Linear Stability Analysis of a Solidifying Ternary Alloy

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Terrance J. Flynn, Jr.
Bachelor of Science
Southern Illinois University, 2000

Director: Dr. Daniel Anderson, Professor
Department of Mathematical Sciences

Spring Semester 2009
George Mason University
Fairfax, VA

Dedication

To Jennifer, Terry and Nancy. I am finished now.

Acknowledgments

I would like to acknowledge my wife and her patience. It has not gone unnoticed. I must also recognize Dr. Daniel Anderson. His generosity with his time, ideas and research have been immeasurable. Not only has he been an advisor, but an academic role model as well.

Table of Contents

List of Tables

List of Figures

Abstract

LINEAR STABILITY ANALYSIS OF A SOLIDIFYING TERNARY ALLOY

Terrance J. Flynn, Jr., MS

George Mason University, 2009

Thesis Director: Dr. Daniel Anderson

Solidification occurs frequently in many natural and industrial settings. Since producing solids from liquids impacts many aspects of daily life, it is worthwhile to understand this process. Modeling the transformation of a liquid into a solid for multicomponent systems can, among other things, provide insights into the quality of the final solid.

In recent years, mathematical models describing the solidification of aqueous ternary alloys have been proposed. These models include governing equations and boundary conditions for each of the four major layers and interfaces present during the phase transition from liquid to solid. The four layers consist of a completely solid and completely liquid layer separated by two distinct mushy layers. The mushy layers are composed of both solidified material and residual liquid and are treated as reactive porous regions. Here, reactive means the amount of solid occupying the mush is influenced by the local temperature and liquid composition. Furthermore, these solute and temperature fields are coupled to the fluid velocity. Tracking the general motion of the fluid within these mushy layers then becomes important to understanding their growth.

This work seeks to improve earlier models describing the change of phase in aqueous ternary alloy systems. One such improvement is the addition of equations allowing transport of heat and solute by both diffusion and convection. With these enhancements made, a base

state problem and solution are identified for the new model. The linear stability of the basic solution is investigated numerically using a Chebyshev pseudospectral collocation method. Results of the analysis are given in the form of neutral stability curves which describe the base state's linear stability to infinitesimal perturbations for some specific cases.

Chapter 1: Introduction

1.1 Preliminaries

The topic of solidification has been studied for many years. During this time, theoretical and experimental studies have been performed by both industrial and academic researchers. Solidification involves the transformation of a substance from its liquid phase to its solid phase. This process is often brought about by sufficiently cooling the liquid until crystallization can occur. One simple example of solidification is the freezing of water into ice. In this case, only a single, pure material is involved. In other cases, the liquid being solidified may have multiple components, in which case an alloy is formed. For instance, some alloys are created by solidifying two constituent materials. Such alloys are referred to as binary alloys. Still other alloys are formed from three components and are called ternary alloys.

During binary alloy solidification, a layer frequently develops between the completely solid and liquid regions of the material. This layer involves a blend of solid and liquid material together. Due to the presence of both liquid and solid phases, this layer is referred to as the mushy layer. When a three component, or ternary alloy is solidified, two such mushy layers form between the completely liquid and solid regions. The two mushy layers are distinguished by the number of components found to be in their solid phases within the mushy layers. Such a ternary system is shown in figure 1.1. Understanding the formation, development and interactions of these mushy layers within ternary alloys can help to characterize the resulting solid.

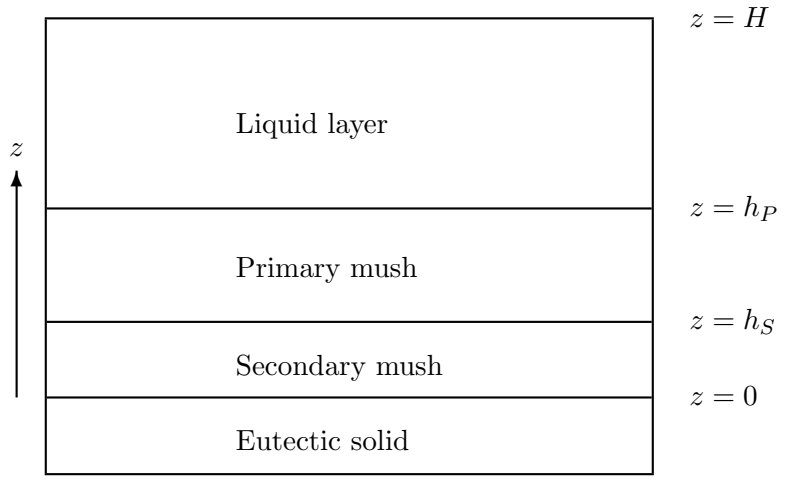

Figure 1.1: The basic geometry of a solidifying ternary system.

1.2 Review of Literature

In recent years, solidification of ternary alloys has become increasingly studied. Many of these studies are built upon prior work done for binary alloys. The work here is no exception. The works of [2] and [3] have provided the motivation for the present effort. In [2], a model is described which accounts for several characteristics of a solidifying ternary alloy. Governing equations and boundary conditions are presented which predict quantities of the alloy through the four main layers (solid, secondary mush, primary mush and liquid) and at the interfaces between each layer. This model is a natural extension of the binary model presented in [7]. One limitation of the model in [2] is that transport of heat and solute is accomplished exclusively by diffusion. Transport of heat and solute by convection is not considered under this model. Similarly, [3] develops a model which treats diffusion of heat as well as convection, but not diffusion of species. While this is an improvement over [2], it is still not complete. To complete the model, the effects of diffusion, both of heat and solute, as well as convection are incorporated in the present study.

Chapter 2: Statement of Problem

2.1 Ternary Phase Diagram

During solidification, different regions of an alloy may be in different phases depending on the local liquid composition and temperature. Given a composition and temperature, a phase diagram describes what material phase will be present. In the case of an equilibrium binary phase diagram, the horizontal axis represents liquid composition and the vertical axis defines temperature. Figure 2.1 is a phase diagram for a simple, symmetric binary alloy with components B and C. The lines separating the completely liquid region from the mush phase are the liquidus lines having slopes m_B and m_C.

The concept of a binary phase diagram can be extended by adding a third component. For instance, if a third component A is included, then three binary phase diagrams can be constructed, one for each pair of A,B and C. These three binary phase diagrams can be combined to form a ternary phase diagram.

A simple ternary phase diagram can be found in [2]. In that work, a simplified ternary phase diagram for a system of three components A, B and C is shown. By combining the three binary phase diagrams for each pair AB, AC and BC at their shared corners, a 3D prism shaped object is formed whose base is an equilateral triangle. Within the interior of this object, a three dimensional surface can be drawn. This surface is comprised of three smaller liquidus surfaces, each of which is separated by a cotectic curve which extends from the binary eutectic of each side toward the ternary eutectic point. As the system is cooled, one of three components will begin to solidify. The remaining liquid will then become enriched in the other two components. A path along one of the liquidus surfaces can then

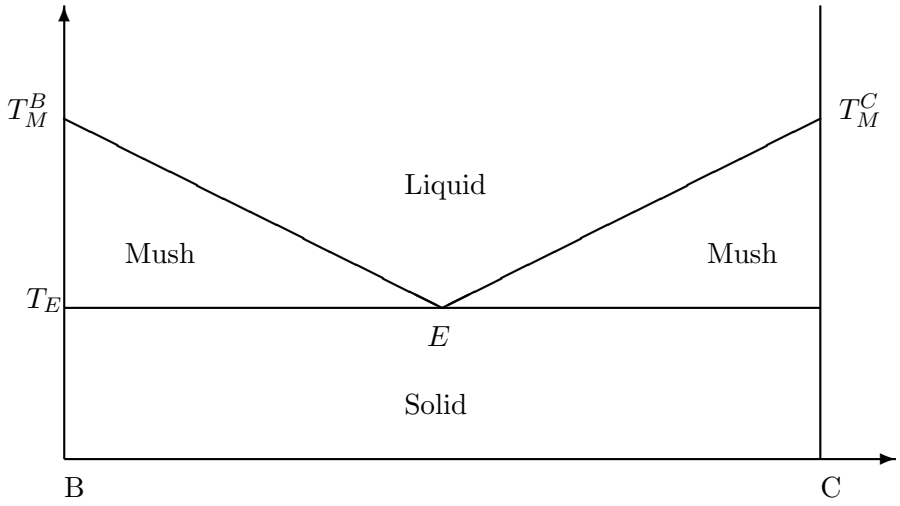

Figure 2.1: A simple binary phase diagram

be traced which moves away from the corner of the solidifying component. This corresponds to the development of the primary mushy layer. Upon further cooling, the solidification path will reach one of the three cotectic curves at which time a second component will undergo a phase transition from liquid to solid. At this point, the solidification path will move along the cotectic curve and will be in the direction of the corner of the final unsolidified component. This corresponds to the development of the secondary mushy layer. As the temperature is lowered further, the ternary eutectic point of the phase diagram will be reached and all three components will be solid. For simplicity, the liquidus surfaces and cotectic curves are assumed to be linear.

2.2 Full Governing Equations and Boundary Conditions

The model under study specifies equations describing several characteristics of a solidifying aqueous solution. Equations exist which represent temporally and spatially varying quantities such as fluid flow; liquid composition; liquid and solid fractions; and temperature. Each of these bulk quantities can be tracked through the various layers. Here, only the liquid, primary and secondary mushy layers are of interest. Boundary conditions which

relate bulk properties across the liquid-mush boundary, given by $z = h_P$, and the mush-mush boundary, given by $z = h_S$, exist as well. Positions of these free boundaries must additionally be determined. Furthermore, it is assumed that the solution is in a laboratory frame of reference translating in the z-direction at a speed V. Accordingly, the dimensional governing equations in the liquid layer, primary mushy layer and secondary mushy layer as well as the accompanying boundary conditions are as follows. Quantities involving a prime denote dimensional variables.

2.2.1 Liquid Layer

The conditions

$$T = T^{\infty} \tag{2.1a}$$

$$B = B^{\infty} \tag{2.1b}$$

$$C = C^{\infty} \tag{2.1c}$$

are imposed in the far-field of the semi-infinite liquid region as $z' \to \infty$.

Within the liquid layer,

$$-\nabla' p' + \mu' \nabla'^2 \vec{u'} + \rho' \vec{g'} = 0 \tag{2.2a}$$

$$\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'} \right) B + \vec{u'} \cdot \nabla' B = D_B \nabla'^2 B \tag{2.2b}$$

$$\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'} \right) C + \vec{u'} \cdot \nabla' C = D_C \nabla'^2 C \tag{2.2c}$$

$$\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'} \right) T' + \vec{u'} \cdot \nabla' T' = \kappa' \nabla'^2 T'. \tag{2.2d}$$

Equation (2.2a) is the Stoke's flow equation describing the fluid velocity $\vec{u'}$ in the completely

liquid layer where p' is the pressure; μ' is the dynamic viscosity; ρ' is the fluid density; and $\vec{g'}$ is the downward force due to gravity. Equations (2.2b) and (2.2c) describe the diffusion of the two solutes B and C, respectively, through the liquid. D_B and D_C are the solutal diffusivity constants for the components B and C. Similarly, equation (2.2d) is the heat equation in the liquid layer where κ is the constant thermal diffusivity of the liquid and T represents temperature. Only two diffusion equations are required since the three components (A, B, and C) are related by $A + B + C = 1$.

At $z' = h_P$, the interface separating the completely liquid region from the primary mushy layer, we have the following boundary conditions.

$$L_v(\vec{V_I'} \cdot \hat{n})[\phi_A]_-^+ = [\bar{k}\nabla' T' \cdot \hat{n}]_-^+ \tag{2.3a}$$

$$(\vec{V_I'} \cdot \hat{n})B[\phi_A]_-^+ = D_B'[\chi\nabla' B \cdot \hat{n}]_-^+ \tag{2.3b}$$

$$(\vec{V_I'} \cdot \hat{n})C[\phi_A]_-^+ = D_C'[\chi\nabla' C \cdot \hat{n}]_-^+ \tag{2.3c}$$

$$[T']_-^+ = 0 \tag{2.3d}$$

$$[B]_-^+ = 0 \tag{2.3e}$$

$$[C]_-^+ = 0 \tag{2.3f}$$

$$T' = T^{\mathcal{L}}(B, C) \tag{2.3g}$$

$$(\nabla' T' \cdot \hat{n})|^+ = m_B'(\nabla' B \cdot \hat{n})|^+ + m_C'(\nabla' C \cdot \hat{n})|^+ \tag{2.3h}$$

$$\vec{u'} \cdot \hat{n}|_-^+ = 0 \tag{2.3i}$$

$$\vec{u'} \cdot \hat{t}|_-^+ = 0 \tag{2.3j}$$

$$p'|_-^+ = 0 \tag{2.3k}$$

Here, L_v represents the latent heat, $\hat{n}$ is the unit vector normal to the liquid-mush interface and $\bar{k}$ is the solid/liquid weighted thermal conductivity. Other quantities from (2.3a) are the vector specifying the velocity of the liquid-mush interface $\vec{V_I'} = \frac{\partial h_P}{\partial t}\hat{k}$ and ϕ_A, the fraction of solid A. Several of the liquid-mush boundary conditions also involve χ, which is the fraction of liquid per unit volume. The liquid and solid fractions are related by the condition that $\phi_A + \phi_B + \phi_C + \chi = 1$ everywhere. In boundary condition (2.3h), m_B' and m_C' represent the slopes of the two liquidus lines on the BC side of the ternary phase diagram. Finally, in equation (2.3j), $\hat{t}$ is the unit vector tangent to the liquid-mush interface.

The symbol $|_-^+$ represents a jump in a quantity across the interface. So, for example, (2.3d) represents the jump in temperature across the primary interface and is the difference between the temperature on the liquid side of the interface and the temperature evaluated on the other side of the interface in the primary mush. In other words, (2.3d) says that $T|^+ - T|^- = 0$ where $T|^+$ is the temperature just above the interface and $T|^-$ is the temperature just below the interface.

2.2.2 Primary Mushy Layer

In the primary mushy layer,

$$\vec{u'} = (\Pi'(\phi)/\mu')[-\nabla' p' + \rho' \vec{g'}] \tag{2.4a}$$

$$\left(\frac{\partial}{\partial t'} - V'\frac{\partial}{\partial z'}\right)(\chi B) + \vec{u'} \cdot \nabla' B = \nabla' \cdot (D_B \chi \nabla' B) \tag{2.4b}$$

$$\left(\frac{\partial}{\partial t'} - V'\frac{\partial}{\partial z'}\right)(\chi C) + \vec{u'} \cdot \nabla' C = \nabla' \cdot (D_C \chi \nabla' C) \tag{2.4c}$$

$$\bar{c'}\left(\frac{\partial}{\partial t'} - V'\frac{\partial}{\partial z'}\right)T' - L_v'\left(-\frac{\partial}{\partial t'} + V'\frac{\partial}{\partial z'}\right)\chi + \bar{c'}(\vec{u'} \cdot \nabla' T') = \bar{k'}\nabla'^2 T' \tag{2.4d}$$

$$T' = T^{\mathcal{L}}(B, C) = T_M' + m_B' B + m_C' C. \tag{2.4e}$$

In the primary mushy layer, it is assumed that there exists one component, assume A, which is beginning to transform from its liquid phase into its solid phase. Therefore, the primary mush is comprised of both solid and liquid within the pores of the solid. Because of this, fluid flow in this porous layer is described by Darcy's equation (2.4a). Fluid motion is influenced by the permeability of the mush $\Pi(\phi)$ which is a function of the local solid fraction ϕ. Equations (2.4b) and (2.4c) describe the diffusion of the two components B and C which are still dissolved in the remaining liquid and have not yet begun to solidify. Equation (2.4d) represents the diffusion of heat through the primary mushy layer where $\bar{c}'$ and $\bar{k}'$ are the constant specific heat and thermal conductivity of the primary mush. Finally, (2.4e) relates changes in liquid compositions B and C to changes in temperature through the liquidus line slopes m'_B and m'_C. T'_M is the melting temperature of A. This condition is a statement of local thermodynamic equilibrium in the primary mush.

At $z' = h_S$, the interface separating the primary and secondary mushy layers, we have

the following boundary conditions.

$$L_v(\vec{V_I'} \cdot \hat{n})[\phi_A + \phi_B]_-^+ = [\bar{k}\nabla' T' \cdot \hat{n}]_-^+ \tag{2.5a}$$

$$(\vec{V_I'} \cdot \hat{n})\{B[\phi_A]_-^+ + (B-1)[\phi_B]_-^+\} = D_B'[\chi \nabla' B \cdot \hat{n}]_-^+ \tag{2.5b}$$

$$(\vec{V_I'} \cdot \hat{n})C[\phi_A + \phi_B]_-^+ = D_C[\chi \nabla C \cdot \hat{n}]_-^+ \tag{2.5c}$$

$$[T']_-^+ = 0 \tag{2.5d}$$

$$[B]_-^+ = 0 \tag{2.5e}$$

$$[C]_-^+ = 0 \tag{2.5f}$$

$$B = B^{\mathcal{C}}(T') \tag{2.5g}$$

$$C = C^{\mathcal{C}}(T') \tag{2.5h}$$

$$m_B^{\mathcal{C}} \nabla B \cdot \hat{n}|^+ = m_C^{\mathcal{C}} \nabla C \cdot \hat{n}|^+ \tag{2.5i}$$

$$\vec{u'} \cdot \hat{n}|^+ \tag{2.5j}$$

$$p'|_-^+ = 0 \tag{2.5k}$$

Here, $\vec{V_I'} = \frac{\partial h_S}{\partial t}\hat{k}$ is the velocity of the mush-mush interface.

2.2.3 Secondary Mushy Layer

The governing equations in the secondary mushy layer are

$$\vec{u'} = (\Pi'(\phi)/\mu')[-\nabla' p' + \rho' \vec{g'}] \tag{2.6a}$$

$$\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'}\right)(\chi B + \phi_B) + \vec{u'} \cdot \nabla' B = \nabla' \cdot (D_B \chi \nabla' B) \tag{2.6b}$$

$$\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'}\right)(\chi C) + \vec{u'} \cdot \nabla' C = \nabla' \cdot (D_C \chi \nabla' C) \tag{2.6c}$$

$$\bar{c'}\left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'}\right) T' + L'_v \left(\frac{\partial}{\partial t'} - V' \frac{\partial}{\partial z'}\right) \chi + \bar{c'}(\vec{u'} \cdot \nabla' T') = \bar{k'} \nabla'^2 T' \tag{2.6d}$$

$$B = B^{\mathcal{C}}(T) = -\frac{1}{m_B^{\mathcal{C}}}(T' - T'_E) + B_E \tag{2.6e}$$

$$C = C^{\mathcal{C}}(T) = -\frac{1}{m_C^{\mathcal{C}}}(T' - T_E^{AB'}). \tag{2.6f}$$

In the secondary mushy layer, component A continues to form a solid. Additionally, a second component, assume B, begins to solidify along with A. Once again, due to the presence of fluid within the voids of the solid, the secondary mushy layer can be modeled as a porous layer and fluid flow can again be described using Darcy's equation (2.6a). Liquid composition is given by (2.6b) and (2.6c). Equations (2.6e) and (2.6f) relate changes in temperature to changes in liquid composition throughout the secondary mushy layer. These equations impose local thermodynamic equilibrium throughout the secondary mushy layer. The ternary phase diagram quantity B_E is the liquid composition of B at the ternary eutectic point E. Also, T_E^{AB} is the eutectic temperature from the AB binary phase diagram.

The density of the liquid within any given layer is

$$\rho' = \rho'_0(1 + \alpha'(T' - T'_P) + \beta_B B + \beta_C C). \tag{2.7}$$

Here, ρ_0 is a reference density, α is the thermal expansion coefficient, and β_B and β_C are constant solutal expansion coefficients describing changes in density with changes in liquid composition B and C, respectively. The temperature at the liquid-mush interface is given by T'_P.

Chapter 3: Solution

3.1 Nondimensionalization

To nondimensionalize the primed, dimensional quantities in the governing equations and boundary conditions above, we use the following scalings from [3]:

$$\vec{u} = \frac{\vec{u'}}{V}, \qquad \vec{x} = \frac{\vec{x'}}{(\kappa/V)}, \qquad t = \frac{t'}{(\kappa/V^2)}, \qquad \nabla = \frac{\kappa}{V}\nabla', \qquad T = \frac{T' - T'_P}{\Delta T'}$$

where $\Delta T' = T'_P - T'_E$, the difference between the temperatures at the liquid-mush interface and at the mush-solid interface. Additionally, we redefine the fluid velocity in terms of the scalar streamfunction ψ such that

$$\vec{u} = \left(-\frac{\partial \psi}{\partial z}, 0, \frac{\partial \psi}{\partial x}\right).$$

From the definition of the streamfunction, we observe that

$$\nabla \cdot \vec{u} = 0$$

is satisfied. From these scalings we obtain the following set of dimensionless governing equations and boundary conditions.

In the liquid layer, the nondimensionalized governing equations are

$$Da\nabla^4\psi = Ra\frac{\partial T}{\partial x} + Ra_B\frac{\partial B}{\partial x} + Ra_C\frac{\partial C}{\partial x} \tag{3.1a}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)B + \vec{u}\cdot\nabla B = \frac{1}{\delta_B^2}\nabla^2 B \tag{3.1b}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)C + \vec{u}\cdot\nabla C = \frac{1}{\delta_C^2}\nabla^2 C \tag{3.1c}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)T + \vec{u}\cdot\nabla T = \nabla^2 T \tag{3.1d}$$

where

$$Da \equiv \frac{V^2\Pi_0}{\kappa^2}, \qquad Ra \equiv \frac{g\alpha\Delta T\Pi_0}{\nu V}, \qquad Ra_B \equiv \frac{g\beta_B\Pi_0}{\nu V} \quad \text{and} \quad Ra_C \equiv \frac{g\beta_C\Pi_0}{\nu V}.$$

are dimensionless parameters of the system. The first of these parameters, Da, is the Darcy number and the other three parameters are Rayleigh numbers. The first Rayleigh number, Ra, is the thermal Rayleigh number and is given in terms of the thermal expansion coefficient α. The other two Rayleigh numbers, Ra_B and Ra_C are solutal Rayleigh numbers. The three Rayleigh numbers can be though of as ratios of destabalizing forces to stabalizing forces. These Rayleigh numbers apply continuously over all three layers of interest. In equations (3.1b) and (3.1c), the terms δ_B^2 and δ_C^2 are defined as

$$\delta_B^2 \equiv \kappa/D_B \qquad \delta_C^2 \equiv \kappa/D_C.$$

The dimensionless boundary conditions at the liquid-mush interface are

$$\vec{V_I} \cdot \hat{n}[\phi_A]_-^+ = \left[\frac{1}{S}\nabla T \cdot \hat{n}\right]_-^+ \tag{3.2a}$$

$$\vec{V_I} \cdot \hat{n}B[\phi_A]_-^+ = \frac{1}{\delta_B^2}[\chi \nabla B \cdot \hat{n}]_-^+ \tag{3.2b}$$

$$\vec{V_I} \cdot \hat{n}C[\phi_A]_-^+ = \frac{1}{\delta_C^2}[\chi \nabla C \cdot \hat{n}]_-^+ \tag{3.2c}$$

$$[T]_-^+ = 0 \tag{3.2d}$$

$$T^{\mathcal{L}}(z = h_P) = T^{\mathcal{L}}(B, C) = T_M + M_B B(z = h_P) + M_C C(z = h_P) \tag{3.2e}$$

$$[B(z = h_P)]_-^+ = 0 \tag{3.2f}$$

$$[C(z = h_P)]_-^+ = 0 \tag{3.2g}$$

$$\nabla T \cdot \hat{n}|^+ = M_B \nabla B \cdot \hat{n}|^+ + M_C \nabla C \cdot \hat{n}|^+ \tag{3.2h}$$

$$\vec{u} \cdot \hat{n}|_-^+ = 0 \tag{3.2i}$$

$$\vec{u} \cdot \hat{t}|_-^+ = 0 \tag{3.2j}$$

$$p|_-^+ = 0 \tag{3.2k}$$

where T_M is the dimensionless melting temperature

$$T_M = \frac{T'_M - T'_P}{\Delta T'}.$$

In the primary mushy layer, the dimensionless governing equations are

$$\nabla^2\psi = \frac{1}{\Pi}(\nabla\Pi\cdot\nabla\psi) - \Pi\left(Ra\frac{\partial T}{\partial x} + Ra_B\frac{\partial B}{\partial x} + Ra_C\frac{\partial C}{\partial x}\right) \tag{3.3a}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)(\chi B) + \vec{u}\cdot\nabla B = \nabla\cdot\left(\frac{1}{\delta_B^2}\chi\nabla B\right) \tag{3.3b}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)(\chi C) + \vec{u}\cdot\nabla C = \nabla\cdot\left(\frac{1}{\delta_C^2}\chi\nabla C\right) \tag{3.3c}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)T + S\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)\chi + \vec{u}\cdot\nabla T = \nabla^2 T \tag{3.3d}$$

$$T = T_M + B_B B + M_C C \tag{3.3e}$$

where the Stefan number S is defined to be $S \equiv L_v/(\bar{c}\Delta T)$. The rescaled liquidus line slopes are

$$M_B = \frac{m_B'}{\Delta T'} \qquad M_C = \frac{m_C'}{\Delta T'}.$$

After nondimensionalizing, the boundary conditions at the mush-mush interface are

$$\vec{V_I} \cdot \hat{n}[\phi_A + \phi_B]_-^+ = \frac{1}{S}[\nabla T \cdot \hat{n}]_-^+ \tag{3.4a}$$

$$\vec{V_I} \cdot \hat{n}\{B[\phi_A]_-^+ + (B-1)[\phi_B]_-^+\} = \frac{1}{\delta_B^2}[\chi \nabla B \cdot \hat{n}]_-^+ \tag{3.4b}$$

$$\vec{V_I} \cdot \hat{n} C[\phi_A + \phi_B]_-^+ = \frac{1}{\delta_C^2}[\chi \nabla C \cdot \hat{n}]_-^+ \tag{3.4c}$$

$$[T]_-^+ = 0 \tag{3.4d}$$

$$[B]_-^+ = 0 \tag{3.4e}$$

$$[C]_-^+ = 0 \tag{3.4f}$$

$$B(z = h_S) = -\frac{1}{M_B^{\mathcal{C}}}(T - T_E) + B_E \tag{3.4g}$$

$$C(z = h_S) = -\frac{1}{M_C^{\mathcal{C}}}(T - T_E^{AB}) \tag{3.4h}$$

$$M_B^{\mathcal{C}} \nabla B \cdot \hat{n}|^+ = M_C^{\mathcal{C}} \nabla C \cdot \hat{n}|^+ \tag{3.4i}$$

$$\vec{u} \cdot \hat{n}|_-^+ = 0 \tag{3.4j}$$

$$p|_-^+ = 0 \tag{3.4k}$$

Finally, in the secondary mushy layer, the dimensionless governing equations are

$$\nabla^2\psi = \frac{1}{\Pi}(\nabla\Pi\cdot\nabla\psi) - \Pi\left(Ra\frac{\partial T}{\partial x} + Ra_B\frac{\partial B}{\partial x} + Ra_C\frac{\partial C}{\partial x}\right) \tag{3.5a}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)(\chi B + \phi_B) + \vec{u}\cdot\nabla B = \nabla\cdot\left(\frac{1}{\delta_B^2}\chi\nabla B\right) \tag{3.5b}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)(\chi C) + \vec{u}\cdot\nabla C = \nabla\cdot\left(\frac{1}{\delta_C^2}\chi\nabla C\right) \tag{3.5c}$$

$$\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)T + S\left(\frac{\partial}{\partial t} - \frac{\partial}{\partial z}\right)\chi + \vec{u}\cdot\nabla T = \nabla^2 T \tag{3.5d}$$

$$B = -\frac{1}{M_B^{\mathcal{C}}}(T+1) + B_E \tag{3.5e}$$

$$C = -\frac{1}{M_C^{\mathcal{C}}}(T - T_E^{AB}). \tag{3.5f}$$

The dimensionless cotectic line slopes and binary eutectic temperature are scaled like

$$M_B^{\mathcal{C}} = \frac{m_B^{\mathcal{C}'}}{\Delta T'} \qquad M_C^{\mathcal{C}} = \frac{m_C^{\mathcal{C}'}}{\Delta T'} \qquad T_E^{AB} = \frac{T_E^{AB'} - T_P'}{\Delta T'}.$$

3.2 Linear Stability Analysis

3.2.1 Base State Problem and Solutions

In order to find steady states of the system under study, we shall remove any time dependence and allow all variables to vary only in the z-direction. With these restrictions in mind, the governing equations and boundary conditions give the following basic problem.

In the far-field,

$$T = T^{\infty} \tag{3.6a}$$

$$B = B^{\infty} \tag{3.6b}$$

$$C = C^{\infty} \tag{3.6c}$$

In the liquid layer,

$$-\frac{dB}{dz} = \frac{1}{\delta_B^2}\frac{d^2B}{dz^2} \tag{3.7a}$$

$$-\frac{dC}{dz} = \frac{1}{\delta_C^2}\frac{d^2C}{dz^2} \tag{3.7b}$$

$$-\frac{dT}{dz} = \frac{d^2T}{dz^2}. \tag{3.7c}$$

At the liquid-mush interface,

$$[\bar{\phi}_A]^+_- = \frac{1}{S}\left[\frac{d\bar{T}}{dz}\right]^+_- \tag{3.8a}$$

$$\bar{B}[\bar{\phi}_A]^+_- = \frac{1}{\delta_B^2}\left[\bar{\chi}\frac{d\bar{B}}{dz}\right]^+_- \tag{3.8b}$$

$$\bar{C}[\bar{\phi}_A]^+_- = \frac{1}{\delta_C^2}\left[\bar{\chi}\frac{d\bar{C}}{dz}\right]^+_- \tag{3.8c}$$

$$[\bar{T}]^+_- = 0 \tag{3.8d}$$

$$[\bar{B}]^+_- = 0 \tag{3.8e}$$

$$[\bar{C}]^+_- = 0 \tag{3.8f}$$

$$T^{\mathcal{L}}(\bar{B},\bar{C}) = T_M + M_B\bar{B} + M_C\bar{C} \tag{3.8g}$$

$$\left.\frac{d\bar{T}}{dz}\right|^+ = M_B\left.\frac{d\bar{B}}{dz}\right|^+ + M_C\left.\frac{d\bar{C}}{dz}\right|^+. \tag{3.8h}$$

In the primary mushy layer,

$$-\frac{d}{dz}(\chi B) = \frac{d}{dz}\left(\frac{1}{\delta_B^2}\chi\frac{dB}{dz}\right) \tag{3.9a}$$

$$-\frac{d}{dz}(\chi C) = \frac{d}{dz}\left(\frac{1}{\delta_C^2}\chi\frac{dC}{dz}\right) \tag{3.9b}$$

$$-\frac{dT}{dz} - S\frac{d\chi}{dz} = \frac{d^2T}{dz^2} \tag{3.9c}$$

$$T = T_M + M_B B + M_C C. \tag{3.9d}$$

At the mush-mush interface,

$$[\bar{\phi}_A + \bar{\phi}_B]^+_- = \frac{1}{S}\left[\frac{d\bar{T}}{dz}\right]^+_- \tag{3.10a}$$

$$\bar{B}[\bar{\phi}_A]^+_- + (\bar{B}-1)[\bar{\phi}_B]^+_- = \frac{1}{\delta_B^2}\left[\bar{\chi}\frac{d\bar{B}}{dz}\right]^+_- \tag{3.10b}$$

$$\bar{C}[\bar{\phi}_A + \bar{\phi}_B]^+_- = \frac{1}{\delta_C^2}\left[\bar{\chi}\frac{d\bar{C}}{dz}\right]^+_- \tag{3.10c}$$

$$[\bar{T}]^+_- = 0 \tag{3.10d}$$

$$[\bar{B}]^+_- = 0 \tag{3.10e}$$

$$[\bar{C}]^+_- = 0 \tag{3.10f}$$

$$\bar{B} = -\frac{1}{M_B^{\mathcal{C}}}(\bar{T} - T_E) + B_E \tag{3.10g}$$

$$\bar{C} = -\frac{1}{M_C^{\mathcal{C}}}(\bar{T} - T_E^{AB}) \tag{3.10h}$$

$$M_B^{\mathcal{C}}\left.\frac{d\bar{B}}{dz}\right|^+ = M_C^{\mathcal{C}}\left.\frac{d\bar{C}}{dz}\right|^+ . \tag{3.10i}$$

In the secondary mushy layer,

$$-\frac{d}{dz}(\chi B + \phi_B) = \frac{d}{dz}\left(\frac{1}{\delta_B^2}\chi\frac{dB}{dz}\right) \tag{3.11a}$$

$$-\frac{d}{dz}(\chi C) = \frac{d}{dz}\left(\frac{1}{\delta_{BC}^2}\chi\frac{dC}{dz}\right) \tag{3.11b}$$

$$-\frac{dT}{dz} - S\frac{d\chi}{dz} = \frac{d^2T}{dz^2} \tag{3.11c}$$

$$B = -\frac{1}{M_B^C}(T+1) + B_E \tag{3.11d}$$

$$C = -\frac{1}{M_C^C}(T - T_E^{AB}). \tag{3.11e}$$

At the interface between the secondary mush and the eutectic solid,

$$T = T_E \tag{3.12a}$$

$$B = B_E \tag{3.12b}$$

$$C = C_E. \tag{3.12c}$$

Part of the solution to the basic problem is finding the position of the liquid-mush interface and mush-mush interface. Since the interface positions h_S and h_P along with the liquid composition of species B at the liquid-mush interface can not be solved for analytically, a numerical code is used to determine them. A system of three equations can be formed whose solution gives the vales for these three unknowns. The complete solution to the base state problem can be reduced to solving for just a few quantities. These quantities include the liquid composition B in the secondary mush and both liquid compositions B and C in the primary mush. These quantities are obtained as a result of finding the three unknowns h_S, h_P, and B_P and can be used to solve for all other quantities of interest.

The base state problem is solved beginning in the secondary mushy layer. At the top of the eutectic solid region, the liquid composition B_E is known. This information can be used as an initial value for the ODE involving liquid composition B in the secondary mushy region. After integrating both sides of equation (3.11a) once with respect to z, the differential equation for liquid composition B in the secondary mush is

$$\frac{dB}{dz} = -\delta_B^2 \left(B + \frac{\phi_B}{\chi} - \frac{B_\infty}{\chi} \right)$$

If a position for the mush-mush interface h_S is guessed, call it h_S^{guess}, the ODE for B above can be integrated vertically in the $+z$-direction over the thickness of the secondary mushy region starting at $z = 0$, the boundary between the secondary mush and the eutectic solid, until the guessed mush-mush boundary $z = h_S^{guess}$ is reached. This integration solves for the liquid composition B at a discrete set of points through the secondary mush. At this stage, a computed liquid composition B_S^{comp} can be obtained. Using this, a computed liquid composition C_S^{comp} at h_S^{guess} can be calculated using the expression

$$C_S = \frac{m_B^{\mathcal{C}}}{m_C^{\mathcal{C}}}(B_S - B_E^{AB}).$$

At the mush-mush interface, B_S^{comp} and C_S^{comp} can be used as initial values for the ODEs for liquid compositions B and C in the primary mushy layer. The ODEs for liquid composition B and C in the primary mushy layer are

$$\frac{dB}{dz} = \delta_B^2 \left(\frac{B_\infty}{\chi} - B \right)$$

$$\frac{dC}{dz} = \delta_C^2 \left(\frac{C_\infty}{\chi} - C \right).$$

These are the result of integrating both sides of equations (3.9a) and (3.9b) with respect to

z. Once again, if a position for the liquid-mush interface $z = h_P$ is guessed, call it h_P^{guess}, the ODEs for B and C can be integrated over the thickness of the primary mushy layer. Once h_P^{guess} is reached, the profiles for liquid compositions B and C in the primary mush are now known at a discrete set of z values. The computed liquid compositions B_P^{comp} and C_P^{comp} at $z = h_P^{guess}$ are therefore determined.

Finally, a system of three equations can now be formed. Finding the value of $\vec{x} = (h_S^{guess}, h_P^{guess}, B_P^{guess})$ which solves this system gives the solutions for the unknowns h_S, h_P and B_P. The system to be solved is

$$f(\vec{x}) = \begin{pmatrix} B_S^{comp}\left(1 - \frac{\delta_C^2}{\delta_B^2}\right) + B_E^{AB}\frac{\delta_C^2}{\delta_B^2} - \frac{1}{\chi^{S+}}\left(B_\infty - \frac{m_C^C}{m_B^C}\frac{\delta_C^2}{\delta_B^2}C_\infty\right) \\ B_P^{comp} - B_P^{guess} \\ C_P^{comp} - C_P^{guess} \end{pmatrix} = 0$$

where the quantity C_P^{guess} is found using the guessed value for liquid composition B at the liquid-mush interface, B_P^{guess}, and is given by

$$C_P^{guess} = \frac{1 - \left(\frac{m_B}{-\Delta T_\infty}\right)(\delta_B^2 - 1)(B_P^{guess} - B_\infty)}{\left(\frac{m_C}{-\Delta T_\infty}\right)(\delta_C^2 - 1)} + C_\infty.$$

The terms $\frac{m_B}{-\Delta T_\infty}$ and $\frac{m_C}{-\Delta T_\infty}$ are given by

$$\frac{m_B}{-\Delta T_\infty} = \frac{1}{\frac{T_M - T_\infty}{m_B} + B_\infty + \frac{m_C}{m_B}C_\infty}$$

$$\frac{m_C}{-\Delta T_\infty} = \frac{1}{\frac{T_M - T_\infty}{m_C} + C_\infty + \frac{m_B}{m_C}B_\infty}.$$

By solving for the vector of unknown values $\vec{x} = (h_S, h_P, B_P)$, the reduced number of quantities needed to solve for all other quantities of the system are now determined. The

remaining quantities of interest are, beginning in the secondary mushy layer layer,

$$C = \frac{-m_B^{\mathcal{C}}(B - B_E) + T_E - T_E^{AB}}{-m_C^{\mathcal{C}}}$$

$$T = -m_B^{\mathcal{C}}(B - B_E) + T_E$$

$$\phi_B = \frac{\delta_C^2}{\delta_B^2}\left((B - B_E^{AB})\chi - \frac{m_C^{\mathcal{C}}}{m_B^{\mathcal{C}}}C_\infty\right) - \chi B + B_\infty$$

$$\chi^2 - \chi\left[\frac{B - B_E}{\delta_B} + S + 1 - \frac{\delta_C^2}{\gamma_B}(B - B_E^{AB})\right] - \frac{\delta_C^2 m_C^{\mathcal{C}} C_\infty}{\gamma_B m_B^{\mathcal{C}}} = 0.$$

In the primary mushy layer, the quantities of interest are

$$T = T_M + m_B B + m_C C$$

$$\chi = \frac{1}{2}\left[1 + b_1 - b_2 + \sqrt{(1 - b_1)^2 + b_2(-2 - 2b_1 + b_2)}\right]$$

$$b_1 = \frac{1}{\gamma}\left(\delta_B^2 B_\infty \bar{m}_B + \delta_C^2 C_\infty \bar{m}_C\right)$$

$$b_2 = \frac{1}{\gamma}\left(\bar{m}_B(B - B_P)(1 - \delta_B^2) + \bar{m}_C(C - C_P)(1 - \delta_C^2)\right).$$

Finally, in the liquid layer,

$$C = C_\infty + (C_P - C_\infty)e^{\delta_C^2(h_P - z)}$$

$$B = B_\infty + (B_P - B_\infty)e^{\delta_B^2(h_P - z)}$$

$$T = T_\infty + (T_P - T_\infty)e^{(h_P - z)}.$$

The solution to the system of equations along with a few additional calculations produces a basic solution for the system. The base state solution is shown in figure 3.1 for the set

of input parameter values given in table 4.1. Figure 3.1 shows that the temperature of the basic solution is increasing with z. The liquid compositions of species B and C are nearly constant through the liquid layer. Within the primary mushy layer, liquid composition C is decreasing as z increases. Through much of the primary mushy layer, liquid composition B is decreasing as z increases. However, liquid composition B is increasing just ahead of the mush-mush interface. In the secondary mushy layer, liquid composition C is again decreasing with z while liquid composition B is increasing with z. This solution also has $\phi_A = \phi_B = \phi_C = 0$ in the liquid layer, $\phi_A \neq 0$ in the primary mushy layer and $\phi_A \neq 0 \neq \phi_B$ in the secondary mushy layer. In the eutectic solid layer, $\phi = \phi_A + \phi_B + \phi_C = 1$ and $\chi = 0$. Finally, there is no fluid flow present in any of the four layers. A variety of different behaviors, interface positions, as well as thermal, solutal and solid fraction profiles are possible. See [2] for some examples.

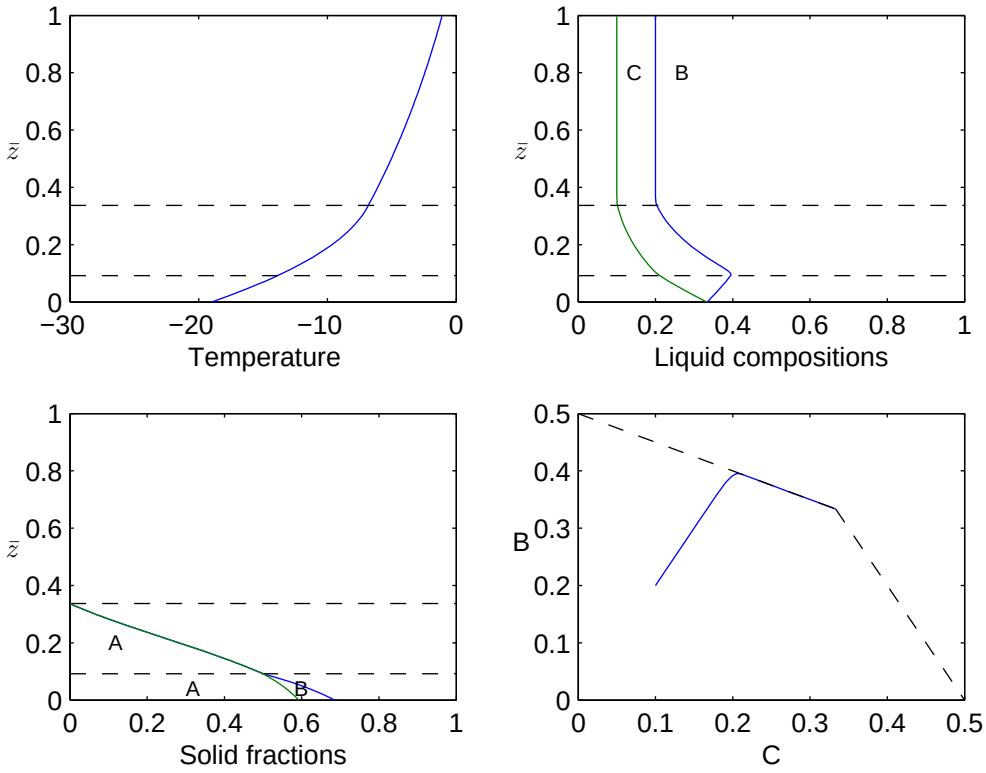

Figure 3.1: Basic solution for the parameter values shown in table 4.1.

3.2.2 Linearized Disturbance Equations

To determine the linear stability of the basic solution found above, we redefine the variables under study to include perturbation quantities. Let

$$T(x,z,t) = \bar{T}(z) + \hat{T}(z)e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$B(x,z,t) = \bar{B}(z) + \hat{B}(z)e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$C(x,z,t) = \bar{C}(z) + \hat{C}(z)e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$\chi(x,z,t) = \bar{\chi}(z) + \hat{\chi}(z)e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$\psi(x,z,t) = 0 + i\hat{\psi}(z)e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$\phi_B(x,z,t) = \bar{\phi}_B(z) + \hat{\phi}_B e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$h_P(x,z,t) = \bar{h}_P + \hat{h}_P e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

$$h_S(x,z,t) = \bar{h}_S + \hat{h}_S e^{\sigma t + i\alpha x} + \text{complex conjugate}$$

where bar denotes the base state solution and the hat and exponential terms are infinitesimal perturbations to the basic solutions. Replacing these into the full, dimensionless governing equations above gives us, after keeping only the terms linear in the unknown hat quantities, the following.

To restrict the height of the system, a maximum height of $z = H$ is selected. For a sufficiently large choice of H, this effectively approximates a semi-infinite system. At

$z = H$, the top of the liquid layer, we have the following conditions,

$$\hat{\psi}_z = 0 \tag{3.13a}$$

$$\hat{\psi}_{zzz} = 0 \tag{3.13b}$$

$$\hat{B} = 0 \tag{3.13c}$$

$$\hat{C} = 0 \tag{3.13d}$$

$$\hat{T} = 0. \tag{3.13e}$$

In the liquid layer,

$$\frac{d^4\hat{\psi}}{dz^4} - 2\alpha^2\frac{d^2\hat{\psi}}{dz^2} + \alpha^4\hat{\psi} - \frac{\alpha}{Da}(Ra\hat{T} + Ra_B\hat{B} + Ra_C\hat{C}) = 0 \tag{3.14a}$$

$$-\frac{d\hat{B}}{dz} + \sigma\hat{B} - \alpha\hat{\psi}\frac{d\bar{B}}{dz} = \frac{1}{\delta_B^2}\left(-\alpha^2\hat{B} + \frac{d^2\hat{B}}{dz^2}\right) \tag{3.14b}$$

$$-\frac{d\hat{C}}{dz} + \sigma\hat{C} - \alpha\hat{\psi}\frac{d\bar{C}}{dz} = \frac{1}{\delta_C^2}\left(-\alpha^2\hat{C} + \frac{d^2\hat{C}}{dz^2}\right) \tag{3.14c}$$

$$-\frac{d\hat{T}}{dz} + \sigma\hat{T} - \alpha\hat{\psi}\frac{d\bar{T}}{dz} = -\alpha^2\hat{T} + \frac{d^2\hat{T}}{dz^2} \tag{3.14d}$$

At the interface between the liquid and primary mush,

$$\left[\hat{h}_P\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A\right]_-^+ + \sigma\hat{h}_P[\bar{\phi}_A]_-^+ = \left[\frac{1}{S}\left(\hat{h}_P\frac{d^2\bar{T}}{dz^2}+\frac{d\hat{T}}{dz}\right)\right]_-^+ \tag{3.15a}$$

$$\left.\begin{aligned}\bar{B}\left[\hat{h}_P\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A\right]_-^+ + \left(\hat{h}_P\frac{d\bar{B}}{dz}+\hat{B}\right)[\bar{\phi}_A]_-^+ + \sigma\hat{h}_P\bar{B}[\bar{\phi}_A]_-^+ \\ = \frac{1}{\delta_B^2}\left[\bar{\chi}\left(\hat{h}_P\frac{d^2\bar{B}}{dz^2}+\frac{d\hat{B}}{dz}\right)+\left(\hat{h}_P\frac{d\bar{\chi}}{dz}+\hat{\chi}\right)\frac{d\bar{B}}{dz}\right]_-^+\end{aligned}\right\} \tag{3.15b}$$

$$\left.\begin{aligned}\bar{C}\left[\hat{h}_P\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A\right]_-^+ + \left(\hat{h}_P\frac{d\bar{C}}{dz}+\hat{C}\right)[\bar{\phi}_A]_-^+ + \sigma\hat{h}_P\bar{C}[\bar{\phi}_A]_-^+ \\ = \frac{1}{\delta_C^2}\left[\bar{\chi}\left(\hat{h}_P\frac{d^2\bar{C}}{dz^2}+\frac{d\hat{C}}{dz}\right)+\left(\hat{h}_P\frac{d\bar{\chi}}{dz}+\hat{\chi}\right)\frac{d\bar{C}}{dz}\right]_-^+\end{aligned}\right\} \tag{3.15c}$$

$$\left[M_B\left(\hat{h}_P\frac{d\bar{B}}{dz}+\hat{B}\right)+M_C\left(\hat{h}_P\frac{d\bar{C}}{dz}+\hat{C}\right)\right]_-^+ = 0 \tag{3.15d}$$

$$\left[\hat{h}_P\frac{d\bar{B}}{dz}+\hat{B}\right]_-^+ = 0 \tag{3.15e}$$

$$\left[\hat{h}^P\frac{d\bar{C}}{dz}+\hat{C}\right]_-^+ = 0 \tag{3.15f}$$

$$\hat{T} = M_B\hat{B}+M_C\hat{C} \tag{3.15g}$$

$$\left(\hat{h}_P\frac{d^2\bar{T}}{dz^2}+\frac{d\hat{T}}{dz}\right)\Big|^+ = M_B\left(\hat{h}_P\frac{d^2\bar{B}}{dz^2}+\frac{d\hat{B}}{dz}\right)\Big|^+ + M_C\left(\hat{h}_P\frac{d^2\bar{C}}{dz^2}+\frac{d\hat{C}}{dz}\right)\Big|^+ \tag{3.15h}$$

$$-\alpha\hat{\psi}\Big|_-^+ = 0 \tag{3.15i}$$

$$-i\frac{d\hat{\psi}}{dz}\Big|_-^+ = 0 \tag{3.15j}$$

$$(\alpha^2\hat{\psi}_z-\hat{\psi}_{zzz})^+ = \frac{1}{Da}\left(\frac{1}{\Pi(\phi)}\hat{\psi}_z\right)^-. \tag{3.15k}$$

Equation (3.15d) can be written as

$$\left(\hat{h_P}\frac{d\bar{T}}{dz}+\hat{T}\right)\Big|_{-}^{+}=0$$

with the help of equations (3.15e), (3.15f), (3.15g) and (3.15h).

In the primary mushy layer,

$$-\alpha^2\hat{\psi}+\frac{d^2\hat{\psi}}{dz^2}=\frac{1}{\Pi}\left(\frac{d\Pi}{d\bar{\chi}}\frac{d\bar{\chi}}{dz}\frac{d\hat{\psi}}{dz}\right)-\Pi\alpha(Ra\hat{T}+Ra_B\hat{B}+Ra_C\hat{C}) \tag{3.16a}$$

$$\left.\begin{aligned}\sigma\bar{\chi}\hat{B}+\sigma\bar{B}\hat{\chi}-\frac{d\bar{\chi}}{dz}\hat{B}-\frac{d\hat{B}}{dz}\bar{\chi}-\frac{d\hat{\chi}}{dz}\bar{B}-\frac{d\bar{B}}{dz}\hat{\chi}-\alpha\hat{\psi}\frac{d\bar{B}}{dz}\\ =\frac{1}{\delta_B^2}\left(-\alpha^2\bar{\chi}\hat{B}+\bar{\chi}\frac{d^2\hat{B}}{dz}+\frac{d\bar{\chi}}{dz}\frac{d\hat{B}}{dz}+\frac{d^2\bar{B}}{dz^2}\hat{\chi}+\frac{d\bar{B}}{dz}\frac{d\hat{\chi}}{dz}\right)\end{aligned}\right\} \tag{3.16b}$$

$$\left.\begin{aligned}\sigma\bar{\chi}\hat{C}+\sigma\bar{C}\hat{\chi}-\frac{d\bar{\chi}}{dz}\hat{C}-\frac{d\hat{C}}{dz}\bar{\chi}-\frac{d\hat{\chi}}{dz}\bar{C}-\frac{d\bar{C}}{dz}\hat{\chi}-\alpha\hat{\psi}\frac{d\bar{C}}{dz}\\ =\frac{1}{\delta_C^2}\left(-\alpha^2\bar{\chi}\hat{C}+\bar{\chi}\frac{d^2\hat{C}}{dz}+\frac{d\bar{\chi}}{dz}\frac{d\hat{C}}{dz}+\frac{d^2\bar{C}}{dz^2}\hat{\chi}+\frac{d\bar{C}}{dz}\frac{d\hat{\chi}}{dz}\right)\end{aligned}\right\} \tag{3.16c}$$

$$\sigma\hat{T}-\frac{d\hat{T}}{dz}-S\left(-\sigma\hat{\chi}+\frac{d\hat{\chi}}{dz}\right)-\alpha\hat{\psi}\frac{d\bar{T}}{dz}=-\alpha^2\hat{T}+\frac{d^2\hat{T}}{dz^2} \tag{3.16d}$$

$$\hat{T}=M_B\hat{B}+M_C\hat{C}. \tag{3.16e}$$

At the boundary between the primary and secondary mush,

$$\left[\hat{h}_S\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A+\hat{h}_S\frac{d\bar{\phi}_B}{dz}+\hat{\phi}_B\right]_-^+ + \sigma\hat{h}_S[\bar{\phi}_A+\bar{\phi}_B]_-^+ = \frac{1}{S}\left[\hat{h}_S\frac{d^2\bar{T}}{dz^2}+\frac{d\hat{T}}{dz}\right]_-^+ \tag{3.17a}$$

$$\left.\begin{aligned}&\left(\hat{h}_S\frac{d\bar{B}}{dz}+\hat{B}\right)[\bar{\phi}_A+\bar{\phi}_B]_-^+ + (\bar{B}-1)\left[\hat{h}_S\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A+\hat{h}_S\frac{d\bar{\phi}_B}{dz}+\hat{\phi}_B\right]_-^+\\ &+\sigma\hat{h}_S\{(\bar{B}-1)[\bar{\phi}_A+\bar{\phi}_B]_-^+\}\\ &=\frac{1}{\delta_B^2}\left[\bar{\chi}\left(\hat{h}_S\frac{d^2\bar{B}}{dz^2}+\frac{d\hat{B}}{dz}\right)+\left(\hat{h}_S\frac{d\bar{\chi}}{dz}+\hat{\chi}\right)\frac{d\bar{B}}{dz}\right]_-^+\end{aligned}\right\} \tag{3.17b}$$

$$\left.\begin{aligned}&\bar{C}\left[\hat{h}_S\frac{d\bar{\phi}_A}{dz}+\hat{\phi}_A+\hat{h}_S\frac{d\bar{\phi}_B}{dz}+\hat{\phi}_B\right]_-^+ + \left(\hat{h}_S\frac{d\bar{C}}{dz}+\hat{C}\right)[\bar{\phi}_A+\bar{\phi}_B]_-^+\\ &+\sigma\hat{h}_S\bar{C}[\bar{\phi}_A+\bar{\phi}_B]_-^+ = \frac{1}{\delta_C^2}\left[\bar{\chi}\left(\hat{h}_S\frac{d^2\bar{C}}{dz^2}+\frac{d\hat{C}}{dz}\right)+\left(\hat{h}_S\frac{d\bar{\chi}}{dz}+\hat{\chi}\right)\frac{d\bar{C}}{dz}\right]_-^+\end{aligned}\right\} \tag{3.17c}$$

$$\left[\hat{h}_S\frac{d\bar{B}}{dz}+\hat{B}\right]_-^+ = 0 \tag{3.17d}$$

$$\left[\hat{h}_S\frac{d\bar{C}}{dz}+\hat{C}\right]_-^+ = 0 \tag{3.17e}$$

$$\hat{h}_S\frac{d\bar{B}}{dz}+\hat{B} = -\frac{1}{M_B^{\mathcal{C}}}\left(\hat{h}_S\frac{d\bar{T}}{dz}+\hat{T}\right) \tag{3.17f}$$

$$\hat{h}_S\frac{d\bar{C}}{dz}+\hat{C} = -\frac{1}{M_C^{\mathcal{C}}}\left(\hat{h}_S\frac{d\bar{T}}{dz}+\hat{T}\right) \tag{3.17g}$$

$$M_B^{\mathcal{C}}\left(\hat{h}_S\frac{d^2\bar{B}}{dz^2}+\frac{d\hat{B}}{dz}\right)\Bigg|^+ = M_C^{\mathcal{C}}\left(\hat{h}_S\frac{d^2\bar{C}}{dz^2}+\frac{d\hat{C}}{dz}\right)\Bigg|^+ \tag{3.17h}$$

$$-\alpha\hat{\psi}\Big|_{-}^{+} = 0 \tag{3.17i}$$

$$\hat{\psi}_z|_{-}^{+} = 0 \tag{3.17j}$$

In the secondary mushy layer, the linearized equations are

$$-\alpha^2\hat{\psi} + \frac{d^2\hat{\psi}}{dz^2} = \frac{1}{\Pi}\left(\frac{d\Pi}{d\bar{\chi}}\frac{d\bar{\chi}}{dz}\frac{d\hat{\psi}}{dz}\right) - \Pi\alpha(Ra\hat{T} + Ra_B\hat{B} + Ra_C\hat{C}) \tag{3.18a}$$

$$\left.\begin{aligned}\sigma\bar{\chi}\hat{B} + \sigma\bar{B}\hat{\chi} - \frac{d\bar{\chi}}{dz}\hat{B} - \bar{\chi}\frac{d\hat{B}}{dz} - \frac{d\hat{\chi}}{dz}\bar{B} - \frac{d\bar{B}}{dz}\hat{\chi} + \sigma\hat{\phi_B} - \frac{d\hat{\phi_B}}{dz} \\ -\alpha\frac{d\bar{B}}{dz}\hat{\psi} = \frac{1}{\delta_B^2}\left(-\alpha^2\hat{B}\bar{\chi} + \frac{d\bar{\chi}}{dz}\frac{d\hat{B}}{dz} + \bar{\chi}\frac{d^2\hat{B}}{dz^2} + \frac{d\hat{\chi}}{dz}\frac{d\bar{B}}{dz} + \hat{\chi}\frac{d^2\bar{B}}{dz^2}\right)\end{aligned}\right\} \tag{3.18b}$$

$$\left.\begin{aligned}\sigma\bar{\chi}\hat{C} + \sigma\bar{C}\hat{\chi} - \frac{d\bar{\chi}}{dz}\hat{C} - \frac{d\hat{C}}{dz}\bar{\chi} - \frac{d\hat{\chi}}{dz}\bar{C} - \frac{d\bar{C}}{dz}\hat{\chi} - \alpha\hat{\psi}\frac{d\bar{C}}{dz} \\ = \frac{1}{\delta_C^2}\left(-\alpha^2\bar{\chi}\hat{C} + \bar{\chi}\frac{d^2\hat{C}}{dz} + \frac{d\bar{\chi}}{dz}\frac{d\hat{C}}{dz} + \frac{d^2\bar{C}}{dz^2}\hat{\chi} + \frac{d\bar{C}}{dz}\frac{d\hat{\chi}}{dz}\right)\end{aligned}\right\} \tag{3.18c}$$

$$\sigma\hat{T} - \frac{d\hat{T}}{dz} - S\left(-\sigma\hat{\chi} + \frac{d\hat{\chi}}{dz}\right) - \alpha\hat{\psi}\frac{d\bar{T}}{dz} = -\alpha^2\hat{T} + \frac{d^2\hat{T}}{dz^2} \tag{3.18d}$$

$$\hat{B} = -\frac{1}{M_B^{\mathcal{C}}}\hat{T} \tag{3.18e}$$

$$\hat{C} = -\frac{1}{M_C^{\mathcal{C}}}\hat{T} \tag{3.18f}$$

At $z = 0$,

$$\hat{\psi} = 0 \tag{3.19a}$$

$$\hat{C} = 0. \tag{3.19b}$$

3.2.3 Solution Method

To determine the unknown z-dependence of the perturbations, a Chebyshev pseudospectral collocation method is used. This same method was successfully used in [3] for similar purposes. Additional Details of this method can be found in [6]. This particular method solves the generalized eigenvalue problem

$$A\vec{y} = \sigma B\vec{y} \tag{3.20}$$

where A and B are matrices, $\vec{y}$ is the eigenvector of unknown, z-dependent perturbations, and σ is the associated eigenvalue whose sign indicates the perturbations' growth over time. The matrix A can be thought of as a 3-by-3 matrix comprised of the submatrices ALL, ALP, ALS, APL, APP, APS, ASL, ASP and ASS. Here, L represents the liquid layer, P represents the primary mushy layer and S represents the secondary mushy layer. Matrix A therefore has the form

$$A = \begin{bmatrix} ALL & ALP & ALS \\ APL & APP & APS \\ ASL & ASP & ASS \end{bmatrix}.$$

The three rows of sub-blocks contain the governing equations specified in each of the liquid layer, the primary mushy layer and the secondary mushy layer respectively. The three columns of sub-blocks represent the liquid variables: B, C and T, the primary mush variables: B, C, T and χ and the secondary mush variables: B, C, T, χ and ϕ_B. For instance, submatrix ALL encodes the governing equations for the liquid layer involving liquid variables. Submatrix ALP contains the liquid equations but only those terms which use primary mushy layer variables. The size of each submatrix depends on the number of discrete Chebyshev points used for the given layer and the number of variables in the layer. In the liquid, primary mush and secondary mush, NL, NP and NS Chebyshev points are used, respectively, for each variable of the layer. This means, for example, that the submatrix

ALL has $3(NL+1)$ rows and $3(NL+1)$ columns which store the various equations and variables. The dimensions of the other eight submatrices can be found in the same way. In addition, boundary conditions are maintained by rows of the A matrix that separate the three rows of submatrices. So, for example, boundary conditions enforced at the top of the system ($z = H$) appear in the first few rows of the matrix A. The particular row of A in which a given boundary condition appears depends on the boundary at which the condition is to be enforced.

The structure of the B matrix from the right hand side of equation (3.20) is similar to that of A. The matrix B contains the nine submatricies BLL, BLP, BLS, BPL, BPP, BPS, BSL, BSP and BSS and looks like

$$B = \begin{bmatrix} BLL & BLP & BLS \\ BPL & BPP & BPS \\ BSL & BSP & BSS \end{bmatrix}.$$

The entries of the B matrix are the coefficients of the terms in the governing equations that involve σ. Because of this, many of the elements of B are 0. The dimensions of A and B are equal. Matlab code showing the details of the matrices A and B is given in Appendix A.

To solve for the set of all $\vec{y}$ and σ, an eigensolver such as Matlab's `eig` command can be used. An eigenvector $\vec{y}$ is a column vector containing the perturbations' z-dependence for each variable in all three layers. The perturbations are evaluated at NL, NP and NS points, the number of Chebyshev points within the liquid, primary mush and secondary mush, respectively. The number of Chebyshev points used in each layer need not be the same.

Since σ appears in the exponent of the additive perturbation terms, its sign determines the overall stability of the base state solution presented previously. If for a given wavenumber α and values for the solutal Rayleigh numbers Ra_B and Ra_C and thermal Rayleigh

number Ra, there exists a σ such that real(σ)$>$ 0, then the basic, steady state solution is linearly unstable. If, on the other hand, real(σ)$<$ 0 for all σ, then the basic solution is linearly stable. Therefore, it is useful to look at ordered pairs (α, RN) such that the largest real part of any σ value is sufficiently close to 0. Such points (α, RN) will be points along a neutral stability curve. Here, α is a wavenumber characterizing the normal mode perturbation and RN is one of the thermal or solutal Rayleigh numbers. The other two Rayleigh numbers can be assumed to be fixed constant values.

Chapter 4: Linear Stability Results

The linear stability of the basic solution presented earlier is determined for various system parameter values. More specifically, an investigation of the basic solution's linear stability is presented for different combinations of values for the thermal Rayleigh number Ra and the solutal Rayleigh numbers Ra_B and Ra_C. In each case, two of the Rayleigh numbers are set to 0 while the third Rayleigh number is allowed to vary. Doing this produces neutral stability curves which identify, for any wave number α, a Rayleigh number that, beyond which, the basic solution becomes linearly unstable.

To obtain the results shown in figures 4.1, 4.2, 4.3, 4.4 and 4.5, a numerical bisection routine was used to solve for a value of the nonzero Rayleigh number such that the largest real part of any eigenvalue σ was sufficiently close to 0. Doing this for each wavenumber α in the range of wavenumbers considered produces the curves seen in each of the figures. These curves divide the plane into regions such that the basic solution is predicted to be linearly stable on one side of the curve and linearly unstable on the other side of the curve. The basic solution is neutrally stable for combinations of α and the nonzero Rayleigh number that lie on the curve. If a curve attains a minimum or maximum value over the range of wavenumbers considered, the Rayleigh number where the extremum occurs is called a critical Rayleigh number. The corresponding wavenumber at which this occurs is called a critical wavenumber. A critical Rayleigh number represents a value below which the basic solution is linearly stable for all wavenumbers if the critical point is a minimum and above which if the critical Rayleigh number is a maximum. With the exception of figure 4.4, all figures were created using $NL = NP = NS = 32$. Producing the curve in figure 4.4

Table 4.1: Parameter values used during linear stability analysis based on those found in [2]

Parameter	Value
H	2.0
B_∞	0.2
C_∞	0.1
T_∞	5
B_E	1/3
T_E	-19
C_E	1/3
B_E^{AB}	0.5
T_E^{AB}	-5
δ_b^2	100
δ_c^2	100
γ	-1
T_M	0
Da	0.05

required that $NL = NP = NS = 16$.

Beginning with the neutral stability curve for the case of $Ra_B = Ra_C = 0$ in figure 4.1, this curve separates a region below the curve where the basic state is predicted to be linearly stable from a region above the curve where the basic state is predicted to be linearly unstable. This curve has a minimum near $\alpha = 1.2$. This wavenumber is denoted as α_{crit}. The associated critical Rayleigh number, Ra_{crit}, at this wavenumber is about $Ra_{crit} = 0.4$. Also shown in the inset plot of the figure is the solution for the streamfunction $\hat{\psi}$ at the point $(\alpha_{crit}, Ra_{crit})$. The streamlines show the overall motion of the fluid throughout the three layers. The dashed lines in the inset plot represent the positions of the interfaces separating the liquid from the primary mush and the primary mush from the secondary mush.

The next case has $Ra = Ra_C = 0$. In this case, two neutral stability curves can be found. One curve, shown in figure 4.2, lies completely above the line $Ra_B = 0$ while the

second curve, shown in figure 4.3, lies completely below the line $Ra_B = 0$. Although they are plotted separately, from figures 4.2 and 4.3 it is apparent that the two curves separate the (α, Ra_B)-plane into three regions. The area above the upper neutral curve of figure 4.2 represents a region of (α, Ra_B) pairs such that the base state is linearly unstable. The area between the two neutral curves represents combinations of α and Ra_B such that the base state is linearly stable. The area below the lower neutral curve in figure 4.3 represents a second region of (α, Ra_B) coordinates in which the base state solution is linearly unstable. Often, only a single curve separates a single stable region from an unstable region. This is the case for some binary systems such as the one described in [9] or ternary systems where diffusion is not included [3]. In this case however, there are clearly two unstable regions and one stable region. This means that the basic solution is unstable for a specific wavenumber α whenever the solutal Rayleigh number Ra_B is either sufficiently positive *or* sufficiently negative.

The presence of two neutral curves is due in part to the derivative of the liquid composition B profile $\frac{dB}{dz}$. From the base state solution, it can be seen that, in the primary mushy layer, both $\frac{dB}{dz} > 0$ and $\frac{dB}{dz} < 0$. Since $Ra = Ra_C = 0$, the derivative of the density ρ looks like $\frac{d\rho}{dz} \sim Ra_B \frac{dB}{dz}$. Whenever $\frac{dB}{dz} > 0$ and $Ra_B > 0$, $\frac{d\rho}{dz} > 0$. This then sets up a situation where denser, and therefore heavier fluid is sitting on top of less dense or lighter fluid. Similarly, the base state can become unstable whenever $\frac{dB}{dz} < 0$ and $Ra_B < 0$. This also has the effect of producing a situation where fluid density is increasing as z increases.

The neutral stability curve in figure 4.2 has two local minima. The first of these occurs near $\alpha = 1.2$ while the second occurs near $\alpha = 29.2$. Once again, the solutions to the streamfunction $\hat{\psi}$ at the local minima are shown in the insets. When α is near 1.2, the fluid motion is present through each of the three layers in large convection cells. In contrast, at the second local minima near $\alpha = 29.2$, the convection is locallized in the secondary mushy

layer in much more compact convection cells. The neutral stability curve of figure 4.3 has a maximum near $\alpha = 10.3$. The fluid flow at this maximum is localized mostly within the primary mushy layer as seen in the inset.

When $Ra = Ra_B = 0$, two neutral stability curves can again be found. These two curves are shown in figures 4.4 and 4.5. In figure 4.4, the neutral stability curve lies above the line $Ra_C = 0$. This curve has a minimum near $\alpha = 1.2$. The inset plot shows the fluid flow for this critical wavenumber. The fluid flow originates in the liquid layer and occurs within large convection cells. The corresponding critical Rayleigh number is approximately $Ra_{crit} = 95.8$. The region above this curve shows pairs of (α, Ra_C) where the basic solution is predicted to be linearly unstable. Below this curve, but above the curve in figure 4.5, the base state solution is predicted to be linearly stable.

Finally, the shape neutral stability plot in figure 4.5 again has $Ra = Ra_B = 0$ and is very similar to the curve in figure 4.3. This curve lies completely below the line $Ra_C = 0$ for the range of wavenumbers considered. The curve in figure 4.5 has a maximum at $\alpha = 10.2$. The solution for the streamfunction in the inset shows the convection is localized in the primary mushy layer under these conditions.

In the case of $Ra = Ra_C = 0$, a solutal buoyancy argument can be made explaining the existence of two neutral curves. This is due to the change in the sign of the derivative $\frac{dB}{dz}$ within the primary mushy layer. When $Ra = Ra_B = 0$, two neutral curves are also present. However, $\frac{dC}{dz} < 0$ causing the system to be (solutally) stably stratified and does not change sign at any point. This explains the lower neutral curve in figure 4.5 since $\frac{d\rho}{dz} \sim Ra_C \frac{dC}{dz} > 0$ whenever $Ra_C < 0$. This argument fails to explain though why sufficiently large, positive values of Ra_C cause the system to become unstable. A different mechanism then must be causing the instability for $Ra_C > 0$. Based on recent work of Anderson, Coriell, McFadden and Murray (Private communication, 2009) on a similar model, such modes appear to be possible in this type of ternary system and warrant further investigation.

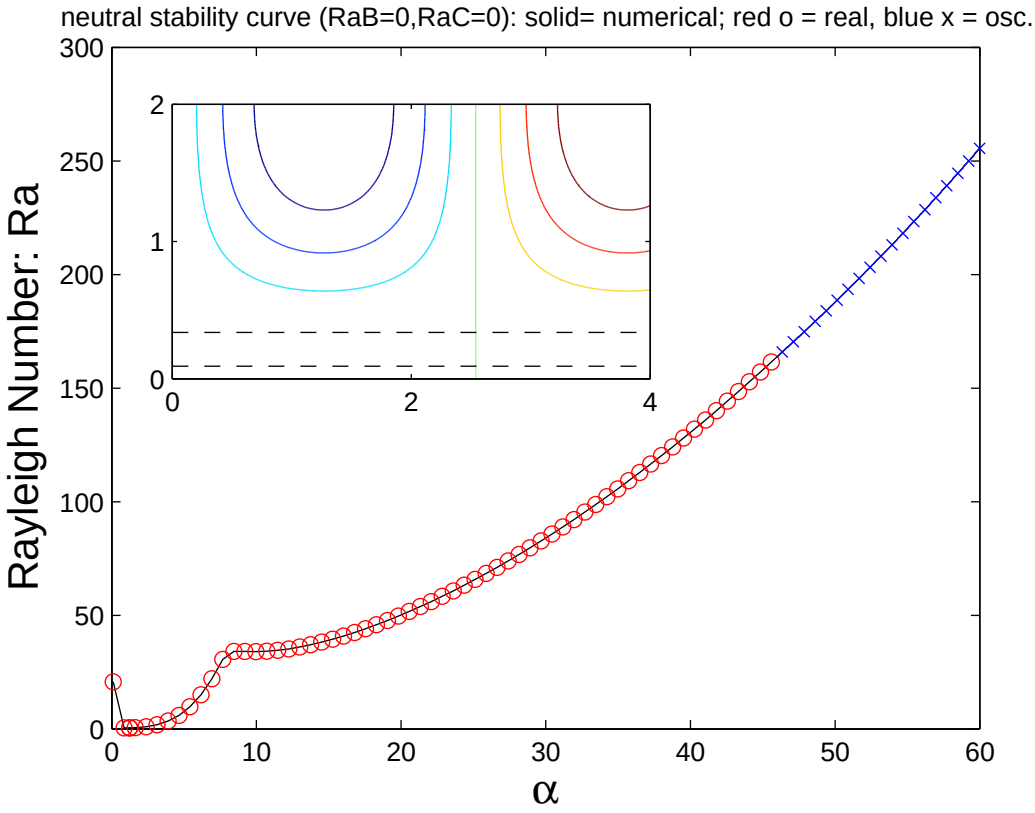

Figure 4.1: Neutral stability curve for the case of $Ra_B = Ra_C = 0$.

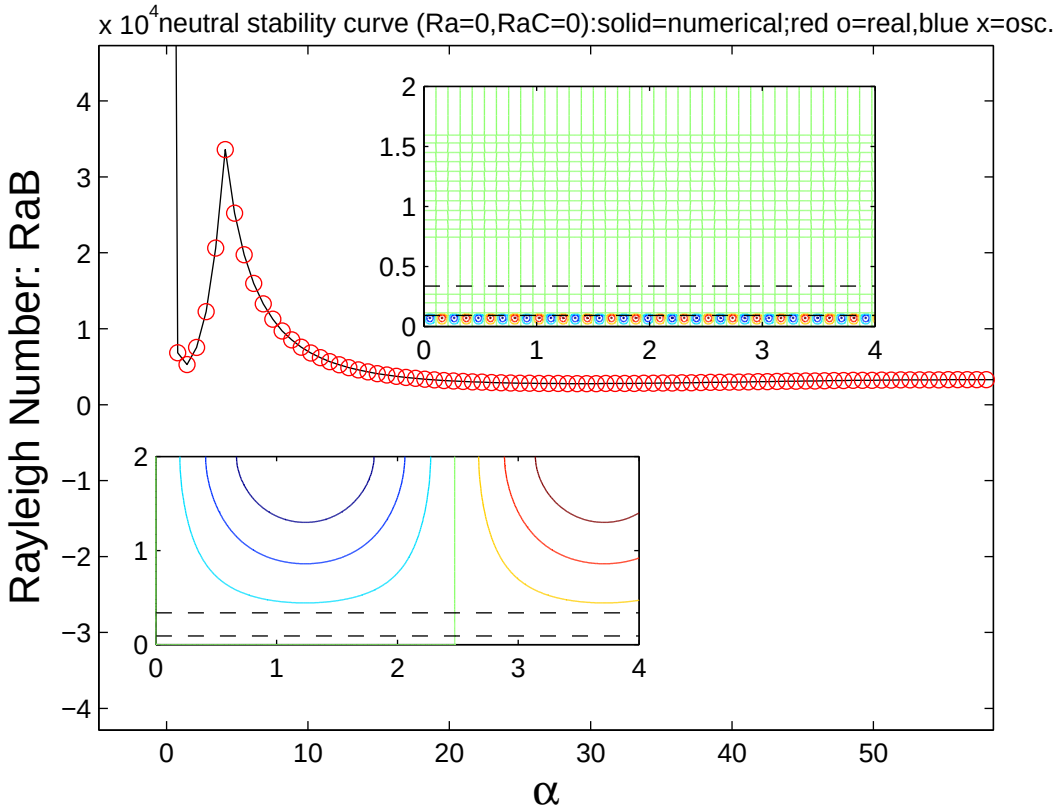

Figure 4.2: Neutral stability curve for the case of $Ra = Ra_C = 0$.

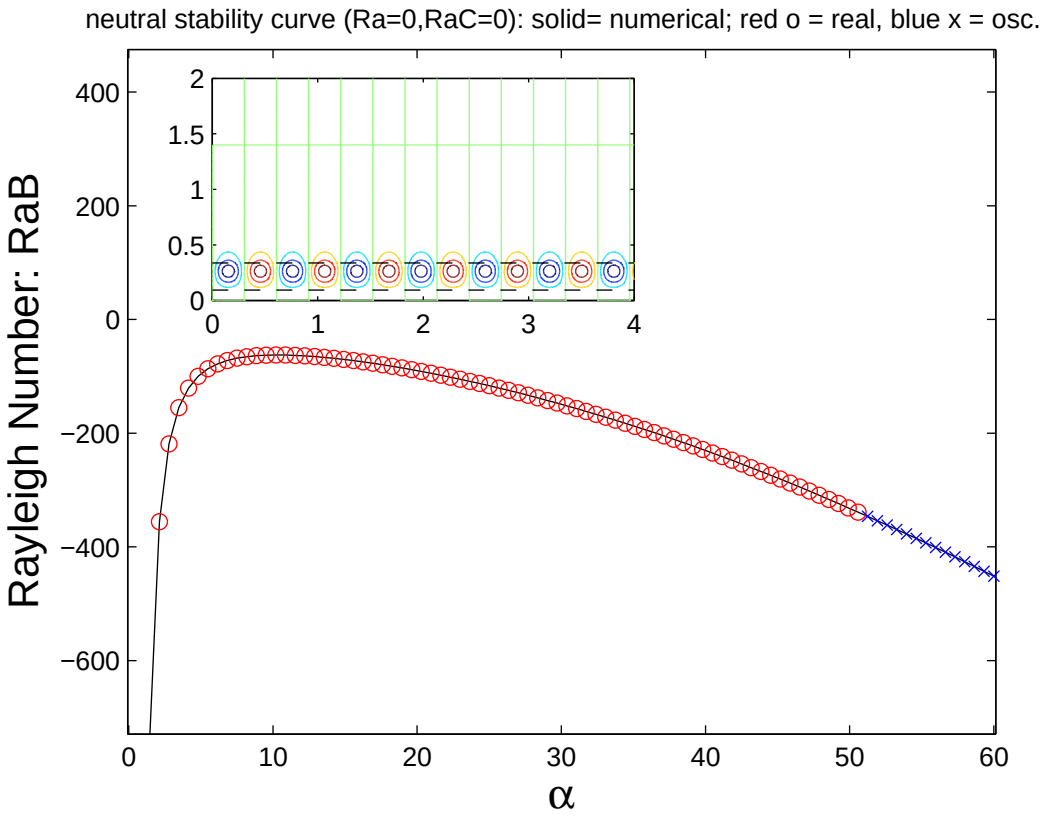

Figure 4.3: Neutral stability curve for the case of $Ra = Ra_C = 0$.

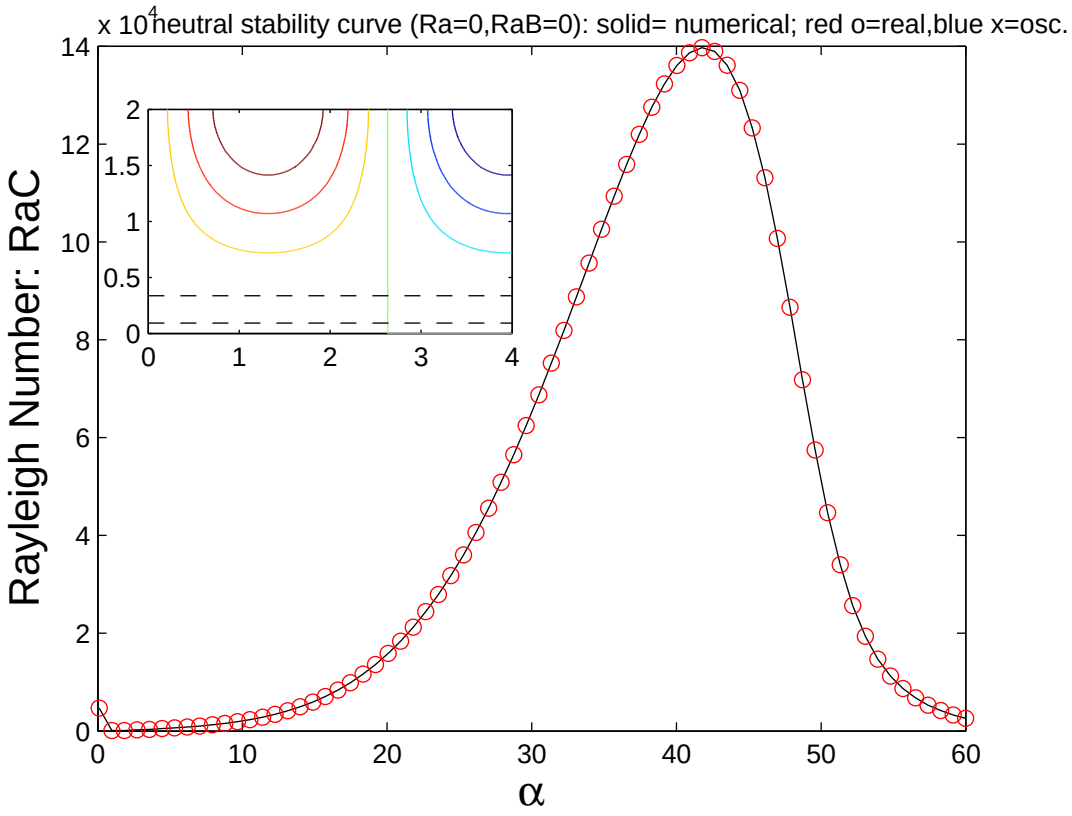

Figure 4.4: Neutral stability curve for the case of $Ra = Ra_B = 0$.

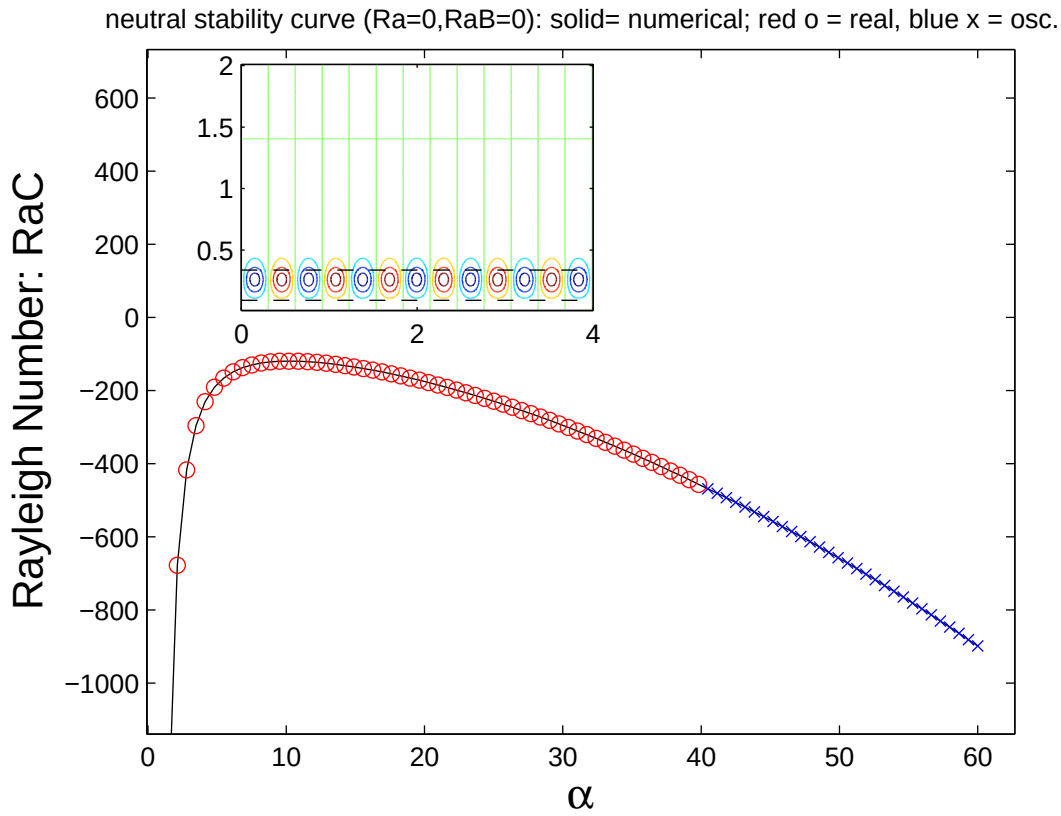

Figure 4.5: Neutral stability curve for the case of $Ra = Ra_B = 0$.

Chapter 5: Conclusion

This work has made a number of contributions to the area of solidifying ternary alloys. In Section 2.2 a new model is presented which builds upon work done in [2] and [3]. This new model incorporates fluid flow, diffusion of heat as well as diffusion of species. Since convection and diffusion can carry heat and solute between layers, it is important to consider these effects simultaneously.

Finding an equilibrium or steady state of this system is done numerically. Once the governing equations and boundary conditions have been properly nondimensionalized and made independent of time and directions other than z, all quantities can be written in terms of just a few unknowns. These unknowns include the liquid composition B profile in the secondary mush and both B and C solute profiles in the primary mush. Additionally, the interface positions $z = h_P$ and $z = h_S$ must be solved for. The three solute profiles are found by numerically integrating the ODEs for these quantities through the appropriate layer and solving a system of three equations. The solution to this system is found using Matlab's `fsolve` command. Once found, all other properties of the base state in each of the layers are then known.

A base state solution for the enhanced model is presented for the set of input parameter values given. The base state shown in figure 3.1 has no fluid flow and temperature that is increasing in z. The liquid compositions of species B and C are nearly constant through the liquid layer. Within the primary mushy layer, the liquid compositions are decreasing with z. However, just ahead of the mush-mush interface, the derivative $\frac{dB}{dz}$ changes sign so that B is briefly increasing with z. In the secondary mushy layer, liquid composition C is again

decreasing with z while liquid composition B is increasing with z. This solution also has no solid fractions in the liquid layer, nonzero ϕ_A in the primary mushy layer and $\phi_A \neq 0 \neq \phi_B$ in the secondary mushy layer. In the eutectic solid layer, we have $\phi_A + \phi_B + \phi_C = 1$ and $\chi = 0$. The liquid-mush and mush-mush interface positions have also been solved for as part of the basic solution.

To determine the base state's linear stability to infinitesimal perturbations, a Chebyshev pseudospectral collocation method is used. This method is used to solve a generalized eigenvalue problem. The eigenfunctions found from this method represent the disturbances' z dependence through each of the layers. The eigenvalues represent the disturbances' growth rates as $t \to \infty$. Details of this particular method can be found in [6] as well as in Appendix A. This new numerical tool can be used to better understand ternary alloy mushy layer systems.

The linear stability of the basic solution has been determined for various system parameter values. More specifically, an investigation of the basic solution's linear stability has been presented for different combinations of values for the thermal Rayleigh number Ra and the two solutal Rayleigh numbers Ra_B and Ra_C. In each case, two of the Rayleigh numbers are set to 0 while the third Rayleigh number is allowed to vary. This produces neutral stability curves which identify, for any wave number α, a critical value for the nonzero Rayleigh number that, beyond which, the basic solution is linearly unstable.

Appendix A: Alphabetical Listing of Matlab Functions

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Bliquid.m
%
% Function to compute the liquid composition profile of B through the
% liquid layer.
%
% INUPTS: BP...liquid fraction of B at the liquid-mush interface
%
%         hP...postion of the liquid-mush interface
%
%         zLiq...vector of z (position) values in the liquid layer at
%                which to evaluate the temperature
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: B profile vector through the liquid layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function B = Bliquid(BP, hP, zLiq, paramVec)
```

```
% separate out phase diagram constants

deltabsq = paramVec(4);

Binfty = paramVec(6);


B = Binfty + (BP - Binfty) .* exp(deltabsq .* (hP - zLiq));
```

```
% This is a matlab m-file from Trefethen, "Spectral Methods in Matlab"
% page 54.  It returns a matrix D and a vector x containing the
% Chebyshev Differentiation matrix D and the Chebyshev grid points x.
%
%
function [D,x]=cheb(N)
if N==0, D=0; x=1; return, end
x=cos(pi*(0:N)/N)';
c=[2; ones(N-1,1); 2].*(-1).^(0:N)';
X=repmat(x,1,N+1);
dX=X-X';
D=(c*(1./c)')./(dX+(eye(N+1)));       % off-diagonal entries
D=D-diag(sum(D'));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% chiPrimary.m
%
% Function to compute the value of chi, the liquid-fraction, within the
% primary mushy layer.
%
% INPUTS: B.........liquid composition of B within the primary mushy layer
%
%         BP.........liquid composition of B at the liquid-mush interface
%
%         C.........liquid composition of C in the primary mushy layer
%
%         CP.........liquid composition of C at the liquid-mush interface
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: chi...liquid fraction within secondary mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function chi = chiPrimary(B, BP, C, CP, paramVec)


% separate out phase diagram constants
deltabsq = paramVec(4);
deltacsq = paramVec(5);
Binfty = paramVec(6);
```

```
Cinfty = paramVec(8);

gamma = paramVec(9);

mBbar = paramVec(18);

mCbar = paramVec(19);


b1 = (1/gamma)*(deltabsq*Binfty*mBbar+deltacsq*Cinfty*mCbar);


b2 = (1/gamma).*(mBbar.*(B-BP).*(1-deltabsq)+mCbar.*(C-CP).*(1-deltacsq));


chi = 1/2 .* (1 + b1 - b2+sqrt((1-b1) .^ 2 + b2 .* (-2 - 2 .* b1 + b2)));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% chiSecondary.m
%
% Function to compute the value of chi, the liquid fraction, within the
% secondary mushy layer for any given value(s) of B.
%
% No value is returned if computed value for chi is not within the range
% 0 <= chi <= 1. If chi is outside of this range an error message is
% printed and execution is halted.
%
% INPUTS: B...liquid composition of B within the secondary mushy layer
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: chi...liquid fraction within secondary mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function chi = chiSecondary(B, paramVec)


% separate out phase diagram constants
BE = paramVec(1);
TE = paramVec(2);
deltacsq = paramVec(5);
Tinfty = paramVec(7);
Cinfty = paramVec(8);
```

```
gamma = paramVec(9);

gammab = paramVec(10);

BEAB = paramVec(11);

mB = paramVec(14);

mC = paramVec(15);

mBC = paramVec(16);

mCC = paramVec(17);


% a Steffan number

S = (Tinfty - TE) / (gamma * (mB + mC));


% coefficient of chi^1 term in the quadratic equation for chi

b = (-B + BE) / gammab - S - 1 + (deltacsq / gammab) * (B - BEAB);


% constant term from quadratic equation for chi

c = -(deltacsq * mCC * Cinfty) / (gammab * mBC);


% use quadratic formula to find roots of chi (find plus root first)

chiplus = (-b + sqrt(b .* b - 4 .* c)) / 2;


% use minus to get second root

chiminus = (-b - sqrt(b .* b - 4 .* c)) / 2;


% somehow determine which root of chi to choose

if (chiplus >= 0 & chiplus <= 1)

    chi = chiplus;
```

```
elseif (chiminus >=0 & chiminus <= 1)
    chi = chiminus;
else
    error('ERROR: no valid value for chi found.');
end
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Cliquid.m
%
% Function to compute the liquid composition profile of C through the
% liquid layer.
%
% INUPTS: CP.....liquid composition of C at the liquid-mush interface
%
%         hP.....position of the liquid-mush interface
%
%         zLiq...vector of z (position) values in the liquid layer at
%                which to evaluate the temperature
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: liquid composition profile vector of C through the liquid layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function C = Cliquid(CP, hP, zLiq, paramVec)


% separate out phase diagram constants
deltacsq = paramVec(5);
Cinfty = paramVec(8);


C = Cinfty + (CP - Cinfty) .* exp(deltacsq .* (hP - zLiq));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Csecondary.m
%
% Function to compute the liquid composition profile of C through the
% secondary mushy layer.
%
% INUPTS: B.....liquid composition of B through the secondary mushy region
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: liquid composition profile vector of C through the secondary
%          mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function C = Csecondary(B, paramVec)


% separate out phase diagram constants
BE = paramVec(1);
TE = paramVec(2);
TEAB = paramVec(12);
mBC = paramVec(16);
mCC = paramVec(17);


C = (-mBC .* (B - BE) + TE - TEAB) ./ -mCC;
```

```
% this file called from pm_three_layer.m ... it plots results
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%          PROCESS AND PLOT RESULTS          %%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(2)
RaC_rec=Ra_rec;
plot(alpha_rec,RaC_rec,'k-');hold on
xlabel('\alpha','FontSize',16);ylabel('Rayleigh Number: RaC','FontSize',16);
title('neutral stability curve (Ra=0,RaB=0): solid= numerical; red o = '...
    'real, blue x = osc.')
%
alpha_length=length(alpha_rec);
for ja=1:alpha_length
    if abs(omI_rec(ja)) >  0
       plot(alpha_rec(ja),RaC_rec(ja),'bx')
    else
       plot(alpha_rec(ja),RaC_rec(ja),'ro')
    end
end
%
xsave_neutral=[alpha_rec' RaC_rec' omR_rec' omI_rec'];
save 'R_vs_a_dat.m' xsave_neutral -ascii -double
print -depsc pm_neutral.eps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
figure(3)
plot(alpha_rec,omR_rec,'r-',alpha_rec,omI_rec,'g--')
xlabel('\alpha','FontSize',16);ylabel('real and imaginary \sigma',...
    'FontSize',16)
title('Real (red) and Imaginary (green) growth rates: neutral stability')
print -deps pm_sig_v_alpha_neutral.eps
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% plot the computed eigenvalue spectrum
%
figure(10)
plot(real(diag(D)),imag(diag(D)),'rs');hold on;
title('Computed eigenvalues')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(4)   %  liquid/primary/secondary
%%%% helpful indices
liqtot=4*(NL+1);
pritot=5*(NP+1)+1;
%
jSL = 1:NL+1;
jBL = NL+2:2*(NL+1);
jCL = 2*(NL+1)+1:3*(NL+1);
jTL = 3*(NL+1)+1:4*(NL+1);
%
jSP = (2:NP+2) + liqtot;
jBP=(NP+3:2*(NP+1)+1) + liqtot;
```

```
jCP=(2*(NP+1)+2:3*(NP+1)+1) + liqtot;

jTP=(3*(NP+1)+2:4*(NP+1)+1) + liqtot;

jXP=(4*(NP+1)+2:5*(NP+1)+1) + liqtot;

%

jSS=(2:NS+2)+liqtot+pritot;

jBS=(NS+3:2*(NS+1)+1)+liqtot+pritot;

jCS=(2*(NS+1)+2:3*(NS+1)+1)+liqtot+pritot;

jTS=(3*(NS+1)+2:4*(NS+1)+1)+liqtot+pritot;

jXS=(4*(NS+1)+2:5*(NS+1)+1)+liqtot+pritot;

jPS=(5*(NS+1)+2:6*(NS+1)+1)+liqtot+pritot;

%

%%%%%%%%plot the temperature perturbation

subplot(3,2,1)

tl_vec=V(jTL,Kval);

tp_vec=V(jTP,Kval);

ts_vec=V(jTS,Kval);

%ts_vec=V(tsstart:tsstop,Kval);

plot(zl',real(tl_vec),zl',imag(tl_vec),'--',...

     zp',real(tp_vec),zp',imag(tp_vec),'--',...

     zs',real(ts_vec),zs',imag(ts_vec),'--');

xlabel('z','FontSize',16);ylabel('T','FontSize',16)

title('At final wavenumber','FontSize',16)

%

%%%%%%%%plot the liquid B perturbation

subplot(3,2,2)

bl_vec=V(jBL,Kval);
```

```
bp_vec=V(jBP,Kval);
bs_vec=V(jBS,Kval);
plot(zl',real(bl_vec),zl',imag(bl_vec),'--',...
     zp',real(bp_vec),zp',imag(bp_vec),'--',...
     zs',real(bs_vec),zs',imag(bs_vec),'--');
xlabel('z','FontSize',16);ylabel('B','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%plot the liquid C perturbation
subplot(3,2,3)
cl_vec=V(jCL,Kval);
cp_vec=V(jCP,Kval);
cs_vec=V(jCS,Kval);
plot(zl',real(cl_vec),zl',imag(cl_vec),'--',...
     zp',real(cp_vec),zp',imag(cp_vec),'--',...
     zs',real(cs_vec),zs',imag(cs_vec),'--');
xlabel('z','FontSize',16);ylabel('C','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%plot the liquid streamfunction perturbation
subplot(3,2,4)
sl_vec=V(jSL,Kval);
sp_vec=V(jSP,Kval);
ss_vec=V(jSS,Kval);
plot(zl',real(sl_vec),zl',imag(sl_vec),'--',...
     zp',real(sp_vec),zp',imag(sp_vec),'--',...
```

```
     zs',real(ss_vec),zs',imag(ss_vec),'--');
xlabel('z','FontSize',16);ylabel('\psi','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%%plot the liquid fraction perturbation
subplot(3,2,5)
xp_vec=V(jSP,Kval);
xs_vec=V(jSS,Kval);
plot(zp',real(xp_vec),zp',imag(xp_vec),'--',...
     zs',real(xs_vec),zs',imag(xs_vec),'--');
xlabel('z','FontSize',16);ylabel('\chi','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%%plot the solid fraction B perturbation
subplot(3,2,6)
ps_vec=V(jPS,Kval);
plot(zs',real(ps_vec),zs',imag(ps_vec),'--');
xlabel('z','FontSize',16);ylabel('\phi_B','FontSize',16)
%title('At final wavenumber','FontSize',16)

%
print -deps pm_pert_final.eps
```

```
% this file called from pm_three_layer.m ... it plots eigenfunctions
% at the critical Rayleigh number and wavenumber
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%          PROCESS AND PLOT RESULTS          %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure(8)
%%%% helpful indices
liqtot=4*(NL+1);
pritot=5*(NP+1)+1;
%
jSL = 1:NL+1;
jBL = NL+2:2*(NL+1);
jCL = 2*(NL+1)+1:3*(NL+1);
jTL = 3*(NL+1)+1:4*(NL+1);
%
jSP = (2:NP+2) + liqtot;
jBP=(NP+3:2*(NP+1)+1) + liqtot;
jCP=(2*(NP+1)+2:3*(NP+1)+1) + liqtot;
jTP=(3*(NP+1)+2:4*(NP+1)+1) + liqtot;
jXP=(4*(NP+1)+2:5*(NP+1)+1) + liqtot;
%
jSS=(2:NS+2)+liqtot+pritot;
jBS=(NS+3:2*(NS+1)+1)+liqtot+pritot;
jCS=(2*(NS+1)+2:3*(NS+1)+1)+liqtot+pritot;
```

```
jTS=(3*(NS+1)+2:4*(NS+1)+1)+liqtot+pritot;
jXS=(4*(NS+1)+2:5*(NS+1)+1)+liqtot+pritot;
jPS=(5*(NS+1)+2:6*(NS+1)+1)+liqtot+pritot;
%
%%%%%%%%plot the temperature perturbation
subplot(3,2,1)
tl_vec=V(jTL,KvalATCRIT);
tp_vec=V(jTP,KvalATCRIT);
ts_vec=V(jTS,KvalATCRIT);
%ts_vec=V(tsstart:tsstop,KvalATCRIT);
plot(zl',real(tl_vec),zl',imag(tl_vec),'--',...
     zp',real(tp_vec),zp',imag(tp_vec),'--',...
     zs',real(ts_vec),zs',imag(ts_vec),'--');
xlabel('z','FontSize',16);ylabel('T','FontSize',16)
title('At critical wavenumber','FontSize',16)
%
%%%%%%%%plot the liquid B perturbation
subplot(3,2,2)
bl_vec=V(jBL,KvalATCRIT);
bp_vec=V(jBP,KvalATCRIT);
bs_vec=V(jBS,KvalATCRIT);
plot(zl',real(bl_vec),zl',imag(bl_vec),'--',...
     zp',real(bp_vec),zp',imag(bp_vec),'--',...
     zs',real(bs_vec),zs',imag(bs_vec),'--');
xlabel('z','FontSize',16);ylabel('B','FontSize',16)
%title('At final wavenumber','FontSize',16)
```

```
%
%%%%%%%%%plot the liquid C perturbation
subplot(3,2,3)
cl_vec=V(jCL,KvalATCRIT);
cp_vec=V(jCP,KvalATCRIT);
cs_vec=V(jCS,KvalATCRIT);
plot(zl',real(cl_vec),zl',imag(cl_vec),'--',...
     zp',real(cp_vec),zp',imag(cp_vec),'--',...
     zs',real(cs_vec),zs',imag(cs_vec),'--');
xlabel('z','FontSize',16);ylabel('C','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%%plot the liquid streamfunction perturbation
subplot(3,2,4)
sl_vec=V(jSL,KvalATCRIT);
sp_vec=V(jSP,KvalATCRIT);
ss_vec=V(jSS,KvalATCRIT);
plot(zl',real(sl_vec),zl',imag(sl_vec),'--',...
     zp',real(sp_vec),zp',imag(sp_vec),'--',...
     zs',real(ss_vec),zs',imag(ss_vec),'--');
xlabel('z','FontSize',16);ylabel('\psi','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%%plot the liquid fraction perturbation
subplot(3,2,5)
xp_vec=V(jSP,KvalATCRIT);
```

```
xs_vec=V(jSS,KvalATCRIT);
plot(zp',real(xp_vec),zp',imag(xp_vec),'--',...
     zs',real(xs_vec),zs',imag(xs_vec),'--');
xlabel('z','FontSize',16);ylabel('\chi','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
%%%%%%%%plot the solid fraction B perturbation
subplot(3,2,6)
ps_vec=V(jPS,KvalATCRIT);
plot(zs',real(ps_vec),zs',imag(ps_vec),'--');
xlabel('z','FontSize',16);ylabel('\phi_B','FontSize',16)
%title('At final wavenumber','FontSize',16)
%
print -deps pm_pertATCRIT.eps
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% findUnknowns.m
%
% Function to determine the unknown quantities hS, hP and BP. These
% correspond to positions of the interfaces at the top of the secondary
% mushy layer, the top of the primary mushy layer and the concentation of
% component B at the top of the primary mushy layer, respectively.
%
% We'll find the values for hS, hP and BP which zero the nonlinear system
% created at the end of this file by using fsolve.
%
% INPUTS: guess...vector of guesses for the unknown quantities [hS,hP,BP]
%
% OUTPUTS: f...vector of differences we're trying to zero
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function f = findUnknowns(guess, paramVec)


% guess(1) = hSguess
% guess(2) = hPguess
% guess(3) = BPguess


% separate out phase diagram constants
BE = paramVec(1);
TE = paramVec(2);
```

```
CE = paramVec(3);

deltabsq = paramVec(4);

deltacsq = paramVec(5);

Binfty = paramVec(6);

Tinfty = paramVec(7);

Cinfty = paramVec(8);

gamma = paramVec(9);

gammab = paramVec(10);

BEAB = paramVec(11);

TEAB = paramVec(12);

TM = paramVec(13);

mB = paramVec(14);

mC = paramVec(15);

mBC = paramVec(16);

mCC = paramVec(17);

mBbar = paramVec(18);

mCbar = paramVec(19);


% modify the relative and absolute error tolerances for ode23

options = odeset('RelTol',1e-5,'AbsTol',1e-8);


% solve ODE for B in the secondary mushy layer (solution is found from the

% bottom of the layer to the top)

[zSec, Bsec] = ode23(@rhsBodeSec, [0 guess(1)], BE, options, paramVec);


[nrows, ncols] = size(Bsec);
```

```
% read out the last entry from the B values (BS)
BScomp = Bsec(nrows);

% using BS, next find CS
CScomp = (mBC / mCC) * (BScomp - BEAB);

deltaTB = 1/((TM-Tinfty)/mB+Binfty+(mCbar/mBbar)*Cinfty);

deltaTC = 1/((TM-Tinfty)/mC+Cinfty+(mBbar/mCbar)*Binfty);

% using guessed value for BP, find CP
CPguess = (-deltaTB*(deltabsq-1)*(guess(3)-Binfty))/(deltaTC*(deltacsq-1)) ...
    + Cinfty + 1/(deltaTC*(deltacsq-1));

% solve ODEs for B and C in the primary mushy layer (again the solution is
% found from the bottom of the layer to the top)
[zPri, BCvalsPri] = ode23(@rhsBCodePri, [guess(1), guess(2)], ...
    [BScomp, CScomp], options, guess(3), CPguess, paramVec);

[nrows, ncols] = size(BCvalsPri);

% read out last value entry from the B values (this is BP)
BPcomp = BCvalsPri(nrows, 1);

% read out last entry from the the C values (this is CP)
```

```
CPcomp = BCvalsPri(nrows, ncols);


% determine chi (liquid-fraction) on the primary side of the interface
% between the two mushy layers
chiSplus = chiPrimary(BScomp,guess(3),CScomp,CPguess,paramVec);


% set up system of nonlinear equations to solve
f(1) = BScomp * (1 - deltacsq/deltabsq) + BEAB * (deltacsq/deltabsq) ...
    - (1/chiSplus) * (Binfty - (mCC/mBC) * (deltacsq/deltabsq) * Cinfty);


f(2) = BPcomp - guess(3);


f(3) = CPcomp - CPguess;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% getABtern3layer.m
%
% Function that returns the A and B matrices for the generalized eigenvalue
% problem A*y = sigma*B*y.
%
% INPUTS:
%
% OUTPUTS: A...matrix from lefthand side of generalized eigenvalue problem
%
%          B...matrix from righthand side of generalized eigenvalue problem
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [A,B] = getABtern3layer(alpha,RaB,RaC,Ra,Da,dBdzBL,deltaB2,...
    deltaC2,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
    dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
    dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
    d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,V,...
    chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
    CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PBS,DS2,...
    chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
    XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
    d2BdzBSdiag)

% define matrix of liquid equations using liquid variables
ALL=zeros(4*(NL+1),4*(NL+1));
```

```
% define a matrix of liquid equations using primary variables
ALP=zeros(4*(NL+1), 5*(NP+1)+1);

% define a matrix of liquid equations using secondary variables
ALS=zeros(4*(NL+1),6*(NS+1)+1);

% define matrix of primary equations using liquid variables
APL=zeros(5*(NP+1)+1,4*(NL+1));

% define matrix of primary equations using primary variables
APP=zeros(5*(NP+1)+1,5*(NP+1)+1);

% define matrix of primary equations using secondary variables
APS=zeros(5*(NP+1)+1,6*(NS+1)+1);

% define matrix of secondary equations using liquid variables
ASL=zeros(6*(NS+1)+1,4*(NL+1));

% define matrix of secondary equations using primary variables
ASP=zeros(6*(NS+1)+1,5*(NP+1)+1);

% define matrix of secondary equations using secondary variables
ASS=zeros(6*(NS+1)+1,6*(NS+1)+1);

% define matrix of liquid equations using liquid variables
```

```
BLL=zeros(4*(NL+1),4*(NL+1));

% define a matrix of liquid equations using primary variables
BLP=zeros(4*(NL+1), 5*(NP+1)+1);

% define a matrix of liquid equations using secondary variables
BLS=zeros(4*(NL+1),6*(NS+1)+1);

% define matrix of primary equations using liquid variables
BPL=zeros(5*(NP+1)+1,4*(NL+1));

% define matrix of primary equations using primary variables
BPP=zeros(5*(NP+1)+1,5*(NP+1)+1);

% define matrix of primary equations using secondary variables
BPS=zeros(5*(NP+1)+1,6*(NS+1)+1);

% define matrix of secondary equations using liquid variables
BSL=zeros(6*(NS+1)+1,4*(NL+1));

% define matrix of secondary equations using primary variables
BSP=zeros(6*(NS+1)+1,5*(NP+1)+1);

% define matrix of secondary equations using secondary variables
BSS=zeros(6*(NS+1)+1,6*(NS+1)+1);
```

```
% create variables for multi-row and multi-column indexing for assigning
% values to interior of layers (i denotes a row index and j a column index)
% (L=liquid, P=primary and S=secondary)
iSL = 3:NL-1;
iBL = NL+3:2*(NL+1)-1;
iCL = 2*(NL+1)+2:3*(NL+1)-1;
iTL = 3*(NL+1)+2:4*(NL+1)-1;

iSP=3:NP+1;
iBP=NP+4:2*(NP+1);
iCP=2*(NP+1)+3:3*(NP+1);
iTP=3*(NP+1)+2:4*(NP+1)+1;
iXP=4*(NP+1)+3:5*(NP+1)+1;

iSS=3:NS+1;
iBS=NS+3:2*(NS+1)+1;
iCS=2*(NS+1)+3:3*(NS+1);
iTS=3*(NS+1)+2:4*(NS+1)+1;
iXS=4*(NS+1)+3:5*(NS+1)+1;
iPS=5*(NS+1)+3:6*(NS+1)+1;

jSL = 1:NL+1;
jBL = NL+2:2*(NL+1);
jCL = 2*(NL+1)+1:3*(NL+1);
jTL = 3*(NL+1)+1:4*(NL+1);
```

```
jSP = 2:NP+2;

jBP=NP+3:2*(NP+1)+1;

jCP=2*(NP+1)+2:3*(NP+1)+1;

jTP=3*(NP+1)+2:4*(NP+1)+1;

jXP=4*(NP+1)+2:5*(NP+1)+1;


jSS=2:NS+2;

jBS=NS+3:2*(NS+1)+1;

jCS=2*(NS+1)+2:3*(NS+1)+1;

jTS=3*(NS+1)+2:4*(NS+1)+1;

jXS=4*(NS+1)+2:5*(NS+1)+1;

jPS=5*(NS+1)+2:6*(NS+1)+1;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX ALL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% BEGIN STREAMFUNCTION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% BC dPsi/dz = 0 at z=H

ALL(1,jSL)=DL(1,:);
```

```
% BC d^3Psi/dz^3 = 0 at z=H
ALL(2,jSL)=DL3(1,:);

% interior equation for streamfunction (psi)
ALL(iSL,jSL)=DL4(3:NL-1,:)-2*alpha^2*DL2(3:NL-1,:)+alpha^4*IL(3:NL-1,:);
%
% ... liquid buoyancy turned off!!!
%
ALL(iSL,jBL)=(-alpha*RaB)/Da*IL(3:NL-1,:);
ALL(iSL,jCL)=(-alpha*RaC)/Da*IL(3:NL-1,:);
ALL(iSL,jTL)=(-alpha*Ra)/Da*IL(3:NL-1,:);

% BC dPsi/dz|^+ - dPsi/dz|^- = 0 at z=h_p
ALL(NL,jSL)=DL(NL+1,:);

% BC psi|^+ - psi|^- = 0 at z=h_p
ALL(NL+1,NL+1)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% END STREAMFUNCTION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BEGIN B EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% BC Bhat=0 at z=H
ALL(NL+2,NL+2)=1;

% interior equation for B
ALL(iBL,jSL)=alpha*dBdzBLdiag(2:NL,:);
ALL(iBL,jBL)=DL(2:NL,:)+(1/deltaB2)*(-alpha^2*IL(2:NL,:)+DL2(2:NL,:));

% BC (h_p*dBB/dz+Bhat)^+ - h_p*dBB/dz+Bhat)^- = 0 at z=h_p
ALL(2*(NL+1),2*(NL+1))=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END B EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BEGIN C EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC Chat=0 at z=H
ALL(2*(NL+1)+1,2*(NL+1)+1)=1;

% interior equation for C
ALL(iCL,jSL)=alpha*dCdzBLdiag(2:NL,:);
ALL(iCL,jCL)=DL(2:NL,:)+(1/deltaC2)*(-alpha^2*IL(2:NL,:)+DL2(2:NL,:));

% BC (h_p*dCB/dz+Chat)^+ - h_p*dCB/dz+Chat)^- = 0 at z=h_p
```

```
ALL(3*(NL+1),3*(NL+1))=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END C EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BEGIN T EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC That = 0 at z=H
ALL(3*(NL+1)+1,3*(NL+1)+1)=1;

% ingerior equation for T
ALL(iTL,jSL)=alpha*dTdzBLdiag(2:NL,:);
ALL(iTL,jTL)=DL(2:NL,:)-alpha^2*IL(2:NL,:)+DL2(2:NL,:);

% BC That=MB*Bhat+MC*Chat at z=h_p
ALL(4*(NL+1),4*(NL+1))=1;
ALL(4*(NL+1),2*(NL+1))=-MB;
ALL(4*(NL+1),3*(NL+1))=-MC;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% END T EQUATION IN LIQUID LAYER USING LIQUID VARIABLES %%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX ALL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX ALP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BEGIN STREAMFUNCTION EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC dPsi/dz|^+ - dPsi/dz|^- = 0 at z=h_p
ALP(NL,jSP)=-DP(1,:);

% BC psi|^+ - psi|^- = 0 at z=h_p
ALP(NL+1,2)=-1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END STREAMFUNCTION EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% BEGIN B EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (h_p*dB/dz + B)^+ - (h_p*dB/dz + B)^- = 0 at z=h_p
ALP(2*(NL+1),1)=dBdzBL(NL+1)-dBdzBP(1);
ALP(2*(NL+1),NP+3)=-1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% END B EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% BEGIN C EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (h_p*dC/dz + C)^+ - (h_p*dC/dz + C)^- = 0 at z=h_p
ALP(3*(NL+1),1)=dCdzBL(NL+1)-dCdzBP(1);
ALP(3*(NL+1),2*(NP+1)+2)=-1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%% END C EQUATION IN LIQUID LAYER USING PRIMARY VARIABLES %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX ALP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX APL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% equation (64h) for interface position h_p using liquid variables
APL(1,jBL)=-MB*DL(NL+1,:);
APL(1,jCL)=-MC*DL(NL+1,:);
APL(1,jTL)=DL(NL+1,:);

% BC (alpha^2*dPsi/dz-d^3Psi/dz^3)^+=(1/Da)*(1/PI*dPsi/dz)^- at z=h_p
APL(2,jSL)=alpha^2*DL(NL+1,:)-DL3(NL+1,:);

% BC (64b) at z=h_p for B
APL(NP+3,jBL)=-V*(-1+XBP(1))*IL(NL+1,:)+1/deltaB2*DL(NL+1,:);

% BC (64c) at z=h_p for C
APL(2*(NP+1)+2,jCL)=-V*(-1+XBP(1))*IL(NL+1,:)+1/deltaC2*DL(NL+1,:);

% BC (64a) for X at z=h_p
APL(4*(NP+1)+2,jTL)=1/S*DL(NL+1,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX APL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX APP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% BEGIN EQUATION FOR h_p USING PRIMARY VARIABLES %%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% equation (64h) for interface position h_p
APP(1,1)=d2TdzBL(NL+1)-MB*d2BdzBL(NL+1)-MC*d2CdzBL(NL+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%% END EQUATION FOR h_p USING PRIMARY VARIABLES %%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BEGIN EQUATION FOR PSI IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (alpha^2*dPsi/dz-d^3Psi/dz^3)^+=(1/Da)*(1/PI*dPsi/dz)^- at z=h_p
APP(2,jSP)=-(1/Da)*(1/XBP(1)^3)*DP(1,:);

% interior streamfunction equation
```

```
APP(iSP,jSP)=-alpha^2*IP(2:NP,:)+DP2(2:NP,:)-PIstuffPtimesDP(2:NP,:);
APP(iSP,jBP)=alpha*RaB*chiCubedBPdiag(2:NP,:);
APP(iSP,jCP)=alpha*RaC*chiCubedBPdiag(2:NP,:);
APP(iSP,jTP)=alpha*Ra*chiCubedBPdiag(2:NP,:);

% BC psi|^+ - psi|^- = 0 at z=hs
APP(NP+2,NP+2)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% END EQUATION FOR PSI IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% BEGIN EQUATION FOR B IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (64b) at z=h_p for B
APP(NP+3,1)=-V*BBL(NL+1)*dXdzBP(1)-V*dBdzBL(NL+1)*(-1+XBP(1))...
    +1/deltaB2*(d2BdzBL(NL+1)-XBP(1)*d2BdzBP(1)-dXdzBP(1)*dBdzBP(1));
APP(NP+3,jBP)=-1/deltaB2*XBP(1)*DP(1,:);
APP(NP+3,4*(NP+1)+2)=-V*BBP(1)-1/deltaB2*dBdzBP(1);

% interior equation for B
APP(iBP,jSP)=alpha*dBdzBPdiag(2:NP,:);
XBPtimesDP=XBPdiag*DP;
XBPtimesDP2=XBPdiag*DP2;
```

```
dXBPtimesDP=dXdzBPdiag*DP;
APP(iBP,jBP)=dXdzBPdiag(2:NP,:)+XBPtimesDP(2:NP,:)...
    +(1/deltaB2)*(-alpha^2*XBPdiag(2:NP,:)+XBPtimesDP2(2:NP,:)...
    +dXBPtimesDP(2:NP,:));
BBPtimesDP=BBPdiag*DP;
dBBPtimesDP=dBdzBPdiag*DP;
APP(iBP,jXP)=BBPtimesDP(2:NP,:)+dBdzBPdiag(2:NP,:)...
    +(1/deltaB2)*(d2BdzBPdiag(2:NP,:)+dBBPtimesDP(2:NP,:));

% BC (h_s*dB/dz+B)^+ - (h_s*dB/dz+B)^- = 0 at z=h_s
APP(2*(NP+1)+1,2*(NP+1)+1)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% END EQUATION FOR B IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% BEGIN EQUATION FOR C IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (64c) at z=h_p for C
APP(2*(NP+1)+2,1)=-V*CBL(NL+1)*dXdzBP(1)-V*dCdzBL(NL+1)*(-1+XBP(1))...
    +1/deltaC2*(d2CdzBL(NL+1)-XBP(1)*d2CdzBP(1)-dXdzBP(1)*dCdzBP(1));
APP(2*(NP+1)+2,jCP)=-1/deltaC2*XBP(1)*DP(1,:);
APP(2*(NP+1)+2,4*(NP+1)+2)=-V*CBP(1)-1/deltaC2*dCdzBP(1);
```

```
% interior equation for C
APP(iCP,jSP)=alpha*dCdzBPdiag(2:NP,:);
XBPtimesDP=XBPdiag*DP;
XBPtimesDP2=XBPdiag*DP2;
dXBPtimesDP=dXdzBPdiag*DP;
APP(iCP,jCP)=dXdzBPdiag(2:NP,:)+XBPtimesDP(2:NP,:)...
    +(1/deltaC2)*(-alpha^2*XBPdiag(2:NP,:)+XBPtimesDP2(2:NP,:)...
    +dXBPtimesDP(2:NP,:));
CBPtimesDP=CBPdiag*DP;
dCBPtimesDP=dCdzBPdiag*DP;
d2CdzBPdiag;
APP(iCP,jXP)=CBPtimesDP(2:NP,:)+dCdzBPdiag(2:NP,:)...
    +(1/deltaC2)*(d2CdzBPdiag(2:NP,:)+dCBPtimesDP(2:NP,:));

% BC (h_s*dC/dz+C)^+ - (h_s*dC/dz+C)^- = 0 at z=h_s
APP(3*(NP+1)+1,3*(NP+1)+1)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% END EQUATION FOR C IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% BEGIN EQUATION FOR T IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% interior equation for T
```

```
APP(iTP,jBP)=-MB*IP;

APP(iTP,jCP)=-MC*IP;

APP(iTP,jTP)=IP;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% END EQUATION FOR T IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BEGIN EQUATION FOR CHI IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% BC (64a) for X at z=h_p
APP(4*(NP+1)+2,1)=-V*dXdzBP(1)+1/S*(d2TdzBL(NL+1)-d2TdzBP(1));
APP(4*(NP+1)+2,jTP)=-1/S*DP(1,:);
APP(4*(NP+1)+2,4*(NP+1)+2)=-V;


% interior equation for chi
APP(iXP,jSP)=alpha*dTdzBPdiag(2:NP+1,:);
APP(iXP,jTP)=DP(2:NP+1,:)-alpha^2*IP(2:NP+1,:)+DP2(2:NP+1,:);
APP(iXP,jXP)=S*DP(2:NP+1,:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% END EQUATION FOR CHI IN THE PRIMARY LAYER USING PRIMARY VARIABLES %%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX APP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX APS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC psi|^+ - psi|^- = 0 at z=h_s
APS(NP+2,2)=-1;

% BC (h_s*dB/dz+B)^+ - (h_s*dB/dz+B)^- = 0 at z=h_s
APS(2*(NP+1)+1,1)=dBdzBP(NP+1)-dBdzBS(1);
APS(2*(NP+1)+1,NS+3)=-1;

% BC (h_s*dC/dz+C)^+ - (h_s*dC/dz+C)^- = 0 at z=h_s
APS(3*(NP+1)+1,1)=dCdzBP(NP+1)-dCdzBS(1);
APS(3*(NP+1)+1,2*(NS+1)+2)=-1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX APS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX ASP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% condition for h_s using primary variables (66i)
ASP(1,jBP)=MBC*DP(NP+1,:);
ASP(1,jCP)=-MCC*DP(NP+1,:);

% BC dPsi/dz|^+ - dPsi/dz|^- = 0 at z=h_s
ASP(2,jSP)=DP(NP+1,:);

% BC (66c) for C at z=h_s
ASP(2*(NS+1)+2,jCP)=-V*(-XBP(NP+1)+XBS(1))*IP(NP+1,:)...
    +1/deltaC2*XBP(NP+1)*DP(NP+1,:);
ASP(2*(NS+1)+2,5*(NP+1)+1)=V*CBP(NP+1)+1/deltaC2*dCdzBP(NP+1);

% BC (66a) for X at z=h_s
ASP(4*(NS+1)+2,5*(NP+1)+1)=V;
ASP(4*(NS+1)+2,jTP)=DP(NP+1,:)/S;

% BC (66b) for phi_b at z=h_s
% ASP(5*(NS+1)+2,jBP)=-V*(-XBP(NP+1)+XBS(1))*IP(NP+1,:)...
%     +1/deltaB2*XBP(NP+1)*DP(NP+1,:);
% ASP(5*(NS+1)+2,5*(NP+1)+1)=V*BBP(NP+1)+1/deltaB2*dBdzBP(NP+1);
ASP(5*(NS+1)+2,jBP)=-V*(-XBP(NP+1)+XBS(1)-PBS(1))*IP(NP+1,:)...
```

```
    +1/deltaB2*XBP(NP+1)*DP(NP+1,:);
ASP(5*(NS+1)+2,5*(NP+1)+1)=V*BBP(NP+1)+1/deltaB2*dBdzBP(NP+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX ASP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX ASS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% BEGIN EQUATION FOR INTERFACE POSITION h_s USING SECONDARY VARIABLES %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% condition for h_s (66i)
ASS(1,1)=MBC*d2BdzBP(NP+1)-MCC*d2CdzBP(NP+1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%% END EQUATION FOR h_s USING SECONDARY VARIABLES %%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEGIN EQUATION FOR PSI IN THE SECONDARY LAYER USING SECONDARY VARIABLES %
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC dPsi/dz|^+ - dPsi/dz|^- = 0 at z=h_s
ASS(2,jSS)=-DS(1,:);

% interior streamfunction equation
ASS(iSS,jSS)=-alpha^2*IS(2:NS,:)+DS2(2:NS,:)-PIstuffStimesDS(2:NS,:);
ASS(iSS,jBS)=alpha*RaB*chiCubedBSdiag(2:NS,:);
ASS(iSS,jCS)=alpha*RaC*chiCubedBSdiag(2:NS,:);
ASS(iSS,jTS)=alpha*Ra*chiCubedBSdiag(2:NS,:);

% BC Psi=0 at z=0
ASS(NS+2,NS+2)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% END EQUATION FOR PSI IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BEGIN EQUATION FOR B IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% interior equation for B
ASS(iBS,jBS)=IS;
ASS(iBS,jTS)=1/MBC*IS;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% END EQUATION FOR B IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BEGIN EQUATION FOR C IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (66c) for C at z=h_s
ASS(2*(NS+1)+2,1)=-V*CBP(NP+1)*(-dXdzBP(NP+1)+dXdzBS(1))...
    -V*dCdzBP(NP+1)*(-XBP(NP+1)+XBS(1))...
    +1/deltaC2*(XBP(NP+1)*d2CdzBP(NP+1)-XBS(1)*d2CdzBS(1)...
    +dXdzBP(NP+1)*dCdzBP(NP+1)-dXdzBS(1)*dCdzBS(1));
ASS(2*(NS+1)+2,jCS)=-1/deltaC2*XBS(1)*DS(1,:);
ASS(2*(NS+1)+2,4*(NS+1)+2)=-V*CBP(NP+1)-1/deltaC2*dCdzBS(1);

% interior equation for C
ASS(iCS,jSS)=alpha*dCdzBSdiag(2:NS,:);
XBStimesDS=XBSdiag*DS;
XBStimesDS2=XBSdiag*DS2;
dXBStimesDS=dXdzBSdiag*DS;
ASS(iCS,jCS)=dXdzBSdiag(2:NS,:)+XBStimesDS(2:NS,:)...
    +(1/deltaC2)*(-alpha^2*XBSdiag(2:NS,:)+XBStimesDS2(2:NS,:)...
    +dXBStimesDS(2:NS,:));
CBStimesDS=CBSdiag*DS;
dCBStimesDS=dCdzBSdiag*DS;
```

```
ASS(iCS,jXS)=CBStimesDS(2:NS,:)+dCdzBSdiag(2:NS,:)...
    +(1/deltaC2)*(d2CdzBSdiag(2:NS,:)+dCBStimesDS(2:NS,:));

% BC Chat=0 at z=0
ASS(3*(NS+1)+1,3*(NS+1)+1)=1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END EQUATION FOR C IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% BEGIN EQUATION FOR T IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% interior equation for T
ASS(iTS,jCS)=IS;
ASS(iTS,jTS)=1/MCC*IS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%% END EQUATION FOR T IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BEGIN EQUATION FOR CHI IN THE SECONDARY LAYER USING SECONDARY VARIABLES %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% BC (66a) for X at z=h_s
ASS(4*(NS+1)+2,1)=-V*(-dXdzBP(NP+1)+dXdzBS(1))...
    +1/S*(d2TdzBP(NP+1)-d2TdzBS(1));
ASS(4*(NS+1)+2,4*(NS+1)+2)=-V;
ASS(4*(NS+1)+2,jTS)=-DS(1,:)/S;

% interior equation for chi
ASS(iXS,jSS)=alpha*dTdzBSdiag(2:NS+1,:);
ASS(iXS,jTS)=DS(2:NS+1,:)-alpha^2*IS(2:NS+1,:)+DS2(2:NS+1,:);
ASS(iXS,jXS)=S*DS(2:NS+1,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% END EQUATION FOR CHI IN THE SECONDARY LAYER USING SECONDARY VARIABLES %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%BEGIN EQUATION FOR PHI_B IN THE SECONDARY LAYER USING SECONDARY VARIABLES%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC (66b) for phi_B at z=h_s
% ASS(5*(NS+1)+2,1)=-V*dBdzBP(NP+1)*(-XBP(NP+1)+XBS(1))...
%     -V*BBP(NP+1)*(-dXdzBP(NP+1)+dXdzBS(1))...
%     +1/deltaB2*(XBP(NP+1)*d2BdzBP(NP+1)-XBS(1)*d2BdzBS(1)...
%     +dXdzBP(NP+1)*dBdzBP(NP+1)-dXdzBS(1)*dBdzBS(1));
% ASS(5*(NS+1)+2,4*(NS+1)+2)=-V*BBP(NP+1)-1/deltaB2*dBdzBS(1);
% ASS(5*(NS+1)+2,jBS)=-1/deltaB2*XBS(1)*DS(1,:);
```

```
ASS(5*(NS+1)+2,1)=-V*dBdzBP(NP+1)*(-XBP(NP+1)+XBS(1))...
    -V*BBP(NP+1)*(-dXdzBP(NP+1)+dXdzBS(1)+dPdzBS(1))...
    -V*(BBP(NP+1)-1)*(-dPdzBS(1))-V*dBdzBP(NP+1)*(-PBS(1))...
    +1/deltaB2*(XBP(NP+1)*d2BdzBP(NP+1)-XBS(1)*d2BdzBS(1)...
    +dXdzBP(NP+1)*dBdzBP(NP+1)-dXdzBS(1)*dBdzBS(1));
ASS(5*(NS+1)+2,4*(NS+1)+2)=-V*BBP(NP+1)-1/deltaB2*dBdzBS(1);
ASS(5*(NS+1)+2,jBS)=-1/deltaB2*XBS(1)*DS(1,:);
ASS(5*(NS+1)+2,5*(NS+1)+2)=-V;

% interior equation for phi_B
ASS(iPS,jSS)=alpha*dBdzBSdiag(2:NS+1,:);
XBStimesDS=XBSdiag*DS;
dXBStimesDS=dXdzBSdiag*DS;
XBStimesDS2=XBSdiag*DS2;
ASS(iPS,jBS)=dXdzBSdiag(2:NS+1,:)+XBStimesDS(2:NS+1,:)...
    +1/deltaB2*(-alpha^2*XBSdiag(2:NS+1,:)+dXBStimesDS(2:NS+1,:)...
    +XBStimesDS2(2:NS+1,:));
BBStimesDS=BBSdiag*DS;
dBBStimesDS=dBdzBSdiag*DS;
ASS(iPS,jXS)=BBStimesDS(2:NS+1,:)+dBdzBSdiag(2:NS+1,:)...
    +1/deltaB2*(dBBStimesDS(2:NS+1,:)+d2BdzBSdiag(2:NS+1,:));
ASS(iPS,jPS)=DS(2:NS+1,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% END EQUATION FOR PHI_B IN THE SECONDARY LAYER USING SECONDARY VARIABLES %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX ASS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% combine all nine submatrices to form the A matrix to be returned
A = [ALL ALP ALS;
    APL APP APS;
    ASL ASP ASS];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX BLL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% interior equation for B in the liquid layer using liquid variables
BLL(iBL,jBL)=IL(2:NL,:);

% interior equation for C in the liquid layer using liquid variables
BLL(iCL,jCL)=IL(2:NL,:);

% interior equation for T in the liquid layer using liquid variables
BLL(iTL,jTL)=IL(2:NL,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX BLL %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX BPP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC for B at z=h_p
BPP(NP+3,1)=BBL(NL+1)*(-1+XBP(1));

% interior B equation in primary layer using primary variables
BPP(iBP,jBP)=XBPdiag(2:NP,:);
BPP(iBP,jXP)=BBPdiag(2:NP,:);

% BC for C at z=h_p
BPP(2*(NP+1)+2,1)=CBL(NL+1)*(-1+XBP(1));

% interior C equation in primary layer using primary variables
BPP(iCP,jCP)=XBPdiag(2:NP,:);
BPP(iCP,jXP)=CBPdiag(2:NP,:);

% BC for X and z=h_p
BPP(4*(NP+1)+2,1)=-1+XBP(1);
```

```
% interior X equation in primary layer using primary variables
BPP(iXP,jTP)=IP(2:NP+1,:);
BPP(iXP,jXP)=S*IP(2:NP+1,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX BPP %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% BEGIN MATRIX BSS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% BC for C at z=h_s
BSS(2*(NS+1)+2,1)=CBP(NP+1)*(-XBP(NP+1)+XBS(1));

% interior equation for C in the secondary layer using secondary variables
BSS(iCS,jCS)=XBSdiag(2:NS,:);
BSS(iCS,jXS)=CBSdiag(2:NS,:);

% BC for X at x=h_s
BSS(4*(NS+1)+2,1)=-XBP(NP+1)+XBS(1);

% interior equation for X in the secondary layer using secondary variables
BSS(iXS,jTS)=IS(2:NS+1,:);
```

```
BSS(iXS,jXS)=S*IS(2:NS+1,:);


% BC for phi_B at z=h_s
%BSS(5*(NS+1)+2,1)=BBP(NP+1)*(-XBP(NP+1)+XBS(1));
BSS(5*(NS+1)+2,1)=BBP(NP+1)*(-XBP(NP+1)+XBS(1)+PBS(1))...
     +(BBP(NP+1)-1)*(-PBS(1));


% interior equation for phi_B in the secondary layer using secondary
% variables
BSS(iPS,jBS)=XBSdiag(2:NS+1,:);
BSS(iPS,jXS)=BBSdiag(2:NS+1,:);
BSS(iPS,jPS)=IS(2:NS+1,:);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%% END MATRIX BSS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% combine all nine submatrices to form the B matrix to be returned
B = [BLL BLP BLS;
    BPL BPP BPS;
    BSL BSP BSS];


return;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% getBaseState.m
%
% Function to compute and return some values associated with the base state
% solution.
%
% INPUTS: guessVec...vector of three guesses for the unknowns hS,hP and BP
%
% RETURNS: zSec.............z values in secondary mush
%          Bsec.............B profile through secondary mush
%          Csec.............C profile through secondary mush
%          tempSec..........temperature profile through secondary mush
%          chiSec...........liquid fraction profile through secondary mush
%          phiBsec..........solid-fraction B profile in secondary mush
%          zPri.............vector of z values in primary mush
%          Bpri.............B profile through primary mush
%          Cpri.............C profile through primary mush
%          tempPri..........temperature profile through primary mush
%          zLiq.............z values through liquid layer
%          Bliq.............B profile through liquid layer
%          Cliq.............C profile through liquid layer
%          tempLiq..........temperature profile through liquid layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [tempLiq,Bliq,Cliq,tempPri,Bpri,Cpri,chiPri,tempSec,Bsec,Csec,...
```

```
    chiSec,phiBsec,hP,hS,zLiq,zPri,zSec] = getBaseState(guessVec,BE,TE,...
    CE,deltabsq,deltacsq,Binfty,Cinfty,Tinfty,gamma,BEAB,TEAB,TM)

% function [zSec,Bsec,Csec,tempSec,chiSec,phiBsec,zPri,Bpri,Cpri, ...
%     tempPri,zLiq,Bliq,Cliq,tempLiq] = getBaseState(guessVec,BE,TE,CE,...
%     deltabsq,deltacsq,Binfty,Cinfty,Tinfty,gamma,BEAB,TEAB,TM)

% model parameters to be used throughout program
% BE = 1/3;       % liquid composition of B at eutectic interface
% TE = -19;       % temperature at the eutectic interface
% CE = 1/3;       % liquid composition of C at the eutectic interface
% deltabsq = 100; % kappa / DB (ratio of thermal to solutal diffusivity)
% deltacsq = 100; % kappa / DC (ratio of thermal to solutal diffusivity)
% Binfty = 0.2;   % far-field liquid composition of B
% Tinfty = 5;     % far-field temperature
% Cinfty = 0.1;   % far-field liquid composition of C
% gamma = -1;     % Lv / cbar*(mB+mC)
% BEAB = 0.5;     % eutectic concentration of B on the AB binary diagram
% TEAB = -5;      % eutectic temperature on the AB binary diagram
% TM = 0;         % temperature above which water is liquid and below which
                % it is solid

% get the slopes
M = getMvec(BE, TE, CE, TEAB, BEAB, TM);

% liquidus and cotectic line slopes
```

```
mB = M(1);

mC = M(2);

mBC = M(3);

mCC = M(4);


mBbar = mB /(mB + mC);

mCbar = mC /(mB + mC);


gammaB = ((mB + mC) / mBC) * gamma;


% set up a vector of (constant) phase diagram parameters to be passed to

% other functions needing these values

pdVec(1) = BE;

pdVec(2) = TE;

pdVec(3) = CE;

pdVec(4) = deltabsq;

pdVec(5) = deltacsq;

pdVec(6) = Binfty;

pdVec(7) = Tinfty;

pdVec(8) = Cinfty;

pdVec(9) = gamma;

pdVec(10) = gammaB;

pdVec(11) = BEAB;

pdVec(12) = TEAB;

pdVec(13) = TM;

pdVec(14) = mB;
```

```
pdVec(15) = mC;

pdVec(16) = mBC;

pdVec(17) = mCC;

pdVec(18) = mBbar;

pdVec(19) = mCbar;


% compute some basic values within the mushy layers

[zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);


% read off interface positions and liquid compositions at those interfaces

hS = zSec(length(zSec));

hP = zPri(length(zPri));

BP = Bpri(length(Bpri));

CP = Cpri(length(Cpri));


% create a vector of z values at which to compute B, C and T in the

% liquid layer

zLiq = [hP:0.01:1.0]';


% compute values for quantities through the different layers

Csec = Csecondary(Bsec, pdVec);

tempSec = tempSecondary(Bsec,pdVec);

tempPri = tempPrimary(Bpri,Cpri,pdVec);

TP = tempPri(length(tempPri));

chiSec = chiSecondary(Bsec, pdVec);

phiBsec = phiBsecondary(Bsec,chiSec,pdVec);
```

```
phiAsec = 1 - chiSec - phiBsec;

chiPri = chiPrimary(Bpri,BP,Cpri,CP,pdVec);

phiApri = 1 - chiPri;

Bliq = Bliquid(BP,hP,zLiq,pdVec);

tempLiq = Tliquid(TP,hP,zLiq,pdVec);

Cliq = Cliquid(CP,hP,zLiq,pdVec);


% make plot of the base state

plotBaseState(zLiq,Bliq,Cliq,tempLiq,zPri,Bpri,Cpri,tempPri,phiApri,...

    chiPri,zSec,Bsec,Csec,tempSec,phiBsec,phiAsec,chiSec);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% getMvec.m
%
% Function to return the four liquidus and cotectic line slopes.
%
% INPUTS: BE.....concentration of B at the eutectic point on the ternary
%                phase diagram
%         TE.....temperature at the ternary eutectic point on the ternary
%                phase diagram
%         CE.....concentration of C at the eutectic point on the ternary
%                phase diagram
%         TEAB...temperature at the eutectic point on the AB phase diagram
%         BEAB...concentration of B at the eutectic point on the AB binary
%                phase diagram
%         TM.....temperature below which water is solid and above which it
%                is liquid
%
% OUTPUTS: Mvec...vector of four slopes [mB, mC, mBC, mCC]
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Mvec = getMvec(BE, TE, CE, TEAB, BEAB, TM)


% create the vector of liquidus and cotectic line slopes
Mvec(1) = -(TM - TEAB) / BEAB;                      % mB
Mvec(3) = (TEAB - TE) / (BE - BEAB);                % mBC
```

```
Mvec(4) = (TEAB - TE) / CE;                          % mCC

Mvec(2) = -Mvec(4) - (Mvec(1) * Mvec(4)) / Mvec(3); % mC
```

```
% makes plot resembling figure 2 from DMA's JFM, 2003 paper


function makeFig2()


guessVec = [0.09,0.4,0.2];

Barr = 0.1:0.01:0.43;

Carr = 0.01:0.01:0.15;


% model parameters to be used throughout program

BE = 1/3;       % liquid composition of B at eutectic point

TE = -19;       % temperature at the eutectic point

CE = 1/3;       % liquid composition C at the eutectic point

deltabsq = 100; % kappa / DB

deltacsq = 100; % kappa / DC

Binfty = 0.2;   % far-field liquid compostion of B

Tinfty = 5;     % far-field temperature

Cinfty = 0.1;   % far-field liquid composition of C

gamma = -1;     % Lv / cbar*(mB+mC)

BEAB = 0.5;     % eutectic B on the AB binary diagram

TEAB = -5;      % eutectic temperature on the AB binary diagram

TM = 0;         % temperature above which water is liquid and below which

                % it is solid


% get the slopes

M = getMvec(BE, TE, CE, TEAB, BEAB, TM);
```

```
% liquidus and cotectic line slopes
mB = M(1);
mC = M(2);
mBC = M(3);
mCC = M(4);

mBbar = mB /(mB + mC);
mCbar = mC /(mB + mC);

gammaB = ((mB + mC) / mBC) * gamma;

% set up a vector of (constant) phase diagram parameters to be passed to
% other functions needing these values
pdVec(1) = BE;
pdVec(2) = TE;
pdVec(3) = CE;
pdVec(4) = deltabsq;
pdVec(5) = deltacsq;
pdVec(6) = Binfty;
pdVec(7) = Tinfty;
pdVec(8) = Cinfty;
pdVec(9) = gamma;
pdVec(10) = gammaB;
pdVec(11) = BEAB;
pdVec(12) = TEAB;
pdVec(13) = TM;
```

```
pdVec(14) = mB;

pdVec(15) = mC;

pdVec(16) = mBC;

pdVec(17) = mCC;

pdVec(18) = mBbar;

pdVec(19) = mCbar;



for i=1:length(Barr)

    pdVec(6)=Barr(i);

    [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);

    hSarr1(i) = zSec(length(zSec));

    hParr1(i) = zPri(length(zPri));

end


subplot (3,1,1);

plot(Barr,hParr1,'r--',Barr,hSarr1,'b--');

xlabel('$B_{\infty}$','Interpreter','latex');

ylabel('$\bar{z}$','Interpreter','latex');

set(get(gca,'YLabel'),'Rotation',0.0);

text(0.25,0.35,'h^P');

text(0.25,0.15,'h^S');

axis([0.1 0.4 0 0.5]);

set(gca,'XTick',0.1:0.1:0.45);

set(gca,'YTick',0:0.1:0.5);
```

```
pdVec(6) = 0.2;


for i=1:length(Carr)
    pdVec(8)=Carr(i);
    [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);
    hSarr2(i) = zSec(length(zSec));
    hParr2(i) = zPri(length(zPri));
end


subplot(3,1,2);
plot(Carr,hParr2,'r--',Carr,hSarr2,'b--');
xlabel('$C_{\infty}$','Interpreter','latex');
ylabel('$\bar{z}$','Interpreter','latex');
set(get(gca,'YLabel'),'Rotation',0.0);
text(0.08,0.3,'h^P');
text(0.08,0.05,'h^S');
axis([0 0.2 0 0.5]);
set(gca,'XTick',0:0.05:0.2);
set(gca,'YTick',0:0.1:0.5);


pdVec(8) = 0.1;


TL = TM + mB * Binfty + mC * Cinfty;


Tarr = 0.01:1:20;
```

```
for i=1:length(Tarr)
    pdVec(7)=Tarr(i)+TL;
    [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);
    hSarr3(i) = zSec(length(zSec));
    hParr3(i) = zPri(length(zPri));
end


subplot(3,1,3);
plot(Tarr,hParr3,'r--',Tarr,hSarr3,'b--');
axis([0 20 0 0.8]);
set(gca,'XTick',0:5:20);
set(gca,'YTick',0:0.2:0.8);
xlabel('$T_{\infty}-T^{\mathcal{L}}(B_{\infty},C_{\infty})$', ...
    'Interpreter','latex');
ylabel('$\bar{z}$','Interpreter','latex');
set(get(gca,'YLabel'),'Rotation',0.0);
text(10,0.5,'h^P');
text(10,0.2,'h^S');
```

```
function makeFig7()


Barr = 0.1:0.01:0.43;

Carr = 0.01:0.01:0.15;


% model parameters to be used throughout program

BE = 1/3;       % liquid composition of B at eutectic point

TE = -19;       % temperature at the eutectic point

CE = 1/3;       % liquid composition of C at the eutectic point

deltabsq = 100; % kappa / DB

deltacsq = 100; % kappa / DC

Binfty = 0.2;   % far-field liquid compostion of B

Tinfty = 5;     % far-field temperature

Cinfty = 0.1;   % far-field liquid composition of C

gamma = -1;     % Lv / cbar*(mB+mC)

BEAB = 0.5;     % eutectic B on the AB binary diagram

TEAB = -5;      % eutectic temperature on the AB binary diagram

TM = 0;         % temperature above which water is liquid and below which

                % it is solid


guessVec = [0.09,0.4,0.2];


% get the slopes

M = getMvec(BE, TE, CE, TEAB, BEAB, TM);


% liquidus and cotectic line slopes
```

```
mB = M(1);

mC = M(2);

mBC = M(3);

mCC = M(4);


mBbar = mB /(mB + mC);

mCbar = mC /(mB + mC);


gammaB = ((mB + mC) / mBC) * gamma;


% set up a vector of (constant) phase diagram parameters to be passed to

% other functions needing these values

pdVec(1) = BE;

pdVec(2) = TE;

pdVec(3) = CE;

pdVec(4) = deltabsq;

pdVec(5) = deltacsq;

pdVec(6) = Binfty;

pdVec(7) = Tinfty;

pdVec(8) = Cinfty;

pdVec(9) = gamma;

pdVec(10) = gammaB;

pdVec(11) = BEAB;

pdVec(12) = TEAB;

pdVec(13) = TM;

pdVec(14) = mB;
```

```
pdVec(15) = mC;

pdVec(16) = mBC;

pdVec(17) = mCC;

pdVec(18) = mBbar;

pdVec(19) = mCbar;



for i=1:length(Barr)

    pdVec(6)=Barr(i);

    % compute some basic values within the mushy layers

    [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);

    hSarr1(i) = zSec(length(zSec));

    hParr1(i) = zPri(length(zPri));

    BP = Bpri(length(Bpri));

    CP = Cpri(length(Cpri));

end


subplot (2,1,1);

plot(Barr,hParr1,'r--',Barr,hSarr1,'b--');

xlabel('$B_{\infty}$','Interpreter','latex');

ylabel('$\bar{z}$','Interpreter','latex');

set(get(gca,'YLabel'),'Rotation',0.0);

text(0.25,0.35,'h^P');

text(0.25,0.15,'h^S');

hold on;
```

```
pdVec(6) = 0.2;


for i=1:length(Carr)
    pdVec(8)=Carr(i);
    % compute some basic values within the mushy layers
    [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, pdVec);
    hSarr2(i) = zSec(length(zSec));
    hParr2(i) = zPri(length(zPri));
    BP = Bpri(length(Bpri));
    CP = Cpri(length(Cpri));
end


subplot(2,1,2);
plot(Carr,hParr2,'r--',Carr,hSarr2,'b--');
xlabel('$C_{\infty}$','Interpreter','latex');
ylabel('$\bar{z}$','Interpreter','latex');
set(get(gca,'YLabel'),'Rotation',0.0);
text(0.08,0.3,'h^P');
text(0.08,0.05,'h^S');
hold off;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% phiBsecondary.m
%
% Computes solid-fraction of B in the secondary layer.
%
% INPUTS: B.........liquid composition of B in the secondary mushy layer
%
%         chi........liquid-fraction in the secondary mushy layer
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: phiB...profile of solid fraction B through the secondary mush
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function phiB = phiBsecondary(B, chi, paramVec)


% separate out phase diagram constants
deltabsq = paramVec(4);
deltacsq = paramVec(5);
Binfty = paramVec(6);
Cinfty = paramVec(8);
BEAB = paramVec(11);
mBC = paramVec(16);
mCC = paramVec(17);
```

```
% solid fraction of B in the secondary mushy layer
phiB = (deltacsq./deltabsq) .* ((B - BEAB) .* chi - (mCC/mBC) .* Cinfty) ...
    - chi .* B + Binfty;
```

```
function plotBaseState(zLiq,Bliq,Cliq,tempLiq,zPri,Bpri,Cpri,tempPri,...
    phiApri,chiPri,zSec,Bsec,Csec,tempSec,phiBsec,phiAsec,chiSec)

% combine some of the input vectors...
z = [zSec; zPri; zLiq];
zMush = [zSec; zPri];
B = [Bsec; Bpri; Bliq];
C = [Csec; Cpri; Cliq];
temp = [tempSec; tempPri; tempLiq];

% obtain the interface positions and values of B and C at the mush-liquid
% interface
hS = zSec(numel(zSec));
hP = zPri(numel(zPri));
BP = Bpri(numel(Bpri));
CP = Cpri(numel(Cpri));

figure;
% plot temperature profile
subplot(2, 2, 1);
plot(temp, z, [-30 0], [hP hP], 'k--', ...
    [-30 0], [hS hS], 'k--');
xlabel('Temperature');
ylabel('$\bar{z}$','Interpreter','latex');
axis([-30 0 0 1]);
set(gca,'XTick',-30:10:0);
```

```
set(gca,'YTick',0:0.2:1.0);
set(get(gca,'YLabel'),'Rotation',0.0);

% plot liquid composition profiles
subplot(2,2,2);
plot(B,z,C,z,[0 1],[hP hP],'k--', ...
    [0 1], [hS hS], 'k--');
xlabel('Liquid compositions');
ylabel('$\bar{z}$','Interpreter','latex');
axis([0 1 0 1]);
set(gca,'XTick',0:0.2:1.0);
set(gca,'YTick',0:0.2:1.0);
set(get(gca,'YLabel'),'Rotation',0.0);
text(0.25,0.8,'B','FontSize',8);
text(0.125,0.8,'C','FontSize',8);

phiA = [phiAsec; phiApri];
phib = zeros(length(zPri), 1);
phib = [phiBsec; phib];
phibPlusPhiA = phib + phiA;

% plot solid-fractions
subplot(2,2,3);
plot(phibPlusPhiA,zMush,[0 1],[hS hS],'k--',[0 1], [hP hP], 'k--', ...
    phiA,zMush);
axis([0 1 0 1]);
```

```
xlabel('Solid fractions');

ylabel('$\bar{z}$','Interpreter','latex');

text(0.1,0.2,'A','FontSize',8);

text(0.3,0.04,'A','FontSize',8);

text(0.58, 0.04,'B','FontSize',8);

set(gca,'XTick',0:0.2:1.0);

set(gca,'YTick',0:0.2:1.0);

set(get(gca,'YLabel'),'Rotation',0.0);


% plot solidification path for symmetric case

subplot(2,2,4);

plot(C,B, [0 1/3], [0.5 1/3], 'k--', [1/3 0.5], [1/3 0], 'k--');

axis([0 0.5 0 0.5]);

xlabel('C');

ylabel('B');

set(gca,'XTick',0:0.1:0.5);

set(gca,'YTick',0:0.1:0.5);

set(get(gca,'YLabel'),'Rotation',0.0);
```

```
%  file 'pm_three_layer.m'
%  Ternary mushy layer linear stability code with solute diffusion
%  and three layers for eutectic phase diagram.

clear all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%% INPUT PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%
% far-field ...
%
H=2.0;
Binfty = 0.2;   % far-field liquid composition of B
Cinfty = 0.1;   % far-field liquid composition of C
Tinfty = 5;     % far-field temperature
% Tinf=0.4;
% Binf=0.55;
% Cinf=0.35;
%
% phase diagram:
%
BE = 1/3;       % liquid composition of B at eutectic interface
TE = -19;       % temperature at the eutectic interface
CE = 1/3;       % liquid composition of C at the eutectic interface
BEAB = 0.5;     % eutectic concentration of B on the AB binary diagram
```

```
TEAB = -5;      % eutectic temperature on the AB binary diagram
%
% C1E=1/3;
% C2E=1/3;
% C1AB=1/2;
% C2AB=1/2;
%
% for symmetric case (with TEAB=TEAC, not just symmetric compositions)
% need a special TEAB
%
%TEAB=-1+(1/6)/(C1inf-1/3);
%
% M1C=(TEAB+1)/(C1AB-C1E);
% M2C=(TEAB+1)/(C2AB-C2E);
% M1=(1-M2C*(C2inf-C2E))/(C1inf-C1E-(M2C/M1C)*(C2inf-C2E));
% M2=M2C*(1-M1/M1C);
%
% other related quantities
%
deltabsq = 100; % kappa / DB (ratio of thermal to solutal diffusivity)
deltacsq = 100; % kappa / DC (ratio of thermal to solutal diffusivity)
gamma = -1;     % Lv / cbar*(mB+mC)
TM = 0;         % temperature above which water is liquid and below which
                % it is solid
guessVec=[0.1,0.2,0.3]; % vector of guesses for hS, hP and BP
```

```
M = getMvec(BE, TE, CE, TEAB, BEAB, TM);

% liquidus and cotectic line slopes

mB = M(1);

mC = M(2);

mBC = M(3);

mCC = M(4);


Da=0.05;

Vel=1;


% TS1star=-1-M1C*C1E;

% TS2star=-1+M2C*(1-C2E);

% TPstar=-1+M1*(1-C1E)-M2*C2E;


% !!!!! etc ...


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%% SETUP THREE LAYER BASE STATE %%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%

[TBLx,BBLx,CBLx,...

TBPx,BBPx,CBPx,XBPx,...

TBSx,BBSx,CBSx,XBSx,PHIBBSx,...

hpB,hsB,zlB,zpB,zsB] = getBaseState(guessVec,BE,TE,CE,...

deltabsq,deltacsq,Binfty,Cinfty,Tinfty,gamma,BEAB,TEAB,TM);
```

```
% [TBLx,BBLx,CBLx,...
%  TBPx,BBPx,CBPx,XBPx,...
%  TBSx,BBSx,CBSx,XBSx,PHIBBSx,...
%  hpB,hsB,zlB,zpB,zsB]=get_base_state(...parameters to pass...);
%
% plot_base_state;
%
% xbs_save=[zlB' TBLx BBLx CBLx ...
%           zpB' TBPx BBPx CBPx XBPx ...
%           zsB' TBSx BBSx CBSx XBSx PHIBBSx];
% save 'basestate_dat.m' xbs_save -ascii -double
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%% Set up Chebyshev Points, Differentiation Matrices %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
NL=16;
NP=16;
NS=16;
[barzl,zl,DL,DL2,DL3,DL4,IL]=setup_cheb(NL,H,hpB);
% barzl are Chebyshev points cos(j*pi/NL) j=0:NL
% zl are points rescaled on [hpB,H]
% DL,DL2 are wrt [hpB,H]
[barzp,zp,DP,DP2,DP3,DP4,IP]=setup_cheb(NP,hpB,hsB);
% barzp are Chebyshev points cos(j*pi/NP) j=0:NP
% zp are points rescaled on [hsB,hpB]
```

```
% DP,DP2 are wrt [hsB,hpB]
[barzs,zs,DS,DS2,DS3,DS4,IS]=setup_cheb(NS,hsB,0);
% barzs are Chebyshev points cos(j*pi/NS) j=0:NS
% zs are points rescaled on [0,hsB]
% DS,DS2 are wrt [0,hsB]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% CONVERT BASE STATE EXPRESSIONS TO CHEBYSHEV POINTS %%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
TBL=spline(zlB,TBLx,zl);  % provides base state liquid temperature at points zl
BBL=spline(zlB,BBLx,zl);
CBL=spline(zlB,CBLx,zl);
TBP=spline(zpB,TBPx,zp);
BBP=spline(zpB,BBPx,zp);
CBP=spline(zpB,CBPx,zp);
XBP=spline(zpB,XBPx,zp);
TBS=spline(zsB,TBSx,zs);
BBS=spline(zsB,BBSx,zs);
CBS=spline(zsB,CBSx,zs);
XBS=spline(zsB,XBSx,zs);
PHIBBS=spline(zsB,PHIBBSx,zs);
% !!!!! etc.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%% DEFINE BASE STATE EXPRESSIONS USED TO DEFINE MATRICES %%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
```

```
dBdzBL=DL*BBL';

dCdzBL=DL*CBL';

dBdzBP=DP*BBP';

dCdzBP=DP*CBP';

dTdzBPdiag=diag(DP*TBP');

dBdzBLdiag=diag(DL*BBL');

dCdzBLdiag=diag(DL*CBL');

dTdzBLdiag=diag(DL*TBL');

dBdzBPdiag=diag(DP*BBP');

dXdzBPdiag=diag(DP*XBP');

XBPdiag=diag(XBP);

BBPdiag=diag(BBP);

d2BdzBPdiag=diag(DP2*BBP');

d2TdzBL=DL2*TBL';

d2TdzBP=DP2*TBP';

d2BdzBL=DL2*BBL';

d2CdzBL=DL2*CBL';

chiCubedBPdiag=diag(XBP).^3;

dXdzBP=DP*XBP';

d2BdzBP=DP2*BBP';

d2CdzBP=DP2*CBP';

dCdzBPdiag=diag(DP*CBP');

CBPdiag=diag(CBP);

d2CdzBPdiag=diag(DP2*CBP');

dBdzBS=DS*BBS';

dCdzBS=DS*CBS';
```

```
chiCubedBSdiag=diag(XBS).^3;

BBSdiag=diag(BBS);

dXdzBS=DS*XBS';

dPdzBS=DS*PHIBBS';

d2CdzBS=DS2*CBS';

dCdzBSdiag=diag(DS*CBS');

XBSdiag=diag(XBS);

CBSdiag=diag(CBS);

d2CdzBSdiag=diag(DS2*CBS');

d2TdzBS=DS2*TBS';

dTdzBSdiag=diag(DS*TBS');

d2BdzBS=DS2*BBS';

dBdzBSdiag=diag(DS*BBS');

dXdzBSdiag=diag(DS*XBS');

d2BdzBSdiag=diag(DS2*BBS');


deltaT=TM-TE;

MB=mB/deltaT;

MC=mC/deltaT;

MBC=mBC/deltaT;

MCC=mCC/deltaT;

%S=167*(1/deltaT);

S=(gamma*(mB+mC))/(TM-TE);


for i=1:NP+1

    PIstuffPtimesDP(i,:)=(3/XBP(i)*dXdzBP(i))*DP(i,:);
```

```
end


for i=1:NS+1
    PIstuffStimesDS(i,:)=(3/XBS(i)*dXdzBS(i))*DS(i,:);
end


% TBLMAT=diag(TBL);      % base state temperature
% D_TBL=diag(DL*TBL);    % first derivative of base state temperature
% D2_TBL=diag(DL2*TBL);  % second derivative of base state temperature


% !!!!! etc....
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%% GET EIGENVALUES AND EIGENFUNCTIONS %%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%
%%%
%%%%
%%%%%
%%%%%%
%%%%%%%
%%%%%%%%
%%%%%%%%%
%%%%%%%%%%  !!!THIS SECTION FOR DEBUGGING AND RUNNING SPECIFIC CASES
% Ratest=0;
```

```
% Ra2test=90000:-1000:0;
% len=length(Ra2test);
% Ra1test=0;
% alpha=20;
% figure(10);
% %[A,B]=getABtern3layer(alpha,Ratest,Ra1test,Ra2test,...parameter list ...);
% for k=1:len
% [A,B] = getABtern3layer(alpha,Ra1test,Ra2test(k),Ratest,Da,dBdzBL,deltabsq,...
%     deltacsq,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
%     dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
%     dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
%     d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,Vel,...
%     chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
%     CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PHIBBS,DS2,...
%     chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
%     XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
%     d2BdzBSdiag);
%     [V,D]=eig(A,B);
% % figure(12);
% % plot(real(diag(D)),imag(diag(D)),'rh');hold on;
%     [eig_sorted,eig_label,beta]=sort_eigs(diag(D));
%     eig_sorted;
%     plot(Ra2test(k),real(eig_sorted(1)),'bs');hold on;
% end
% %figure(13);
% %spy(A);
```

```
% %figure(14);
% %spy(B);
%return;
%%%%%%%%%
%%%%%%%%
%%%%%%%
%%%%%%
%%%%%
%%%%
%%%
%%
%
%
% set min and max wavenumbers to scan through
% (set them the same to compute a specific, single wavenumber)

modewatch=1; % normally set this to = 1 for tracking the eigenvalue
             % with the largest real part.  Set = 2 for second eigenvalue
             % with the second largest real part, etc.
alphamin=0.2;
alphamax=2.0;
%
%  set numalpha to be the number of wavenumbers values to calculate on
%  the interval [alphamin,alphamax]
%
```

```
numalpha=20;  % the smallest this can be is 2 ...
%
%  set min and max Rayleigh number bracket
%
Rmin=0;
Rmax=200000000;
tol=10^(-8);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%  start the loop through different wavenumber values   %%%%%%%%%%%
%%%%%%%%%  using bisection to zero out the real part of the     %%%%%%%%%%%
%%%%%%%%%  numbered 'modewatch'                                 %%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Ra=0;
RaB=0;
for ia=1:numalpha
    alpha=alphamin+(alphamax-alphamin)*(ia-1)/(numalpha-1);
    a2=alpha^2;
    Rlow=Rmin;
    Rhigh=Rmax;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% get omegamax_min (the largest real(omega) for R=Rmin)
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% need to decide which Rayleigh numbers to fix and which ones to vary !!!!!
    RaC=Rmin;
    %[A,B]=getAB3layer(alpha,Ra,RaB,RaC,...parameter list...);
    [A,B] = getABtern3layer(alpha,RaB,RaC,Ra,Da,dBdzBL,deltabsq,...
    deltacsq,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
    dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
    dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
    d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,Vel,...
    chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
    CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PHIBBS,DS2,...
    chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
    XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
    d2BdzBSdiag);
    [V,D]=eig(A,B);
%
%   The following function -- sort_eigs -- sorts the eigenvalues
%   for these two cases (largest to smallest), keeping track of their
%   original locations so that the corresponding eigenfunctions can be
%   retrieved and identifies the infinite eigenvalues (identified
%   by beta=0).
%
    [eig_sorted,eig_label,beta]=sort_eigs(diag(D));
%
```

```
%   Identify specifically the max eigenvalue and corresponding
%   eigenfunction location for the first Rmin case ... (this gets
%   repeated below for the Rmax case and then the general Rm case
%   within the bisection code.
%
    omegamax_min=eig_sorted(modewatch);
    Kval=eig_label(modewatch);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% get omegamax_max (the largest real(omega) for R=Rmax)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% need to decide which Rayleigh numbers to fix and which ones to vary !!!!!
    RaC=Rmax;
    %[A,B]=getAB3layer(alpha,Ra,Ra1,Ra2,...parameter list...);
    [A,B] = getABtern3layer(alpha,RaB,RaC,Ra,Da,dBdzBL,deltabsq,...
    deltacsq,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
    dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
    dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
    d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,Vel,...
    chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
    CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PHIBBS,DS2,...
    chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
    XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
```

```
    d2BdzBSdiag);
    [V,D]=eig(A,B);
%
%   The following function -- sort_eigs -- sorts the eigenvalues
%   for these two cases (largest to smallest), keeping track of their
%   original locations so that the corresponding eigenfunctions can be
%   retrieved and identifies the infinite eigenvalues (identified
%   by beta=0).
%
    [eig_sorted,eig_label,beta]=sort_eigs(diag(D));
%
%   Identify specifically the max eigenvalue and corresponding
%   eigenfunction location for the first Rmin case ... (this gets
%   repeated below for the Rmax case and then the general Rm case
%   within the bisection code.
%
    omegamax_max=eig_sorted(modewatch);
    Kval=eig_label(modewatch);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
    if sign(real(omegamax_max))==sign(real(omegamax_min))
       display('WARNING: BRACKET NOT FOUND FOR BISECTION')
       omegamax_max
       omegamax_min
       return;
    end
```

```
%
    k=1;
    Rmid(1)=Rlow+(Rhigh-Rlow)/2;
    err(1)=(Rhigh-Rlow)/2;
    while err(k)>tol
          k=k+1;
          Rm=Rlow+(Rhigh-Rlow)/2;
% need to decide which Rayleigh numbers to fix and which ones to vary !!!!!
          RaC=Rm;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% get omegamax (the largest real(omega) for RaT=Rm)
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 %         [A,B]=getAB3layer(alpha,Ra,Ra1,Ra2,...parameter list...);
 [A,B] = getABtern3layer(alpha,RaB,RaC,Ra,Da,dBdzBL,deltabsq,...
    deltacsq,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
    dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
    dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
    d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,Vel,...
    chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
    CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PHIBBS,DS2,...
    chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
    XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
```

```
    d2BdzBSdiag);
  [V,D]=eig(A,B);
%
%   The following function -- sort_eigs -- sorts the eigenvalues
%   for these two cases (largest to smallest), keeping track of their
%   original locations so that the corresponding eigenfunctions can be
%   retrieved and identifies the infinite eigenvalues (identified
%   by beta=0).
%
          [eig_sorted,eig_label,beta]=sort_eigs(diag(D));
%
%   Identify specifically the max eigenvalue and corresponding
%   eigenfunction location for the first Rmin case ... (this gets
%   repeated below for the Rmax case and then the general Rm case
%   within the bisection code.
%
           omegamax=eig_sorted(modewatch);
           Kval=eig_label(modewatch);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% actual bisection step %%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
           if sign(real(omegamax))==sign(real(omegamax_min))
              Rlow=Rm;
              omegamax_min=omegamax;
           else
              Rhigh=Rm;
```

```
            omegamax_max=omegamax;
         end
         Rmid(k)=Rm;
         err(k)=(Rhigh-Rlow)/2;
    end % bisection (while) loop
%
    Kval_rec(ia)=Kval
    alpha_rec(ia)=alpha
    Ra_rec(ia)=Rm  % this could represent RaB, RaC or Ra
    omR_rec(ia)=real(omegamax);
    omI_rec(ia)=imag(omegamax);
end  % alpha loop
%
% test the computed Rayleigh numbers over the given range in alpha to
% identify the critical values alphacrit, Racrit, Ra1crit, Ra2crit
% omIcrit and omRcrit
%
% !!!!! maybe need to use min(Ra_rec) ?????
[Ra_BIG,ci]=min(Ra_rec);    % Ra_BIG = maximum discrete Rayleigh number
                            % ci = index of critical value
%
% apply a local parabolic fit to identify the actual critical values
%
if ci ~= length(alpha_rec);
   if ci ~= 1;
      alphaMAT=[alpha_rec(ci-1)^2 alpha_rec(ci-1) 1;...
```

```
                alpha_rec(ci)^2 alpha_rec(ci) 1;...
                alpha_rec(ci+1)^2 alpha_rec(ci+1) 1];
      Rvec=[Ra_rec(ci-1);Ra_rec(ci);Ra_rec(ci+1)];
      xABC=alphaMAT\Rvec;
      alphacrit=-xABC(2)/(2*xABC(1))
      RaCcrit=xABC(3)-xABC(2)^2/(4*xABC(1))
      RaBcrit=RaB;                      % note Ra is fixed
      Racrit=Ra;                    % note RaC is fixed
%
%  with the critical values, resolve to get omega, etc.
%
      a2crit=alphacrit^2;
%      [A,B]=getAB3layer(alphacrit,Racrit,Ra1crit,Ra2crit,...parameter list...);
   [A,B] = getABtern3layer(alphacrit,RaBcrit,RaCcrit,Racrit,Da,dBdzBL,deltabsq,...
    deltacsq,dCdzBL,dBdzBP,dCdzBP,DL2,DL3,DL4,NL,NP,NS,IL,IP,MB,MC,...
    dTdzBPdiag,dBdzBLdiag,dCdzBLdiag,dTdzBLdiag,DP2,S,PIstuffPtimesDP,...
    dBdzBPdiag,dXdzBPdiag,XBPdiag,BBPdiag,d2BdzBPdiag,XBP,d2TdzBL,...
    d2TdzBP,d2BdzBL,d2CdzBL,MBC,MCC,IS,DL,DP,DS,PIstuffStimesDS,Vel,...
    chiCubedBPdiag,BBL,dXdzBP,d2BdzBP,BBP,CBL,d2CdzBP,...
    CBP,dCdzBPdiag,CBPdiag,d2CdzBPdiag,dBdzBS,dCdzBS,XBS,PHIBBS,DS2,...
    chiCubedBSdiag,BBSdiag,dXdzBS,dPdzBS,d2CdzBS,dCdzBSdiag,dXdzBSdiag,...
    XBSdiag,CBSdiag,d2CdzBSdiag,d2TdzBS,dTdzBSdiag,d2BdzBS,dBdzBSdiag,...
    d2BdzBSdiag);
      [V,D]=eig(A,B);
      [eig_sorted,eig_label,beta]=sort_eigs(diag(D));
      omegamaxATCRIT=eig_sorted(modewatch);
```

```
      KvalATCRIT=eig_label(modewatch);
%
      omIcritATCRIT=imag(omegamaxATCRIT);
      omRcritATCRIT=real(omegamaxATCRIT);
%
% call m-file that plots results at the critical wavenumber
%
      eigfunctionATCRIT_plotter_V;


      streamfunctionATCRIT_plotter_V;
   else
      display('NO CRITICAL POINT IN WAVENUMBER BRACKET (min at left)')
   end
else
display('NO CRITICAL POINT IN WAVENUMBER BRACKET (min at right)')
end
%
% call m-file that plots various bits of these results
%
eigfunction_plotter_V;


streamfunction_plotter_V;


return;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% rhsBCodePri.m
%
% Function to compute the right hand side of the ODEs for B and C in the
% primary mushy layer.
%
% This system of coupled ODEs is solved using ode23.
%
% INPUTS: z........a position within the secondary musy layer
%
%         BCvals...liquid compositions of B and C in the primary mush
%
%         BPguess...guessed value for B at the top of the primary mushy
%                   layer
%
%         CPguess...guessed value for C at the top of the primary mushy
%                   layer
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: vector of B and C values in primary mushy layer at z
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function rhsBC = rhsBCodePri(z, BCvals, BPguess, CPguess, paramVec)
% dB/dz = deltabsq(Binfty/chi - B)
```

```
% dC/dz = deltacsq(Cinfty/chi - C)

% separate out phase diagram constants
deltabsq = paramVec(4);
deltacsq = paramVec(5);
Binfty = paramVec(6);
Cinfty = paramVec(8);

% in order to compute B and C at z, we need the liquid fraction at z
chi = chiPrimary(BCvals(1), BPguess, BCvals(2), CPguess, paramVec);

% compute B at z
rhsBC(1,1) = deltabsq * (Binfty / chi - BCvals(1));

% compute C at z
rhsBC(2,1) = deltacsq * (Cinfty / chi - BCvals(2));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% rhsBodeSec.m
%
% Function to compute the right hand side of the ODE for B in the secondary
% mushy layer.
%
% INPUTS: z...position within the secondary musy layer
%
%         B...liquid composition of B
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: rhs...value of B at current positiion z
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function rhs = rhsBodeSec(z, B, paramVec)
% dB/bz = -delta_b^2(B + phi_b/chi - B_inf/chi)


% separate out phase diagram constants
deltabsq = paramVec(4);
Binfty = paramVec(6);


% determine chi (liquid fraction) for current value of B
chi = chiSecondary(B, paramVec);
```

```
% solid fraction of B in the secondary mushy layer
phi_b = phiBsecondary(B, chi, paramVec);

% solve right hand side of ODE for B in the secondary mushy layer
rhs = -deltabsq * (B + phi_b / chi - Binfty / chi);
```

```
function [barzp,zp,DP,DP2,DP3,DP4,IP]=setup_cheb(NP,Htop,Hbot);
%  edited (6-6-08, DMA)
%  This is the Matlab function file that sets up
%  the basic grid and Chebyshev differentiation
%  matrices.  It is called by pm_three_layer.m.
%
%
% Chebyshev Points:  bar{z}=cos(j*pi/N), j=0,1,2,...,N
%
barzp=cos([0:NP]*pi/NP);
%
% rescalings for mushy layer thickness in terms of Chebyshev scaling
%
LP=(Htop-Hbot)/2;
AP=1/LP;
zp=Htop+LP*(barzp-1);
%
% define chebyshev differentiation matrix:
%
%[D,x]=cheb(N);D2=D^2;D3=D^3;D4=D^4;
%
[DP,x]=cheb(NP);
DP=AP*DP;
DP2=DP^2;
DP3=DP^3;
DP4=DP^4;
```

```
%
% define identity matrix for use in setting up matrices
%
IP=eye(NP+1);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% solveMush.m
%
% Main driver used to compute model parameters within the mushy layers.
%
% INPUTS: guessVec...three element vector of guesses for hS, hP and BP
%
%         paramVec...vector of ternary phase diagram constants
%
% OUTPUTS: zSec...vector of z values through secondary mushy layer
%
%          Bsec...liquid composition of B through secondary mushy layer
%
%          zPri...vector of z values throuhg primary mushy layer
%
%          Bpri...liquid composition of B through primary mushy layer
%
%          Cpri...liquid composition of C through primary mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [zSec,Bsec,zPri,Bpri,Cpri] = solveMush(guessVec, paramVec)


% separate out phase diagram constants
BE = paramVec(1);
TE = paramVec(2);
```

```
CE = paramVec(3);

deltabsq = paramVec(4);

deltacsq = paramVec(5);

Binfty = paramVec(6);

Tinfty = paramVec(7);

Cinfty = paramVec(8);

gamma = paramVec(9);

gammab = paramVec(10);

BEAB = paramVec(11);

TEAB = paramVec(12);

TM = paramVec(13);

mB = paramVec(14);

mC = paramVec(15);

mBC = paramVec(16);

mCC = paramVec(17);

mBbar = paramVec(18);

mCbar = paramVec(19);


% decrease the relative and absolute error tolerances for ode23

options = odeset('RelTol',1e-5,'AbsTol',1e-8);


% determine unknown quantities (hS, hP and BP)

unknowns = fsolve(@findUnknowns, guessVec, [], paramVec);


% determine model variable values using previously unknown quantities

% (first solve ODE for B in the secondary mushy layer)
```

```
[zSec, Bsec] = ode23(@rhsBodeSec, [0 unknowns(1)], BE, options, paramVec);

deltaTB = 1/((TM-Tinfty)/mB+Binfty+(mCbar/mBbar)*Cinfty);

deltaTC = 1/((TM-Tinfty)/mC+Cinfty+(mBbar/mCbar)*Binfty);

% using guessed value for BP, find CP
CPguess = (-deltaTB*(deltabsq-1)*(unknowns(3)-Binfty))/(deltaTC*(deltacsq-1)) ...
    + Cinfty + 1/(deltaTC*(deltacsq-1));

% read out the last entry from the B values (BS)
BScomp = Bsec(length(Bsec));

% using BS, next find CS
CScomp = (mBC / mCC) * (BScomp - BEAB);

% next solve ODEs for B and C in the primary mushy layer
[zPri, BCvalsPri] = ode23(@rhsBCodePri, [unknowns(1), unknowns(2)], ...
    [BScomp, CScomp], options, unknowns(3), CPguess, paramVec);

Bpri = BCvalsPri(:,1);

Cpri = BCvalsPri(:,2);
```

```
function [eig_vec_ordered,eig_label_vec,beta]=sort_eigs(DD);
%  edited (9-25-07, DMA)
%  This is the Matlab function file that sorts the
%  eigenvalues in terms of their real parts (largest to smallest)
%  from a list of eigenvalues DD subject to omission of infinite
%  eigenvalues (determined by whether or not 1/D(kk)==0).
%
%  This function file returns
%
%   eig_vec_ordered = eigenvalues ordered from maximum real part
%                     to minimum real part ... the non-infinite
%                     ones are stored at the beginning of the
%                     vector so, for example, eig_vec_ordered(1)
%                     should always contain the largest non-infinite
%                     eigenvalue.
%
%   eig_label_vec = vector list of addressess for the original
%                   location of the eigenvalues.  This is used
%                   in the calling code to identify the correct
%                   eigenfunction associated with each (now shuffled)
%                   eigenvalue.
%
%   beta = this is a vector with entry 1 if the eigenvalue was finite
%                                     0 if the eigenvalue was infinite
%
betavec=1./DD;
```

```
sizeDD=length(DD);
eig_vec_ordered=zeros(sizeDD,1);
eig_label_vec=zeros(sizeDD,1);
beta=zeros(sizeDD,1);
%
[yvec,ivec]=sort(-real(DD));
i=0;
ib=sizeDD+1;
for kk=1:sizeDD
    j=ivec(kk);
%    if betavec(j)~=0;     % only interested in non-infinite eigenvalues
    if DD(j) < 10^(7)
       i=i+1;
       eig_vec_ordered(i,1)=DD(j);
       eig_label_vec(i,1)=j;
       beta(i,1)=1;
   else
       ib=ib-1;
       eig_vec_ordered(ib,1)=0;
       eig_label_vec(ib,1)=j;
       beta(ib,1)=0;
   end
end
```

```
% this file called from pm_dd_bisection_V.m ... it plots streamfunctions
% at the critical Rayleigh number and wavenumber
fprintf 'Plotting streamfunction...\n'


Nplot=NL+NP+NS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%          PROCESS AND PLOT RESULTS          %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% primary layer vertical velocity perturbation
%
%wpstart=1;
%wpstop=Nplot+1;
%wp_vec=V(wpstart:wpstop,KvalATCRIT);
%
%wpREAL=real(wp_vec);
%wpIMAG=imag(wp_vec);
%
liqtot=4*(NL+1);
pritot=5*(NP+1)+1;
jSL = 1:NL+1;
jSP = (2:NP+2) + liqtot;
jSS=(2:NS+2)+liqtot+pritot;
%
```

```
psiLIQ=V(jSL,Kval);
psiPRI=V(jSP,Kval);
psiSEC=V(jSS,Kval);
PSI=[psiLIQ;psiPRI;psiSEC];
%
PSIr=real(PSI);
PSIi=imag(PSI);
%
figure(9)
%
% set up x-z plane
%
   XMIN=0;
   XMAX=4;              %   4*pi/alphacrit;
   DELTAX=0.005;
   xp = [XMIN:DELTAX:XMAX];
   [XPPLOT,ZPPLOT]=meshgrid(xp,[zl,zp,zs]);
%
if imag(omegamax) == 0   % real mode
%
% define z-dependent part of streamfunction
%
   for i=1:length(xp)
       %WPHAT(:,i)=wpREAL;
       PSIHAT(:,i)=PSIr;
   end
```

```
   %PSIP=(2/alphacrit)*sin(alphacrit*XPPLOT).*WPHAT;
   PSIP=-2*(PSIHAT.*sin(alpha*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
xlabel('x','FontSize',16);ylabel('z','FontSize',16)
title('Streamfunction','FontSize',16)
%
else  % oscillatory mode
   for i=1:length(xp)
       %WPHATr(:,i)=wpREAL;
       %WPHATi(:,i)=wpIMAG;
       PSIHATr(:,i)=PSIr;
       PSIHATi(:,i)=PSIi;
   end
%%%%
   subplot(2,2,1);
   t=0;
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                     WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(imag(omegamax)*t + alpha*XPPLOT) + ...
                        PSIHATr.*sin(imag(omegamax)*t + alpha*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t=0','FontSize',16)
%%%%
```

```
   subplot(2,2,2)
   t=pi/(2*imag(omegamax));
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(imag(omegamax)*t + alpha*XPPLOT) + ...
                         PSIHATr.*sin(imag(omegamax)*t + alpha*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= \pi/2','FontSize',16)
%%%%
   subplot(2,2,3)
   t=pi/(imag(omegamax));
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(imag(omegamax)*t + alpha*XPPLOT) + ...
                         PSIHATr.*sin(imag(omegamax)*t + alpha*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= \pi','FontSize',16)
%%%%
   subplot(2,2,4)
   t=3*pi/(2*imag(omegamax));
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
```

```
   PSIP=-2*(PSIHATi.*cos(imag(omegamax)*t + alpha*XPPLOT) + ...
                        PSIHATr.*sin(imag(omegamax)*t + alpha*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= 3\pi/2','FontSize',16)
end
%
print -deps pm_streampertATCRIT.eps
```

```
% this file called from pm_dd_bisection_V.m ... it plots streamfunctions
% at the critical Rayleigh number and wavenumber
fprintf 'Plotting streamfunction...\n'


Nplot=NL+NP+NS;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%          PROCESS AND PLOT RESULTS          %%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% primary layer vertical velocity perturbation
%
%wpstart=1;
%wpstop=Nplot+1;
%wp_vec=V(wpstart:wpstop,KvalATCRIT);
%
%wpREAL=real(wp_vec);
%wpIMAG=imag(wp_vec);
%
liqtot=4*(NL+1);
pritot=5*(NP+1)+1;
jSL = 1:NL+1;
jSP = (2:NP+2) + liqtot;
jSS=(2:NS+2)+liqtot+pritot;
%
```

```
psiLIQ=V(jSL,KvalATCRIT);
psiPRI=V(jSP,KvalATCRIT);
psiSEC=V(jSS,KvalATCRIT);
PSI=[psiLIQ;psiPRI;psiSEC];
%
PSIr=real(PSI);
PSIi=imag(PSI);
%
figure(9)
%
% set up x-z plane
%
   XMIN=0;
   XMAX=4;              %   4*pi/alphacrit;
   DELTAX=0.005;
   xp = [XMIN:DELTAX:XMAX];
   [XPPLOT,ZPPLOT]=meshgrid(xp,[zl,zp,zs]);
%
if omIcritATCRIT == 0   % real mode
%
% define z-dependent part of streamfunction
%
   for i=1:length(xp)
       %WPHAT(:,i)=wpREAL;
       PSIHAT(:,i)=PSIr;
   end
```

```
   %PSIP=(2/alphacrit)*sin(alphacrit*XPPLOT).*WPHAT;
   PSIP=-2*(PSIHAT.*sin(alphacrit*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
xlabel('x','FontSize',16);ylabel('z','FontSize',16)
title('Streamfunction At Critical Rayleigh Number','FontSize',16)
%
else  % oscillatory mode
   for i=1:length(xp)
       %WPHATr(:,i)=wpREAL;
       %WPHATi(:,i)=wpIMAG;
       PSIHATr(:,i)=PSIr;
       PSIHATi(:,i)=PSIi;
   end
%%%%
   subplot(2,2,1);
   t=0;
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                     WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
                        PSIHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t=0','FontSize',16)
%%%%
```

```
   subplot(2,2,2)
   t=pi/(2*omIcritATCRIT);
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
                        PSIHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= \pi/2','FontSize',16)
%%%%
   subplot(2,2,3)
   t=pi/(omIcritATCRIT);
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   PSIP=-2*(PSIHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
                        PSIHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= \pi','FontSize',16)
%%%%
   subplot(2,2,4)
   t=3*pi/(2*omIcritATCRIT);
   %PSIP=(2/alphacrit)*(WPHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
   %                      WPHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
```

```
   PSIP=-2*(PSIHATi.*cos(omIcritATCRIT*t + alphacrit*XPPLOT) + ...
                        PSIHATr.*sin(omIcritATCRIT*t + alphacrit*XPPLOT));
   contour(XPPLOT,ZPPLOT,PSIP);
%
   xlabel('x','FontSize',16);ylabel('z','FontSize',16)
   title('Streamfunction: \sigma_I t= 3\pi/2','FontSize',16)
end
%
print -deps pm_streampertATCRIT.eps
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% tempPrimary.m
%
% Function to compute the temperature profile through the primary mushy
% layer for the given input values of B and C.
%
% INPUTS: B...vector of B composition values
%
%         C...vector of C composition values
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: temp...temperature profile through the primary mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function temp = tempPrimary(B,C,paramVec)


% separate out phase diagram constants
TM = paramVec(13);
mB = paramVec(14);
mC = paramVec(15);


temp = TM + mB .* B + mC .* C;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% tempSecondary.m
%
% Function to compute the temperature profile through the secondary mushy
% layer for the given values of B input.
%
% INPUTS: B...vector of B composition values
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: temp...temperature profile through the secondary mushy layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function temp = tempSecondary(B, paramVec)


% separate out phase diagram constants
BE = paramVec(1);
TE = paramVec(2);
mBC = paramVec(16);


temp = -mBC .* (B - BE) + TE;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Tliquid.m
%
% Function to compute the temperature profile through the liquid layer.
%
% INUPTS: TP...temperature at the liquid-mush interface
%
%         hP...position of the liquid-mush interface
%
%         zLiq...vector of z (position) values in the liquid layer at
%                which to evaluate the temperature
%
%         paramVec...vector of phase diagram constants
%
% OUTPUTS: temperature profile vector through the liquid layer
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T = Tliquid(TP,hP,zLiq,paramVec)


% separate out phase diagram constants
Tinfty = paramVec(7);


T = Tinfty + (TP - Tinfty) .* exp(hP - zLiq);
```

Bibliography

Bibliography

[1] A. Aitta, H.E. Huppert, and M.G. Worster, *Diffusion-controlled solidification of a ternary melt from a cooled boundary*, J. Fluid Mech. **432** (2001), 201–217.

[2] D.M. Anderson, *A model for diffusion-controlled solidification of ternary alloys in mushy layers*, J. Fluid Mech. **483** (2003), 165–197.

[3] D.M. Anderson and T.P. Schulze, *Linear and nonlinear convection in solidifying ternary alloys*, J. Fluid Mech. **545** (2005), 213–243.

[4] S.H. Davis, *Theory of solidification*, Cambridge University Press, 2001.

[5] A.F. Thompson, H.E. Huppert, M.G. Worster, and A. Aitta, *Solidification and compositional convection of a ternary alloy*, J. Fluid Mech. **497** (2003), 167–199.

[6] L.N. Trefethen, *Spectral methods in matlab*, Society for Industrial and Applied Mathematics, 2000.

[7] M.G. Worster, *Solidification of an alloy from a cooled boundary*, J. Fluid Mech. **167** (1986), 481–501.

[8] ———, *Natural convection in a mushy layer*, J. Fluid Mech. **224** (1991), 335–359.

[9] ———, *Instabilities of the liquid and mushy regions during solidification of alloys*, J. Fluid Mech. **237** (1992), 649–669.

[10] ———, *Interfaces on all scales during solidification and melting*, Interfaces for the Twenty-First Century (M.K. Smith, M.J. Miksis, G.B. McFadden, G.P. Neitzel, and D.R. Canright, eds.), Imperial College Press, 2002, pp. 187–201.

Curriculum Vitae

Terrance J. Flynn, Jr. received a Bachelor of Science from Southern Illinois University in 2000.