     pass
321
322 elif clawdata.checkpt_style == 2:
323     # Specify a list of checkpoint times.
324     clawdata.checkpt_times = [0.1,0.15]
325
326 elif clawdata.checkpt_style == 3:
327     # Checkpoint every checkpt_interval timesteps (on Level 1)
328     # and at the final time.
329     clawdata.checkpt_interval = 5
330
331
332
333 # -----
334 # AMR parameters:
335 # -----
336
337 amrdata = rundata.amrdata
338
339 # max number of refinement levels:
340 amrdata.amr_levels_max = 3
341
342 # List of refinement ratios at each level (length at least
343 # amr_level_max-1)
344 amrdata.refinement_ratios_x = [2,2**4]
345 amrdata.refinement_ratios_y = [1,1]
346 amrdata.refinement_ratios_t = [2,2**4]
347
348 # Specify type of each aux variable in clawdata.auxtype.
349 # This must be a list of length num_aux, each element of which is
350 # one of:
351 # 'center', 'capacity', 'xleft', or 'yleft' (see documentation)
352 .
353 amrdata.aux_type = ['center', 'center']
354
355 # Flag for refinement based on Richardson error estimator:

```

```

355 amrdata.flag_richardson = False # use Richardson?
356 amrdata.flag_richardson_tol = 0.001000e+00 # Richardson tolerance
357
358 # Flag for refinement using routine flag2refine:
359 amrdata.flag2refine = True # use this?
360 amrdata.flag2refine_tol = 1e-4 # tolerance used in this routine
361 # User can modify flag2refine to change the criterion for flagging.
362 # Default: check maximum absolute difference of first component of
363 # q
364 # between a cell and each of its neighbors.
365
366 # steps to take on each level L between regriddings of level L+1:
367 amrdata.regrid_interval = 2
368
369 # width of buffer zone around flagged points:
370 # (typically the same as regrid_interval so waves don't escape):
371 amrdata.regrid_buffer_width = 2
372
373 # clustering alg. cutoff for (# flagged pts) / (total # of cells
374 # refined)
375 # (closer to 1.0 => more small grids may be needed to cover flagged
376 # cells)
377 amrdata.clustering_cutoff = 0.7
378
379 # print info about each regridding up to this level:
380 amrdata.verbosity_regrid = 0
381
382 # -----
383 # Regions:
384 # -----
385 rundata.regiondata.regions = []
386 # to specify regions of refinement append lines of the form
387 # [minlevel, maxlevel, t1, t2, x1, x2, y1, y2]
388
389 # ----- For developers -----
390 # Toggle debugging print statements:
391 amrdata.dprint = False # print domain flags
392 amrdata.eprint = False # print err est flags
393 amrdata.edebug = False # even more err est flags
394 amrdata.gprint = False # grid bisection/clustering
395 amrdata.nprint = False # proper nesting output
396 amrdata.pprint = False # proj. of tagged points
397 amrdata.rprint = False # print regridding summary
398 amrdata.sprint = False # space/memory output
399 amrdata.tprint = False # time step reporting each level
400 amrdata.uprint = False # update/upbnd reporting
401
402 return rundata

```

```

403 # end of function setrun
404 # -----
405
406
407 if __name__ == '__main__':
408     # Set up run-time parameters and write all data files.
409     import sys
410     rundata = setrun(*sys.argv[1:])
411     rundata.write()

```

### A.3.3 Main Script

```

1 #!/bin/bash
2
3
4
5 export Ntot=$(( $1 * $1 ))
6
7 printf -v pad_n "%03d" $1
8
9
10 export StorDir="Test"$pad_n"b"$pad_n
11
12 if [ -d "$StorDir" ]; then
13     rm -r $StorDir
14     mkdir $StorDir
15 else
16     mkdir $StorDir
17 fi
18
19
20
21
22 export MaxSize=$1
23
24 for i in $(seq 1 $Ntot);
25 do
26
27 #   echo $i[';p
28
29 #   if [ -d "output$i" ]; then
30
31 #       echo "WARNING output directories already exists you might"
32 #       echo "be rewriting your results over"
33 #       echo "you have 10 seconds to stop me"#       sleep 10
34 #   fi
35
36 #mkdir $StorDir/input$i
37 printf -v pad_i "%05d" $i
38
39     mkdir $StorDir/output$i
40 # mkdir $StorDir/output$(printf %04d $i)

```

```

41     wait
42     let ahah=$((i-1))
43     export SGE_TASK_ID=$ahah
44     wait
45     python setrun.py
46     wait
47 #     cp *.data $StorDir/output$(printf %04d $i)/.
48     cp *.data $StorDir/output$i/
49 done
50
51 echo "Samples are on an $1 by $1 grid"
52 echo "So the total Number of Samples is $Ntot"
53
54
55
56 qsub -t 1-$Ntot SGEscript_F.sh $1 $StorDir/output

```

### A.3.4 SGE Script

```

1 #!/bin/bash
2 ## -N test
3 ## -cwd
4 ## -V
5 ## -M wcsanguinet@wpi.edu
6 ##### -m abe
7 ##### -pe orte 1
8
9 ## -pe omp 10
10
11 ## -q math.q
12 ##### -q all.q
13
14 ##### -t 1-$Ntot
15
16 ## -o outlog/
17 ## -e outlog/
18
19 ##### THE JOB ITSELF #####
20 #
21
22 #setenv CLAW /home/wcsanguinet/clawpack-5.2.2
23
24 setenv OMP_NUM_THREADS 10
25
26 #cd /home/wcsanguinet/clawpack-5.2.2/amrclaw/examples/TemporaryResearch
27 #setenv SPECIAL_PATH Test$1b$1/output$SGE_TASK_ID
28
29 printf -v pad_sg "%05d" $SGE_TASK_ID
30
31 python $CLAW/clawutil/src/python/clawutil/runclaw.py xamr ./
32     $2$SGE_TASK_ID True False ./ $2$SGE_TASK_ID

```

```

32
33 ##python setrun.py
34
35 #make .output
36
37 #echo $HOSTNAME
38 #echo $OMP_NUM_THREADS
39 #echo $CLAW

```

### A.3.5 setrun.py

```

1 """
2 Module to set up run time parameters for Clawpack.
3
4 The values set in the function setrun are then written out to data
5 files
6 that will be read in by the Fortran code.
7 """
8
9 import os
10 import numpy as np
11
12 import sys
13
14
15
16 gamma1=1.0;
17 gamma2=gamma1;
18
19 v1=1.1;
20 v2=0.55;
21
22
23 bulk1_A=gamma1*v1 #This is our (Lurie,Weekes) k1
24 bulk2_A=gamma2*v2 #This is our (Lurie,Weekes) k2
25
26 rho1_A=gamma1/v1 #This is our (Lurie,Weekes) rho2
27 rho2_A=gamma2/v2 #This is our (Lurie,Weekes) rho2
28
29
30
31
32 #bulk1=bulk1_A
33 #bulk2=bulk2_A
34
35 #rho1=rho1_A
36 #rho2=rho2_A
37
38 bulk1=1/rho1_A #This sets correct dependence of clawpack k1 on our rho1
39 bulk2=1/rho2_A #This sets correct dependence of clawpack k2 on our rho2
40 rho1=1/bulk1_A #This sets correct dependence of clawpack rho1 on our k1

```

```

41 rho2=1/bulk2_A #This sets correct dependence of clawpack rho1 on our k2
42
43
44
45 eps = 1.0; tau = 1.0; pm = 0.5; qn = 0.5;
46
47
48 Pe = min ([pm*eps ,(1 - pm) * eps , qn * tau ,(1 - qn) * tau ]) / 2
49
50 alp = .0001;
51 bet = .0001;
52
53
54 t_0 = 0.0; t_F = 4 * tau ;
55
56
57 #-----
58 def setrun ( claw_pkg = 'amrclaw ' ) :
59 #-----
60     ##### THIS NEEDS TO
61     #####
62     MaxSize = int ( os . environ [ " MaxSize " ] )
63
64
65     m_Min = 0.0; m_Max = 1.0;
66     n_Min = 0.0; n_Max = 1.0;
67
68     Nt = np . linspace ( m_Min , m_Max , MaxSize )
69     Mt = np . linspace ( n_Min , n_Max , MaxSize )
70
71     NNT = [[ Nt [ i ] for i in range ( MaxSize ) ] for j in range ( MaxSize ) ]
72     MMT = [[ Mt [ j ] for i in range ( MaxSize ) ] for j in range ( MaxSize ) ]
73
74     MMT2 = np . array ( MMT ) . reshape ( -1 )
75     NNT2 = np . array ( NNT ) . reshape ( -1 )
76
77     ## print ( MMT2 , ' , MMT2 ' )
78     ## print ( NNT2 , ' , NNT2 ' )
79
80     ## print ( MMT2 . size , ' , MMT2 . size ' )
81     ## print ( NNT2 . size , ' , NNT2 . size ' )
82     sge = int ( os . environ [ " SGE_TASK_ID " ] )
83
84     # print ( sge , ' , SGE_TASK_ID ' )
85     # print ( MMT2 [ sge ] , ' , MMT2 [ sge ] ' )
86     # print ( NNT2 [ sge ] , ' , NNT2 [ sge ] ' )
87
88     # print ( sge )
89
90     # sys . exit ( )

```

```

91
92
93
94 MM=MMT2[ sge ]
95 NN=NNT2[ sge ]
96
97
98 #
#####

99 """
100 Define the parameters used for running Clawpack.
101
102 INPUT:
103     claw_pkg expected to be "amrclaw" for this setrun.
104
105 OUTPUT:
106     rundata - object of class ClawRunData
107
108 """
109
110 from clawpack.clawutil import data
111
112
113 assert claw_pkg.lower() == 'amrclaw', "Expected claw_pkg = '
amrclaw'"
114
115 num_dim = 2
116 rundata = data.ClawRunData(claw_pkg, num_dim)
117
118 #-----
119 # Problem-specific parameters to be written to setprob.data:
120 #-----
121
122 probdata = rundata.new_UserData(name='probdata', fname='setprob.data
')
123
124 probdata.add_param('rho1', rho1, 'density of medium1')
125 probdata.add_param('bulk1', bulk1, 'bulk modulus of medium1')
126 probdata.add_param('gammal', gammal, 'impedance of medium 1')
127 probdata.add_param('v1', v1, 'wave speed of medium1')
128
129 probdata.add_param('rho2', rho2, 'density of medium')
130 probdata.add_param('bulk2', bulk2, 'bulk modulus')
131 probdata.add_param('gamma2', gamma2, 'impedance of medium 2')
132 probdata.add_param('v2', v2, 'wave speed of medium2')
133
134 probdata.add_param('pm', MM, 'spatial volume fraction')
135 probdata.add_param('qn', NN, 'temporal volume fraction')
136 probdata.add_param('eps', eps, 'spatial period')
137 probdata.add_param('tau', tau, 'temporal period')

```



```

138
139     probdata.add_param('alp', alp, 'spatial smoothing')
140     probdata.add_param('bet', bet, 'temporal smoothing')
141
142
143     # -----
144     # Standard Clawpack parameters to be written to claw.data:
145     # (or to amrclaw.data for AMR)
146     # -----
147
148     clawdata = rundata.clawdata # initialized when rundata
instantiated
149
150
151     # Set single grid parameters first.
152     # See below for AMR parameters.
153
154
155     # -----
156     # Spatial domain:
157     # -----
158
159     # Number of space dimensions:
160     clawdata.num_dim = num_dim
161
162     # Lower and upper edge of computational domain:
163     clawdata.lower[0] = 0.000000e+00 # xlower
164     clawdata.upper[0] = 15.000000e+00 # xupper
165     clawdata.lower[1] = 0.000000e+00 # ylower
166     clawdata.upper[1] = 1.000000e+00 # yupper
167
168     # Number of grid cells:
169     clawdata.num_cells[0] = 5000 # mx
170     clawdata.num_cells[1] = 4 # my
171
172
173     # -----
174     # Size of system:
175     # -----
176
177     # Number of equations in the system:
178     clawdata.num_eqn = 3
179
180     # Number of auxiliary variables in the aux array (initialized in
setaux)
181     clawdata.num_aux = 2
182
183     # Index of aux array corresponding to capacity function, if there
is one:
184     clawdata.capacity_index = 0
185

```

```

186
187 # -----
188 # Initial time:
189 # -----
190
191 clawdata.t0 =t_0
192
193
194 # Restart from checkpoint file of a previous run?
195 # Note: If restarting, you must also change the Makefile to set:
196 #   RESTART = True
197 # If restarting, t0 above should be from original run, and the
198 # restart_file 'fort.chkNNNNN' specified below should be in
199 # the OUTDIR indicated in Makefile.
200
201 clawdata.restart = False           # True to restart from prior
   results
202 clawdata.restart_file = 'fort.chk00006' # File to use for restart
   data
203
204
205 # -----
206 # Output times:
207 # -----
208
209 # Specify at what times the results should be written to fort.q
   files.
210 # Note that the time integration stops after the final output time.
211
212 clawdata.output_style = 2
213
214 if clawdata.output_style==1:
215     # Output ntimes frames at equally spaced times up to tfinal:
216     # Can specify num_output_times = 0 for no output
217     N_t=50;
218     clawdata.num_output_times = N_t
219     clawdata.tfinal = t_F
220     clawdata.output_t0 = True # output at initial (or restart)
   time?
221
222 elif clawdata.output_style == 2:
223     Nt=200
224     clawdata.output_times=np.linspace(t_0,t_F,Nt)
225
226 elif clawdata.output_style == 3:
227     # Output every step_interval timesteps over total_steps
   timesteps:
228     clawdata.output_step_interval = 2
229     clawdata.total_steps = 4
230     clawdata.output_t0 = True # output at initial (or restart)
   time?

```

```

231
232
233     clawdata.output_format = 'ascii'          # 'ascii', 'binary', 'netcdf
,
234
235     clawdata.output_q_components = 'all'      # could be list such as [
True,True]
236     clawdata.output_aux_components = 'all'    # could be list
237     clawdata.output_aux_onlyonce = False     # output aux arrays only at
t0
238
239
240     # -----
241     # Verbosity of messages to screen during integration:
242     # -----
243
244     # The current t, dt, and cfl will be printed every time step
245     # at AMR levels <= verbosity. Set verbosity = 0 for no printing.
246     # (E.g. verbosity == 2 means print only on levels 1 and 2.)
247     clawdata.verbosity = 2
248
249
250
251     # -----
252     # Time stepping:
253     # -----
254
255     # if dt_variable==True: variable time steps used based on
cfl_desired,
256     # if dt_variable==False: fixed time steps dt = dt_initial always
used.
257     clawdata.dt_variable = True
258
259     # Initial time step for variable dt.
260     # (If dt_variable==0 then dt=dt_initial for all steps)
261     clawdata.dt_initial = 2.00000e-06
262
263     # Max time step to be allowed if variable dt used:
264     clawdata.dt_max = 1.000000e+99
265
266     # Desired Courant number if variable dt used
267     clawdata.cfl_desired = 0.95
268     # max Courant number to allow without retaking step with a smaller
dt:
269     clawdata.cfl_max = 1.000000
270
271     # Maximum number of time steps to allow between output times:
272     clawdata.steps_max = 100000 #Original value 50000
273
274
275     # -----

```

```

276 # Method to be used:
277 # _____
278
279 # Order of accuracy: 1 => Godunov, 2 => Lax-Wendroff plus
280 # limiters
281 clawdata.order = 2
282
283 # Use dimensional splitting?
284 clawdata.dimensional_split = 'unsplit'
285
286 # For unsplit method, transverse_waves can be
287 # 0 or 'none' ==> donor cell (only normal solver used)
288 # 1 or 'increment' ==> corner transport of waves
289 # 2 or 'all' ==> corner transport of 2nd order corrections
290 # too
291 clawdata.transverse_waves = 2
292
293 # Number of waves in the Riemann solution:
294 clawdata.num_waves = 2
295
296 # List of limiters to use for each wave family:
297 # Required: len(limiter) == num_waves
298 # Some options:
299 # 0 or 'none' ==> no limiter (Lax-Wendroff)
300 # 1 or 'minmod' ==> minmod
301 # 2 or 'superbee' ==> superbee
302 # 3 or 'vanleer' ==> van Leer
303 # 4 or 'mc' ==> MC limiter
304 # clawdata.limiter = [0,0]
305 clawdata.limiter = ['minmod', 'minmod']
306
307
308 clawdata.use_fwaves = False # True ==> use f-wave version of
309 # algorithms
310
311 # Source terms splitting:
312 # src_split == 0 or 'none' ==> no source term (src routine
313 # never called)
314 # src_split == 1 or 'godunov' ==> Godunov (1st order) splitting
315 # used,
316 # src_split == 2 or 'strang' ==> Strang (2nd order) splitting
317 # used, not recommended.
318 clawdata.source_split = 0
319
320 # _____
321 # Boundary conditions:
322 # _____

```

```

321 # Number of ghost cells (usually 2)
322 clawdata.num_ghost = 2
323
324 # Choice of BCs at xlower and xupper:
325 # 0 or 'user'      => user specified (must modify bcNamr.f to use
this option)
326 # 1 or 'extrap'   => extrapolation (non-reflecting outflow)
327 # 2 or 'periodic' => periodic (must specify this at both
boundaries)
328 # 3 or 'wall'    => solid wall for systems where q(2) is normal
velocity
329
330 clawdata.bc_lower[0] = 'periodic' # at xlower
331 clawdata.bc_upper[0] = 'periodic' # at xupper
332
333 clawdata.bc_lower[1] = 'extrap' # at ylower
334 clawdata.bc_upper[1] = 'extrap' # at yupper
335
336
337 # -----
338 # Gauges:
339 # -----
340 rundata.gaugedata.gauges = []
341 # for gauges append lines of the form [gaugen0, x, y, t1, t2]
342 rundata.gaugedata.gauges.append([0, 0.0, 0.0, 0., 10.])
343 rundata.gaugedata.gauges.append([1, 0.7, 0.0, 0., 10.])
344 rundata.gaugedata.gauges.append([2, 0.7/np.sqrt(2.), 0.7/np.sqrt
(2.), 0., 10.])
345
346 # -----
347 # Checkpointing:
348 # -----
349
350 # Specify when checkpoint files should be created that can be
351 # used to restart a computation.
352
353 clawdata.checkpt_style = 0
354
355 if clawdata.checkpt_style == 0:
356     # Do not checkpoint at all
357     pass
358
359 elif clawdata.checkpt_style == 1:
360     # Checkpoint only at tfinal.
361     pass
362
363 elif clawdata.checkpt_style == 2:
364     # Specify a list of checkpoint times.
365     clawdata.checkpt_times = [0.1,0.15]
366
367 elif clawdata.checkpt_style == 3:

```

```

368     # Checkpoint every checkpt_interval timesteps (on Level 1)
369     # and at the final time.
370     clawdata.checkpt_interval = 5
371
372
373
374     # -----
375     # AMR parameters:
376     # -----
377
378     amrdata = rundata.amrdata
379
380     # max number of refinement levels:
381     amrdata.amr_levels_max = 3
382
383     # List of refinement ratios at each level (length at least
384     # amr_level_max-1)
385     amrdata.refinement_ratios_x = [2,2**4]
386     amrdata.refinement_ratios_y = [1,1]
387     amrdata.refinement_ratios_t = [2,2**4]
388
389     # Specify type of each aux variable in clawdata.auxtype.
390     # This must be a list of length num_aux, each element of which is
391     # one of:
392     # 'center', 'capacity', 'xleft', or 'yleft' (see documentation)
393     .
394     amrdata.aux_type = ['center', 'center']
395
396     # Flag for refinement based on Richardson error estimator:
397     amrdata.flag_richardson = False # use Richardson?
398     amrdata.flag_richardson_tol = 0.001000e+00 # Richardson tolerance
399
400     # Flag for refinement using routine flag2refine:
401     amrdata.flag2refine = True # use this?
402     amrdata.flag2refine_tol = 1e-4 # tolerance used in this routine
403     # User can modify flag2refine to change the criterion for flagging.
404     # Default: check maximum absolute difference of first component of
405     # q
406     # between a cell and each of its neighbors.
407
408     # steps to take on each level L between regriddings of level L+1:
409     amrdata.regrid_interval = 2
410
411     # width of buffer zone around flagged points:
412     # (typically the same as regrid_interval so waves don't escape):
413     amrdata.regrid_buffer_width = 2
414
415     # clustering alg. cutoff for (# flagged pts) / (total # of cells
416     # refined)

```

```

414 # (closer to 1.0 => more small grids may be needed to cover flagged
      cells)
415 amrdata.clustering_cutoff = 0.7
416
417 # print info about each regridding up to this level:
418 amrdata.verbosity_regrid = 0
419
420
421 # -----
422 # Regions:
423 # -----
424 rundata.regiondata.regions = []
425 # to specify regions of refinement append lines of the form
426 # [minlevel,maxlevel,t1,t2,x1,x2,y1,y2]
427
428
429 # ----- For developers -----
430 # Toggle debugging print statements:
431 amrdata.dprint = False      # print domain flags
432 amrdata.eprint = False      # print err est flags
433 amrdata.edebug = False      # even more err est flags
434 amrdata.gprint = False      # grid bisection/clustering
435 amrdata.nprint = False      # proper nesting output
436 amrdata.pprint = False      # proj. of tagged points
437 amrdata.rprint = False      # print regridding summary
438 amrdata.sprint = False      # space/memory output
439 amrdata.tprint = False      # time step reporting each level
440 amrdata.uprint = False      # update/upbnd reporting
441
442 return rundata
443
444 # end of function setrun
445 # -----
446
447
448 if __name__ == '__main__':
449     # Set up run-time parameters and write all data files.
450     import sys
451     rundata = setrun(*sys.argv[1:])
452     rundata.write()

```

# Bibliography

- [1] S. S. Antman. The equations for large vibrations of strings. *American Mathematical Monthly*, pages 359–370, 1980.
- [2] S. S. Antman. Nonlinear problems of elasticity. *Applied Mathematical Sciences*, 107, 2005.
- [3] Y. Bařar and D. Weichert. *Nonlinear Continuum Mechanics of Solids: Fundamental Mathematical and Physical Concepts*. Springer, 2000.
- [4] M. J. Berger and R. J. LeVeque. Adaptive mesh refinement using wave-propagation algorithms for hyperbolic systems. *SIAM J. Numer. Anal.*, 35:2298–2316, 1998.
- [5] I. Blekhman and K. Lurie. On dynamic materials. *Proceedings of the Russian Academy of Sciences (Doklady)*, 171(1):1–5, 2000.
- [6] I. Blekhman and K. A. Lurie. On dynamic materials. In *Doklady Physics*, volume 45, pages 118–121. Springer, 2000.
- [7] I. I. Blekhman. *Vibrational Mechanics: Nonlinear Dynamic Effects, General Approach, Applications*. World Scientific, 2000.
- [8] J. M. Burgers. A mathematical model illustrating the theory of turbulence. *Adv. in Appl. Mech.*, 1:171–199, 1948.
- [9] Clawpack Development Team. Clawpack software. <http://www.clawpack.org>, 2014. Version 5.2.
- [10] J. D. Cole. On a quasi-linear parabolic equation occurring in aerodynamics. *Quarterly of applied mathematics*, 9(3):225–236, 1951.
- [11] R. Fazio and R. J. LeVeque. Moving-mesh methods for one-dimensional hyperbolic problems using clawpack. *Computers and Mathematics with Applications*, 45:273–298, 2003.
- [12] To Hansun. *Homogenization of Dynamic Materials*. PhD thesis, Temple University, 2004.



- [13] E. Hopf. The partial differential equation  $u_t + uu_x = \mu x x$ . *Communications on Pure and Applied mathematics*, 3(3):201–230, 1950.
- [14] M. Humi. A generalized cole–hopf transformation for nonlinear odes. *Journal of Physics A: Mathematical and Theoretical*, 46(32):325202, 2013.
- [15] S. Krylov and K. A. Lurie. Compliant structures with time-varying moment of inertia and non-zero average momentum and their applications in angular rate microsensors. *J. of Sounds and Vibration*, 330:4875–4895, 2011.
- [16] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*, volume 31. Cambridge University Press, 2002.
- [17] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady State and Time Dependent Problems*, volume 98. SIAM, 2007.
- [18] W. H. Louisell. *Coupled mode and parametric electronics*. Wiley, 1960.
- [19] A. I. Lurie. *Theory of Elasticity: Foundations of engineering mechanics*. Springer, Berlin, 2005.
- [20] A. I. Lurie. *Non-linear Theory of Elasticity*. Elsevier, 2012.
- [21] K. A. Lurie. Effective properties of smart elastic laminates and the screening phenomenon. *International Journal of Solids and Structures*, 34(13):1633–1643, 1997.
- [22] K. A. Lurie. The problem of effective parameters of a mixture of two isotropic dielectrics distributed in space-time and the conservation law for wave impedance in one-dimensional wave propagation. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 454(1975):1767–1779, 1998.
- [23] K. A. Lurie. Control in the coefficients of linear hyperbolic equations via spacio-temporal components. *Series on Advances in Mathematics for Applied Sciences*, 50:285–315, 1999.
- [24] K. A. Lurie. *An Introduction to the Mathematical Theory of Dynamic Materials*, volume 15 of *Advances in Mechanics and Mathematics*. Springer, 2007.
- [25] K. A. Lurie and S. L. Weekes. Effective and averaged energy densities in one-dimensional wave propagation through spatio-temporal dielectric laminates with negative effective values of  $\epsilon$  and  $\mu$ . *Nonlinear Analysis and Applications: To V. Lakshmikantham on his 80th Birthday*, page 767, 2003.

- [26] K. A. Lurie and S. L. Weekes. Wave propagation and energy exchange in a spatio-temporal material composite with rectangular microstructure. *Journal of Mathematical Analysis and Applications*, 314(1):286–310, 2006.
- [27] K. A. Lurie, S. L. Weekes, and D. Onofrei. Mathematical analysis of the waves propagation through a rectangular material structure in space–time. *Journal of Mathematical Analysis and Applications*, 355(1):180–194, 2009.
- [28] Graeme W Milton and Ornella Mattei. Field patterns: a new mathematical object. In *Proc. R. Soc. A*, volume 473, page 20160819. The Royal Society, 2017.
- [29] J. B. Pendry and D. R. Smith. Reversing light with negative refraction. *Physics today*, 57:37–43, 2004.
- [30] M. Rousseau, G. A. Maugin, and M. Berezovski. Elements of study on dynamic materials. *Archive of Applied Mechanics*, 81(7):925–942, 2011.
- [31] W. C. Sanguinet. The homogenized equations of motion for an activated elastic lamination in plane strain. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 91(12):944–956, 2011.
- [32] W. C. Sanguinet and K. A. Lurie. Propagation of dilatation and shear waves through a dynamic checkerboard material in 1d-space+time. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 93(12):937–943, 2013.
- [33] F. Santosa and W. W. Symes. A dispersive effective medium for wave propagation in periodic composites. *SIAM Journal on Applied Mathematics*, 51(4):984–1005, 1991.
- [34] C Truesdell. Mechanics of solids. *Encyclopedia of Physics*, pages 93–116, 1984.
- [35] C. Truesdell and W. Noll. *The Non-linear Field Theories of Mechanics*. Springer, 2004.
- [36] V. G. Veselago. The electrodynamics of substances with simultaneously negative values of  $\epsilon$  and  $\mu$ . *Physics-Uspokhi*, 10(4):509–514, 1968.
- [37] S. L. Weekes. Numerical computation of wave propagation in dynamic materials. *Applied Numerical Mathematics*, 37(4):417–440, 2001.
- [38] S. L. Weekes. A stable scheme for the numerical computation of long wave propagation in temporal laminates. *Journal of Computational Physics*, 176(2):345–362, 2002.

- [39] S. L. Weekes. A dispersive effective equation for wave propagation through dynamic laminates. *Wave Motion*, 38(1):25–41, 2003.
- [40] M. Zrinyi, L. Barsi, D. Szabo, and H. G. Kilian. Direct observation of abrupt shape transition in ferrogels induced by nonuniform magnetic field. *The Journal of chemical physics*, 106(13):5685–5692, 1997.