This is to certify that the

dissertation entitled

Numerical Simulations of Waves in a

Magnetically Structured Atmosphere

presented by

Thomas Peter Espinola

has been accepted towards fulfillment
of the requirements for

Ph.D. degree in Physics

Major professor

Date 18 April 1989

Numerical Simulations of Waves in a Magnetically Structured Atmosphere

by

Thomas Peter Espinola

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Physics and Astronomy

1988

Abstact

Numerical Simulations of Waves in a Magnetically Structered Atmosphere

by
Thomas Peter Espinola

A physical model for simulating waves in a stellar atmosphere was developed from a combination of basic fluid mechanics, plasma physics, and electrodynamics. The model was three dimensional and included the effects of gravity, magnetic fields, and viscosity. An algorithm was developed to numerically implement this model. The resulting program used an explicit time integration scheme based on Runge-Kutta and a combination of finite difference and spectral methods to evaluate the spatial derivatives. A number of numerical boundary conditions were developed— the most successful used a modified Sommerfeld radiation condition. The program was written and coded in Fortran on a Vax computer. Additional routines were written to evaluate the required fast fourier transforms and to graph and display the data. The program was tested on a large number of one and two dimensional problems for which the solutions were known. These problems included acoustic waves, Alfvén waves, magnetoacoustic waves, shocks, rarefactions, and contact discontinuities. The numerical results agreed with the analytic solutions of the physical problems to within the precision requested of the simulation. The program proved to be stable and robust for all the problems attempted.

This program was then used to simulate three problems for which analytic solutions are not known. All three simulations concerned the propagation of waves in magnetically structured atmospheres and may be applied to outstanding problems in solar physics. First, the interactions of non-linear waves and a flux slab were studied. From the result it is apparent that sources of shocks and rarefactions, such

as the solar convection zone, do not concentrate the magnetic field in flux sheaths. Next I used the program to simulate the interaction of non-linear waves with a flux tube. The results suggest that the magnetic fields in flux tubes are also not concentrated by pairs of passing shocks and rarefactions; however, a complete demonstration of this would require a finer grid than was practical with the available computing power. The last and most interesting problem simulated was the interaction of oblique acoustic waves and a plane magnetic interface. The results show that while a region of the interface does remove energy from the waves by resonant heating, resonant heating within a sunspot-photosphere interface cannot account for the observed absorption of solar p-mode waves. Sufficient transmission of energy occurs so that the results are consistent with the observations if the absorption takes place by another mechanism in the sunspot interior. The simulation suggests that a two dimensional fine magnetic structure may be capable of absorbing the observed fraction of wave energy.

Contents

1
Physical Theory

The basic physical theory, developed in the following chapter, is a combination of fluid mechanics, plasma physics, and electrodynamics. The theory is greatly simplified by a number of assumptions about the atmosphere. The resulting eight partial differential equations give a complete local description of the behavior of the gas and fields.

1.1 Electrodynamics

The problems of interest concern the electrodynamics of moving media; therefore, two frames of reference are of primary importance. The laboratory frame of reference is at rest relative to the observer while the co-moving frame is at rest relative to the moving media. Quantities in the laboratory frame will be represented by unprimed variables and those in the co-moving frame by primed variables. For a complete glossary of symbols, see Appendix A.

First, consider the Maxwell equations which hold for either frame:

$$\nabla \cdot \mathbf{D} = \rho_e \tag{1.1a}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.1b}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.1c}$$

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \tag{1.1d}$$

where **D** is the electric flux density in coulombs/meter, ρ_e is the charge density in coulombs/m^3, **E** is the electric field in volts/meter, **B** is the magnetic flux density in Teslas, **H** is the magnetic field in amperes/meter, and **J** is the current density in amperes/$meter^2$. A well posed problem requires three additional equations. I will be considering only linear, isotropic media, for which Ohm's holds in the co-moving frame

$$\mathbf{J}' = \sigma \mathbf{E}' \tag{1.2}$$

where σ is the conductivity of the medium in mho/meter. The constituent equations also hold only in the co-moving frame,

$$\mathbf{D}' = \varepsilon \mathbf{E}' \tag{1.3a}$$

$$\mathbf{B}' = \mu \mathbf{H}' \tag{1.3b}$$

where ε is the permitivity in farads/meter and μ is the permeability in henrys/meter.

These seven equations interrelate the electromagnetic field quantities and seem to be uncoupled from the motion of the fluid; however, **J** may depend explicitly on the velocity of the medium. The motion of the fluid is coupled to the fields through the Lorentz force equation, which is covariant,

$$\mathbf{F} = q(\mathbf{E} + \mathbf{V} \times \mathbf{B}) \tag{1.4}$$

For the problems of interest, one may make the following simplifying assumptions, which are known as the magnetohydrodynamic, or MHD, approximations (Hughes & Young, 1966, §6.3):

1. The fluid velocities will remain nonrelativistic. One may then replace γ with 1 in the Lorentz transforms. Therefore,

$$\mathbf{E}' = \mathbf{E} + \mathbf{V}\times\mathbf{B} \qquad (1.5a)$$

$$\mathbf{D}' = \mathbf{D} + \frac{\mathbf{V}\times\mathbf{H}}{c^2} \qquad (1.5b)$$

$$\mathbf{H}' = \mathbf{H} - \mathbf{V}\times\mathbf{D} \qquad (1.5c)$$

$$\mathbf{B}' = \mathbf{B} - \frac{\mathbf{V}\times\mathbf{E}}{c^2} \qquad (1.5d)$$

$$\mathbf{J}' = \mathbf{J} - \rho_e\mathbf{V} \qquad (1.5e)$$

$$\rho_e' = \rho_e - \frac{\mathbf{V}\cdot\mathbf{J}}{c^2} \qquad (1.5f)$$

2. The conductivity is very large so that $\mathbf{E}' \simeq 0$. The Lorentz transform then yields $\mathbf{E} \simeq -\mathbf{V}\times\mathbf{B}$ and $\mathbf{D} \simeq -\mathbf{V}\times\mathbf{H}/c^2 \simeq 0$.

3. The permitivity and permeability equal those of free space. Both these assumptions are very good for astrophysical plasmas. The assumption that $\varepsilon=\varepsilon_0$ sometimes breaks down for other uses of the MHD approximations.

4. The induced magnetic fields are small compared to the external fields. This assumption, which follows for (1), yields $\mathbf{B}' = \mathbf{B}$ as follows

$$\mathbf{B}' = \mathbf{B} - \frac{\mathbf{V}\times\mathbf{E}}{c^2} \simeq \mathbf{B} - \frac{\mathbf{V}\times(-\mathbf{V}\times\mathbf{B})}{c^2} \simeq \mathbf{B} + \mathrm{Or}\left(\frac{v^2}{c^2}\right)$$

$$\mathbf{B}' \simeq \mathbf{B}$$

   Since $\mu=\mu_0$, $\mathbf{H} = \mathbf{H}'$ follows directly. For another permeability, $\mathbf{H} = \mathbf{H}'$ may still be a good approximation.

5. The displacement current may be neglected as compared to the conduction current. Therefore, high frequency phenomena cannot be considered.

6. The current densities in the laboratory and co-moving frames are equal since

$$\mathbf{J}' = \mathbf{J} - \rho_e\mathbf{V} = \mathbf{J} - \left(\rho_e' + \frac{\mathbf{V}\cdot\mathbf{J}}{c^2}\right)\mathbf{V} \simeq \mathbf{J} - \mathrm{Or}\left(\mathbf{J}\,\frac{v^2}{c^2}\right)$$

   Also, the charge density in the co-moving frame, ρ'_e, vanishes since the medium is a conductor. Thus

$$\mathbf{J}' = \mathbf{J} = \sigma(\mathbf{E} + \mathbf{V}\times\mathbf{B})$$

   Moreover, σ is assumed constant and independent of frequency, **B**, and **E** but not necessarily independent of temperature or other fluid variables.

The Maxwell equations with the MHD approximations thus may be written as:

$$\nabla \cdot \mathbf{D} = \frac{\nabla \cdot \mathbf{J}}{c^2} \tag{1.6a}$$

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \tag{1.6b}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{1.6c}$$

$$\nabla \times \mathbf{H} = \mathbf{J} \tag{1.6d}$$

and Ohm's law is

$$\mathbf{J} = \sigma(\mathbf{E} + \mathbf{V} \times \mathbf{B}) \tag{1.6e}$$

Using these and the constituent equations allows one to solve for the time evolution of any of the fields. Of most interest here is the evolution of the magnetic field. Solving Ohm's law for **E** yields

$$\mathbf{E} = \frac{\mathbf{J}}{\sigma} - \mathbf{V} \times \mathbf{B}$$

Substituting $\nabla \times \mathbf{H}$ for **J**

$$\mathbf{E} = \frac{\nabla \times \mathbf{H}}{\sigma} - \mathbf{V} \times \mathbf{B}$$

Using this in the curl **E** equation (1.6b) yields

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{V} \times \mathbf{B}) - \nabla \times \frac{\nabla \times \mathbf{H}}{\sigma}$$

Therefore, one gets the following for the time evolution of **B**:

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{V} \times \mathbf{B}) - \nabla \times (\eta \nabla \times \mathbf{B}) \tag{1.7a}$$

where $\eta = 1/\sigma\mu_0$ is the magnetic diffusivity. This equation may be simplified when written using the Einstein summation convention.

$$\frac{\partial B_i}{\partial t} + \frac{\partial}{\partial x_j}\left[B_i v_j - B_j v_i + \eta\left(\frac{\partial B_j}{\partial x_i} - \frac{\partial B_i}{\partial x_j}\right)\right] = 0 \tag{1.7b}$$

If η is independent of position, one may use the identity

$$\nabla \times \nabla \times \mathbf{B} = \nabla(\nabla \cdot \mathbf{B}) - \nabla^2 \mathbf{B}$$

to get

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{V} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B} \tag{1.7c}$$

Equation 1.7 fully specifies the electrodynamics under the MHD approximations.

1.2 Fluid Equations

A complete description of the MHD system requires equations describing the behavior of the material particles as well as those describing the electromagnetic fields. The media of primary interest are astrophysical plasmas. The plasma equations can be quite complicated when the electrons, ions and neutral particles are treated separately and the binary interactions are considered in detail. For simplicity, I will consider only ideal plasmas which may be described by one-fluid equations (Krall & Trivelpiece, 1973, §3.5). The equations describing such a plasma reduce to those of a fluid with a local charge density.

First consider the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0 \qquad (1.8)$$

The validity of this equation may be shown by considering an arbitrary closed, fixed volume, V, within the viscous, compressible fluid. It has a boundary (surface area) ∂V. Let M be the total mass, in kilograms, within this volume. Thus

$$M = \int_V \rho \, d^3x$$

Where ρ is the mass density in kilograms/meter^3. Taking the time derivative of both sides and remembering the volume is fixed,

$$\frac{\partial M}{\partial t} = \int_V \frac{\partial \rho}{\partial t} \, d^3x$$

However, the change of mass in the absence of sources or sinks must result from the flow across the boundary. Thus

$$\frac{\partial M}{\partial t} = \int_{\partial V} -\rho \mathbf{v} \cdot d\mathbf{A}$$

where the negative comes from the convention that $d\mathbf{A}$ is directed outward causing positive flows to decrease the mass within the volume. Equating these two forms for the time change in mass yields

$$\int_V \frac{\partial \rho}{\partial t} \, d^3x = \int_{\partial V} -\rho \mathbf{v} \cdot d\mathbf{A}$$

The integral on the right hand side may be converted to a volume integral using the divergence theorem to yield

$$\int_V \frac{\partial \rho}{\partial t} d^3x = -\int_V (\nabla \cdot \rho \mathbf{v})\, d^3x$$

$$\int_V \left(\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v}\right) d^3x = 0$$

Since the volume is arbitrary, the integrand must vanish everywhere; thus the continuity equation is verified.

To derive the momentum equation, once again consider the closed volume, V. Let **m** be the total momentum in this volume. Thus

$$\mathbf{m} = \int_V \rho \mathbf{v}\, d^3x$$

and

$$\frac{\partial \mathbf{m}}{\partial t} = \int_V \left(\frac{\partial}{\partial t} \rho \mathbf{v}\right) d^3x \qquad (1.9)$$

The change of momentum within the volume, in the absence of sinks or sources for mass, results from the following three sources:

1. External body forces acting within the volume. Let **f** be the sum of all such forces. I will postpone considering the nature of these forces until later. The time rate of change of momentum resulting from these forces is given by
$$\frac{\partial \mathbf{m}}{\partial t}_1 = \int_V \mathbf{f}\, d^3x$$

2. The flow of momentum across the boundary given by
$$\frac{\partial \mathbf{m}}{\partial t}_2 = \int_{\partial V} -(\rho \mathbf{v})\mathbf{v} \cdot d\mathbf{A}$$

3. Pressure and viscous forces acting on the boundary;
$$\frac{\partial \mathbf{m}}{\partial t}_3 = \int_{\partial V} -P\, d\mathbf{A} + \int_{\partial V} \underline{\tau} \cdot d\mathbf{A}$$
where τ is the viscous stress tensor discussed later.

Thus the total change in momentum in the volume is given by

$$\frac{\partial \mathbf{m}}{\partial t} = \int_V \mathbf{f}\, d^3x + \int_{\partial V} -(\rho \mathbf{v})\mathbf{v} \cdot d\mathbf{A} + \int_{\partial V} -P\, d\mathbf{A} + \int_{\partial V} \underline{\tau} \cdot d\mathbf{A}$$

The three surface integrals may be combined to yield

$$\frac{\partial \mathbf{m}}{\partial t} = \int_V \mathbf{f}\, d^3x + \int_{\partial V} -(\rho \mathbf{v}\mathbf{v} + P\underline{\mathbf{1}} - \underline{\tau})\cdot d\mathbf{A}$$

where **1** is the unit tensor (order two). This surface integral may then be converted to a volume integral to yield

$$\frac{\partial \mathbf{m}}{\partial t} = \int_V \mathbf{f}\, d^3x + \int_V -\nabla\cdot(\rho \mathbf{v}\mathbf{v} + P\underline{\mathbf{1}} - \underline{\tau})\, d^3x$$

Equating this to the time rate of change of momentum from equation (1.9) yields

$$\int_V \left[\frac{\partial}{\partial t}\rho \mathbf{v} - \mathbf{f} + \nabla\cdot(\rho \mathbf{v}\mathbf{v} + P\underline{\mathbf{1}} - \underline{\tau})\right] d^3x = 0$$

Once again the integrand must vanish since the volume is arbitrary. This gives the momentum equation

$$\frac{\partial}{\partial t}\rho \mathbf{v} + \nabla\cdot(\rho \mathbf{v}\mathbf{v} + P\underline{\mathbf{1}} - \underline{\tau}) = \mathbf{f} \qquad (1.10)$$

The viscous stress tensor, τ,is defined such that

$$d\mathbf{f}_v = \underline{\tau}\cdot d\mathbf{A}$$

or, using the Einstein summation convention

$$df_i = \tau_{ij} dA_j$$

where $\mathbf{f}_v$ is the viscous force acting on the boundary of the volume. For the form of τ_{ij}, one must rely on a combination of experiment and theory to give (Huang, 1963, pp 113-116)

$$\tau_{ij} = \zeta_1\left[\frac{\partial v_i}{\partial x_i} + \frac{\partial v_j}{\partial x_i} - \frac{2}{3}\delta_{ij}\frac{\partial v_k}{\partial x_k}\right] + \zeta_2\frac{\partial v_k}{\partial x_k}$$

where ξ_1 and ξ_2 are the coefficients of viscosity which may be functions of the fluid variables.

The external forces of primary interest include gravity,

$$\mathbf{f}_g = \rho \mathbf{g} \qquad (1.11)$$

and the magnetic body force

$$\mathbf{f}_m = \mathbf{J}\times\mathbf{B} = -\mathbf{B}\times\mathbf{J} = -\mathbf{B}\times\nabla\times\mathbf{H} = \frac{-1}{\mu_o}\mathbf{B}\times\nabla\times\mathbf{B} \qquad (1.12a)$$

$$\mathbf{f}_m = \frac{(\mathbf{B}\cdot\nabla)\mathbf{B}}{\mu_o} - \frac{\nabla|\mathbf{B}|^2}{2\mu_o} \qquad (1.12b)$$

or in summation notation

$$(f_m)_i = \frac{1}{\mu_o}B_j\frac{\partial B_i}{\partial x_j} - \frac{\partial}{\partial x_i}\frac{B_j B_j}{2\mu_o} \qquad (1.12c)$$

Combining these two external forces gives

$$\mathbf{f} = \rho\mathbf{g} + \frac{(\mathbf{B}\cdot\nabla)\mathbf{B}}{\mu_o} - \frac{\nabla|\mathbf{B}|^2}{2\mu_o} \tag{1.13}$$

The last term in equation (1.13) may be grouped with the pressure on the left side of the momentum equation to yield

$$\frac{\partial}{\partial t}\rho\mathbf{v} + \nabla\cdot\left[\rho\mathbf{v}\mathbf{v} + \left(P + \frac{|\mathbf{B}|^2}{2\mu_o}\right)\underline{\mathbf{1}} - \underline{\mathbf{T}}\right] = \rho\mathbf{g} + \frac{(\mathbf{B}\cdot\nabla)\mathbf{B}}{\mu_o} \tag{1.14}$$

Consideration of the energy equations begins with the first law of thermodynamics

$$dU = \delta q + \delta W$$

which holds in the co-moving frame. U is the internal energy contained in dV, a small volume moving with the fluid. Defining e as the internal energy per unit mass and using -PdV for work yields

$$M de = dq - P\,dV$$

$$de = \frac{dq}{M} - \frac{P\,dV}{M}$$

The dq term includes the energy gained from external forces, viscous damping, joule heating, as well as heat transfer. As the volume is allowed to become smaller and smaller, one may make the following substitutions

$$\rho = \frac{M}{V} \Rightarrow \begin{array}{l} dV = -\dfrac{M}{\rho^2}d\rho \\ \dfrac{dq}{M} = \dfrac{1}{\rho}\dfrac{dq}{V} = \dfrac{1}{\rho}Q \end{array}$$

where Q is the energy input per unit time per unit volume. Combining these gives

$$\frac{de}{dt} = \frac{P}{\rho^2}\frac{d\rho}{dt} + \frac{Q}{\rho} \tag{1.15}$$

as the time rate of change for e. To find the rate of change of e in the laboratory frame, one must use

$$\frac{de}{dt} = \frac{\partial e}{\partial t} + \mathbf{v}\cdot\nabla e$$

and

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \mathbf{v}\cdot\nabla\rho = -\rho\nabla\cdot\mathbf{v}$$

in equation (1.15) to yield

$$\frac{\partial e}{\partial t} + \mathbf{v}\cdot\nabla e = \frac{P}{\rho^2}\left(-\rho\nabla\cdot\mathbf{v}\right) + \frac{Q}{\rho} = -\frac{P}{\rho}\nabla\cdot\mathbf{v} + \frac{Q}{\rho}$$

I am interested in finding the time rate of change of ρe; so,

$$\frac{\partial}{\partial t}\rho e = e\,\frac{\partial \rho}{\partial t} + \rho\,\frac{\partial e}{\partial t} = e\,\frac{\partial \rho}{\partial t} + \rho\left(-\frac{P}{\rho}\nabla\cdot\mathbf{v} + \frac{Q}{\rho} - \mathbf{v}\cdot\nabla e\right)$$

$$\frac{\partial}{\partial t}\rho e = e\,\frac{\partial \rho}{\partial t} - \rho\mathbf{v}\cdot\nabla e - P\,\nabla\cdot\mathbf{v} + Q = e\,\frac{\partial \rho}{\partial t} + e\,\nabla\cdot\rho\mathbf{v} - \nabla\cdot\rho e\mathbf{v} - P\,\nabla\cdot\mathbf{v} + Q$$

From the continuity equation, the first two terms on the right hand side equal zero so

$$\frac{\partial}{\partial t}\rho e = -\nabla\rho e\mathbf{v} - P\,\nabla\cdot\mathbf{v} + Q = -\nabla\rho e\mathbf{v} - \nabla\cdot P\mathbf{v} + \mathbf{v}\cdot\nabla P + Q$$

Thus

$$\frac{\partial}{\partial t}\rho e + \nabla\cdot\left[\mathbf{v}\,(\rho e + P)\right] = \mathbf{v}\cdot\nabla P + Q \qquad (1.16)$$

This equation determines the time rate of change of the internal energy. Q contains the external heating as well as heat transport, so

$$Q = Q_T + Q_{rad} + Q_{EM} + Q_{vis}$$

where

$$Q_T = \nabla\cdot k_T\nabla T \qquad (1.17a)$$

$$Q_{vis} = \tau_{ij}\,\frac{\partial v_i}{\partial x_j} \qquad (1.17b)$$

$$Q_{EM} = \mathbf{J}'\cdot\mathbf{E}' = \frac{J^2}{\sigma} = \frac{|\nabla\times\mathbf{H}|^2}{\sigma} = \frac{\eta}{\mu_o}|\nabla\times\mathbf{B}|^2 \qquad (1.17c)$$

Therefore the internal energy equation has the final form

$$\frac{\partial}{\partial t}\rho e + \nabla\cdot\left[\mathbf{v}\,(\rho e + P)\right] = \mathbf{v}\cdot\nabla P + \nabla\cdot k_T\nabla T + \tau_{ij}\,\frac{\partial v_i}{\partial x_j} + \frac{\eta}{\mu_o}|\nabla\times\mathbf{B}|^2 \qquad (1.18)$$

The Q_{rad} term has been dropped since radiative transport is not being considered.

An additional equation, which considers the evolution of kinetic energy, can be obtained by taking the scalar product of the momentum equation and the the velocity. This equation, although unnecessary to describe the fluid, is useful in verifying the conservation of energy since it may be added to the internal energy equation to obtain the total (mechanical) energy equation. The scalar product is easiest to evaluate in summation notation.

$$v_i\,\frac{\partial}{\partial t}\rho v_i + v_i\,\frac{\partial}{\partial x_j}\left(\rho v_i v_j + P\delta_{ij} - \tau_{ij}\right) = v_i f_i$$

This may be simplified by using

$$\frac{\partial}{\partial t}\,\frac{1}{2}\rho v^2 = v_i\,\frac{\partial}{\partial t}\rho v_i + \frac{1}{2}v^2\,\frac{\partial \rho}{\partial t}$$

$$\frac{\partial}{\partial x_j}\,\frac{1}{2}\rho v^2 v_j = v_i\,\frac{\partial}{\partial x_j}\rho v_i v_j + \frac{1}{2}v^2\,\frac{\partial}{\partial x_j}\rho v_j$$

to yield

$$\frac{\partial}{\partial t}\frac{1}{2}\rho v^2-\frac{1}{2}v^2\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x_j}\frac{1}{2}\rho v^2 v_j-\frac{1}{2}v^2\frac{\partial}{\partial x_j}\rho v_j+v_i\frac{\partial P}{\partial x_i}-v_i\frac{\partial}{\partial x_i}\tau_{ij}=v_i f_i$$

and

$$\frac{\partial}{\partial t}\frac{1}{2}\rho v^2-\frac{1}{2}v^2\left(\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x_j}\rho v_j\right)+\frac{\partial}{\partial x_j}\frac{1}{2}\rho v^2 v_j+v_i\frac{\partial P}{\partial x_i}-v_i\frac{\partial}{\partial x_i}\tau_{ij}=v_i f_i$$

The term in the parentheses vanishes from the continuity equation. Thus

$$\frac{\partial}{\partial t}\frac{1}{2}\rho v^2+\frac{\partial}{\partial x_j}\frac{1}{2}\rho v^2 v_j+v_j\frac{\partial P}{\partial x_j}-v_i\frac{\partial}{\partial x_j}\tau_{ij}=v_i f_i$$

$$\frac{\partial}{\partial t}\frac{1}{2}\rho v^2+\frac{\partial}{\partial x_j}\frac{1}{2}\rho v^2 v_j+v_j\frac{\partial P}{\partial x_j}-\frac{\partial}{\partial x_j}v_i\tau_{ij}+\tau_{ij}\frac{\partial v_i}{\partial x_j}=v_i f_i$$

$$\frac{\partial}{\partial t}\frac{1}{2}\rho v^2+\frac{\partial}{\partial x_j}\left(\frac{1}{2}\rho v^2 v_j-v_i\tau_{ij}\right)=v_i f_i-\tau_{ij}\frac{\partial v_i}{\partial x_j}-v_j\frac{\partial P}{\partial x_j} \qquad (1.19)$$

Adding the internal energy equation (1.18) and the kinetic energy equation (1.19) yields an equation for the total (mechanical energy),

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho v^2+\rho e\right)+\frac{\partial}{\partial x_j}\left[\left(\frac{1}{2}\rho v^2+\rho e+P\right)v_j-v_i\tau_{ij}\right]=v_i f_i+Q_T+Q_{EM} \qquad (1.20a)$$

or in vector notation

$$\frac{\partial}{\partial t}\left(\frac{1}{2}\rho v^2+\rho e\right)+\nabla\cdot\left[\mathbf{v}\left(\frac{1}{2}\rho v^2+\rho e+P\right)-\mathbf{v}\cdot\underline{\tau}-k_T\nabla T\right]=\mathbf{v}\cdot\mathbf{f}+\frac{\eta}{\mu_o}|\nabla\times\mathbf{B}|^2 \qquad (1.20b)$$

Notice that the viscous heating no longer appears on the right hand side. Viscosity only changes kinetic energy into thermal energy so does not appear as a source in the total energy equation. Similarly, the resistive heating term would not appear as a source if the "total" energy included the magnetic energy density, $B^2/2\mu_o$.

These fluid equations are not complete since there is one more variable than equation. The equations are closed using an appropriate equation of state. For the completely ionized astrophysical plasmas being considered, the ideal gas law may be used. For the variables chosen, the law has the form

$$P=(\gamma-1)\rho e \qquad (1.21)$$

where γ is now the ratio of specific heats. Use of this equation of state assumes that all the internal energy is thermal. For other MHD problems, as well as for astrophysical problems considering ionization, a more complicated equation may be required.

1.3 Equations in Conservation Form

The eight equations which describe the time evolution of the system may be simplified, both conceptually and computationally, by transforming them into the conservation form. Each equation then assumes the form

$$\frac{\partial R}{\partial t} + \nabla \cdot \mathbf{F} = S \qquad (1.22)$$

where R is the variable (such as ρ or ρe), **F** is the flux, and S is the source term.

The fluid equations assume this form if we make the following identifications:

1. For the continuity equation (1.8),

   $R = \rho$ $\qquad$ $\mathbf{F} = \rho \mathbf{v}$ $\qquad$ $S = 0$

   $F_i = \rho v_i$

2. For the momentum equations (1.10), consider the x component equation. The other components have the analogous form.

   $R = \rho v_x$ $\qquad$ $F_i = \rho V_x V_i + P\delta_{xi} - \tau_{xi}$ $\qquad$ $S = f_x$

3. For the internal energy equation (1.17)

   $R = \rho e$ $\qquad$ $\mathbf{F} = (\rho e + P)\mathbf{v}$ $\qquad$ $S = \mathbf{v} \cdot \nabla P + Q$

   $F_i = (\rho e + P) v_i$ $\qquad$ $S = v_i + \frac{\partial P}{\partial x_i} + Q$

   The magnetic equations (1.7) also assume the conservation form if we make the following identifications:

4. Once again consider only the x component.

   $R = B_x$ $\qquad$ $F_i = B_x v_i - B_i v_x + \eta\left(\frac{\partial B_x}{\partial x_i} - \frac{\partial B_i}{\partial x}\right)$ $\qquad$ $S = 0$

These eight conservation equations completely specify the time evolution of the magnetohydrodynamic fluid. They are written out in full, with the addition of stratification, in appendix B.

Notice that these equations may be written as a single matrix equation with the form

$$\frac{\partial R}{\partial t}^{\alpha} + \nabla \cdot \mathbf{F}_{\alpha} = S_{\alpha}$$

The program treats the equations this way with α=1 for ρ; α=2,3,4 for ρv_x, ρv_y, ρv_z; α=5 for ρe; and α=6,7,8 for B_x, B_y, B_z.

1.4 Addition of Stratification

In the presence of an uniform external gravitational field, the compressible fluid will exhibit stratification. The equations, as derived to this point, have an equilibrium solution giving the resulting stratification for any given temperature profile. I have chosen to eliminate this explicit dependence in the equations, however, by a change of variables. I believe this is preferable because it emphasizes the variation from equilibrium, minimizes the range in magnitude of the fluid variables, and, perhaps most importantly, reduces the dependence of the numerical equilibrium solution on the numerical boundary conditions. The exact equilibrium solution satisfying the equations under the approximations introduced by the the numerical scheme depends on the choice of numerical method. For problems in which the variation from equilibrium is small compared to the equilibrium values themselves, the errors introduced by requiring the equilibrium solution to satisfy the numerical scheme may exceed the numerical errors in the variation from equilibrium by orders of magnitude. The numerical method at the boundary is often of a lower order than in the interior; so, this effect is exaggerated near the boundary. This boundary effect on the equilibrium solution may be removed by eliminating the explicit stratification in the variables as follows.

First, include the gravity body force as given in equation (1.11) with the restriction the **g** be constant in the negative z direction. In effect, the direction of gravity defines the z direction. For an isothermal atmosphere, the equilibrium atmosphere is given by

$$\rho(z)=\rho(0)\exp(-\frac{z}{H}) \qquad H=\frac{\rho g}{P} \tag{1.23}$$

where H is the scale height. If one lets ρ_0 and P_0 be defined as follows:

$$\rho_0 \equiv \frac{\rho}{\exp(-\frac{z}{H})}$$
$$P_0 \equiv \frac{P}{\exp(-\frac{z}{H})} \tag{1.24}$$

then the fluid equations in conservation form reduce to the form

$$\frac{\partial R_0}{\partial t}+\nabla\cdot F_0 = S_0+\frac{R_0 v_z}{H}$$

where the subscript indicates the fluid variable has the stratification

divided out as in equation (1.24). The gravitational force term cancels out of the momentum equations and the calculations are considerably simplified. A similar technique can be employed for any case in which the equilibrium temperature profile is explicitly known as a function of height. The full equations giving the flux and source terms are summarized in Appendix B.

1.5 Equations In Dimensionless Conservation Form

There are three primary reasons for stating the equations in dimensionless form. First, by appropriate scaling of the dimensionless variables, one may avoid the computational difficulties arising from very small or large numbers. The computer program is less likely to suffer from rounding errors and the user is less likely to be buried in numbers which are difficult to interpret. Second, one may discover the dimensionless physical parameters which determine the problem. Third, the results of a run may be applied to all problems which have the same dimensionless parameters. The effects of scaling will be removed from the equations.

To make the equations dimensionless, I chose three scaling parameters. The first, M, has dimensionality of density and is characteristic of the plasma densities in the problem. The second, L, is a characteristic length in the problem, while the third, T, is a characteristic time scale. If one makes the following substitutions:

$$\rho = \frac{\rho_o}{M}$$

$$x = \frac{x_o}{L} \qquad y = \frac{y_o}{L} \qquad z = \frac{z_o}{L}$$

$$e = e_o \frac{T^2}{L^2}$$

$$v = v_o \frac{T}{L} \qquad c = c_o \frac{T}{L}$$

where the variable with the nought subscript are the variables from the previous section and the unsubscripted variables are dimensionless, the equations are unchanged in form. The value of these three constant were chosen so that the density, the cell size, and the speed of sound each equal 1.0 for the equilibrium solution.

The parameters which specify the problem must also be scaled as follows:

$$g = g_o \frac{T^2}{L}$$

$$B = B_o \frac{T}{L} M^{-\frac{1}{2}}$$

$$\xi = \xi_o \frac{T}{ML^2}$$

$$\eta = \eta_o \frac{T}{L^2}$$

With these substitutions, the eight equations retain their form but the variables are all dimensionless.

$$\frac{\partial B_i}{\partial t} + \frac{\partial}{\partial x_j}\left[B_i v_j - B_j v_i + \eta\left(\frac{\partial B_j}{\partial x_i} - \frac{\partial B_i}{\partial x_j}\right)\right] = 0$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0$$

$$\frac{\partial}{\partial t}\rho \mathbf{v} + \nabla \cdot \left[\rho \mathbf{v}\mathbf{v} + \left(P + \frac{|\mathbf{B}|^2}{2\mu_o}\right)\underline{1} - \underline{\tau}\right] = \rho \mathbf{g} + \frac{(\mathbf{B}\cdot\nabla)\mathbf{B}}{\mu_o}$$

$$\frac{\partial}{\partial t}\rho e + \nabla \cdot \left[\mathbf{v}\left(\rho e + P\right)\right] = \mathbf{v}\cdot\nabla P + \tau_{ij}\frac{\partial v_i}{\partial x_j} + \frac{\eta}{\mu_o}|\nabla \times \mathbf{B}|^2$$

A number of dimensionless parameters appear in the usual theoretical consideration of fluid mechanics and magnetohydrodyanmics. These parameters determine the nature of the problem independent of the physical scaling. A number of these parameters, their physical interpretation, and their dependence on program variables are outlined below (Hughes&Young,1966).

1. The Reynolds number ($=LV_o/\xi$) is a measure of the ratio of inertial to viscous forces. It is specified in the choice of ξ; however, the real viscosity is vanishingly small. Only the numerical viscosity is of importance and its size is determined by computational considerations. In this program, the Reynolds number is always large.

2. The Frounde number ($=V_o^2/gL$) is a measure of the ratio of inertial forces to gravitational forces. It and the scale height are determined by the choice of g.

3. The Prandtl number ($=c_p\xi\rho_o/k_T$) is determined by k_T(see 1.18a) and ξ. It is a measure of the ratio of viscosity to thermal diffusivity. As with the Reynolds number, it is not physically significant for the problems considered but may be used for computational purposes.

4. The plasma β ($=2\mu_o P/B^2$) is determined by the choice of B. This parameter, which is the ratio of gas pressure to magnetic pressure, determines the Alvfén speed and, therefore, the slow mode and fast mode magnetoacoustic wave speeds.

5. The Strouhal number ($=v_o T/L$) may be of importance for fluid flows including oscillations under the action of external driving forces (of period T). It is a measure of the size of the oscillation to the characteristic length. For this program, the Strouhal number is generally in the range of 10-100.

2
Numerical Methods

The eight partial differential equations derived in chapter 1 describe the time evolution of the fluid in the laboratory frame. Several numerical methods must be developed to use these equations computationally. Finite difference methods are used to evaluate the z spatial derivatives while pseudospectral methods are used for the x and y derivatives. The grid points are allowed to move arbitrarily in the z direction, so a coordinate transformation is required. Several different scheme were tried for time integration of the equations. I have chosen to use a fourth order Runge Kutta like scheme which treats the equations as coupled ordinary differential equations. Each of these numerical techniques is developed in the following sections.

2.1 Spatial Derivatives Using Finite Difference Methods

I wish to approximate the z spatial derivative of a function whose value is known at a finite set of discrete points. Assume the value of the function $f_\ell = f(z_\ell)$ is known for all$\{\ell : \ell = 0,1,2,\ldots,NL\}$. To find the finite differrence approximation to the derivative $\partial f/\partial z$, define L to be an independent, continuous variable. Also define $z(L) = z_\ell$ and $f(L) = f_\ell$ for $L = \ell$. For other values of L, I only require that f(L) and z(L) be continuous and sufficiently differentiable. Since f(L)=f(x,y,z,t),

$$\frac{\partial f}{\partial L} = \frac{\partial f}{\partial z}\frac{\partial z}{\partial L} + \frac{\partial f}{\partial x}\frac{\partial x}{\partial L}^{0} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial L}^{0} + \frac{\partial f}{\partial t}\frac{\partial t}{\partial L}^{0}$$

$$\frac{\partial f}{\partial z} = \frac{\frac{\partial f}{\partial L}}{\frac{\partial z}{\partial L}} \qquad (2.1)$$

Therefore the problem reduces to finding the finite difference approximation for $\partial f/\partial L$ and $\partial z/\partial L$. Values of these functions are known for an *evenly spaced* (in L) set of points.

Expand f about the point $L = \ell$ using a Taylor expansion to find an approximation for f at the grid points $L < \ell$,

$$f_{\ell-4} = f_\ell - 4\frac{\partial f}{\partial L} + 8\frac{\partial^2 f}{\partial L^2} - \frac{32}{3}\frac{\partial^3 f}{\partial L^3} + \frac{32}{3}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell-3} = f_\ell - 3\frac{\partial f}{\partial L} + \frac{9}{2}\frac{\partial^2 f}{\partial L^2} - \frac{9}{2}\frac{\partial^3 f}{\partial L^3} + \frac{27}{8}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell-2} = f_\ell - 2\frac{\partial f}{\partial L} + 2\frac{\partial^2 f}{\partial L^2} - \frac{4}{3}\frac{\partial^3 f}{\partial L^3} + \frac{2}{3}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell-1} = f_\ell - 1\frac{\partial f}{\partial L} + \frac{1}{2}\frac{\partial^2 f}{\partial L^2} - \frac{1}{6}\frac{\partial^3 f}{\partial L^3} + \frac{1}{24}\frac{\partial^4 f}{\partial L^4}$$

where all the partial derivatives are evaluated at $L = \ell$. Use has been made of the fact $\Delta L = 1$ between adjacent points. Similarly for points with $L > \ell$.

$$f_{\ell+1} = f_\ell + 1\frac{\partial f}{\partial L} + \frac{1}{2}\frac{\partial^2 f}{\partial L^2} + \frac{1}{6}\frac{\partial^3 f}{\partial L^3} + \frac{1}{24}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell+2} = f_\ell + 2\frac{\partial f}{\partial L} + 2\frac{\partial^2 f}{\partial L^2} + \frac{4}{3}\frac{\partial^3 f}{\partial L^3} + \frac{2}{3}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell+3} = f_\ell + 3\frac{\partial f}{\partial L} + \frac{9}{2}\frac{\partial^2 f}{\partial L^2} + \frac{9}{2}\frac{\partial^3 f}{\partial L^3} + \frac{27}{8}\frac{\partial^4 f}{\partial L^4}$$

$$f_{\ell+4} = f_\ell + 4\frac{\partial f}{\partial L} + 8\frac{\partial^2 f}{\partial L^2} + \frac{32}{3}\frac{\partial^3 f}{\partial L^3} + \frac{32}{3}\frac{\partial^4 f}{\partial L^4}$$

This expansion can be inverted (neglecting terms of order greater than four) to give the required relation between ∂f/∂L and f at adjacent points. This inversion is most easily done using matrix methods. As an example, consider the fourth order centered difference. The earlier expansion becomes, in matrix form,

$$\begin{pmatrix} f_{\ell-2} \\ f_{\ell-1} \\ f_\ell \\ f_{\ell+1} \\ f_{\ell+2} \end{pmatrix} = \begin{pmatrix} 1 & -2 & 2 & -\frac{4}{3} & \frac{2}{3} \\ 1 & -1 & \frac{1}{2} & -\frac{1}{6} & \frac{1}{24} \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & \frac{1}{2} & \frac{1}{6} & \frac{1}{24} \\ 1 & 2 & 2 & \frac{4}{3} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} f_\ell \\ \frac{\partial f}{\partial L} \\ \frac{\partial^2 f}{\partial L^2} \\ \frac{\partial^3 f}{\partial L^3} \\ \frac{\partial^4 f}{\partial L^4} \end{pmatrix}$$

This has the form $\mathbf{f}=\tilde{\mathbf{A}}\cdot\partial\mathbf{f}$. The solution is then $\partial\mathbf{f}=\tilde{\mathbf{A}}^{-1}\cdot\mathbf{f}$. A microcomputer was used to invert the matrix to yield

$$\begin{pmatrix} f_\ell \\ \frac{\partial f}{\partial L} \\ \frac{\partial^2 f}{\partial L^2} \\ \frac{\partial^3 f}{\partial L^3} \\ \frac{\partial^4 f}{\partial L^4} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \\ -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} \\ -\frac{1}{2} & 1 & 0 & -1 & \frac{1}{2} \\ 1 & -4 & 6 & -4 & 1 \end{pmatrix} \begin{pmatrix} f_{\ell-2} \\ f_{\ell-1} \\ f_\ell \\ f_{\ell+1} \\ f_{\ell+2} \end{pmatrix}$$

Considering only ∂f/∂L one finds

$$\frac{\partial f}{\partial L} = (f_{\ell-2} - 8f_{\ell-1} + 8f_{\ell+1} - f_{\ell+2})/12 \qquad (2.2a)$$

Similarly

$$\frac{\partial z}{\partial L} = (z_{\ell-2} - 8z_{\ell-1} + 8z_{\ell+1} - z_{\ell+2})/12 \qquad (2.2b)$$

The same method is used to find the approximation for ∂f/∂L near the boundary; however, the presence of the boundary requires an off center scheme. The resulting fourth order off-center differences are

$$\left(\frac{\partial f}{\partial L}\right)_1 = (-25f_1 + 48f_2 - 36f_3 + 16f_4 - 3f_5)/12 \qquad (2.2c)$$

$$\left(\frac{\partial f}{\partial L}\right)_2 = (-3f_1 - 10f_2 + 18f_3 - 6f_4 + f_5)/12 \qquad (2.2d)$$

$$\left(\frac{\partial f}{\partial L}\right)_{NL-1} = (-f_{NL-4} + 6f_{NL-3} - 18f_{NL-2} + 10f_{NL-1} + 3f_{NL})/12 \qquad (2.2e)$$

$$\left(\frac{\partial f}{\partial L}\right)_{NL} = (3f_{NL-4} - 16f_{NL-3} + 36f_{NL-2} - 48f_{NL-1} + 25f_{NL})/12 \qquad (2.2f)$$

Sometimes a lower order method may be used at the boundary to minimize the boundary effects.

These finite difference approximations are good to fourth order only for continuous, four time differentiable functions. This limitation is sometimes important because several test problems have discontinuous initial conditions.

2.2 Spatial Derivatives Using Spectral Methods

Derivatives in the x and y direction are evaluated using a pseudo-spectral method and a fast Fourier transform (Gottlieb and Orzsag,1977). For simplicity, I will discuss the x direction only. The y direction is completely analogous.

The function, f, is known at a discrete set of evenly spaced points ($\{x_j\}$ j=0,1,...,NX). This function can be approximated be a complex Fourier transform of the form

$$f(x)= \sum_{j=0}^{NJ} \hat{f}_j \, e^{\left(\frac{2\pi i}{\Delta x \, NX}\right) jx} \tag{2.3}$$

If NJ=NX, the Fourier series will give the exact values for $x=x_j$ while giving a smooth approximation to f between the grid points. One may take the derivative of this function to yield

$$\frac{\partial f}{\partial x} = \sum_{j=0}^{NJ} \hat{f}_j \left(\frac{2\pi i}{\Delta x \, NX}\right) j \, e^{\left(\frac{2\pi i}{\Delta x \, NX}\right) jx} \tag{2.4}$$

In my program a fast Fourier transform is used to find the coefficients in equation (2.3) and the inverse transform is then used to reconstruct the derivatives in equation (2.4). Since the Fourier approximation to f is continuous, this method does not work well for some functions. The well known Gibbs phenomena makes this method especially inappropriate for problems in which the function or its first derivative is discontinuous. This becomes important for several of the test problems which have discontinuous initial conditions.

2.3 Coordinate Transformations for Dynamic Grid

The equations developed in chapter one are for a system of coordinates at rest relative to the laboratory. If I wish to evaluate the functions and their derivatives in another reference frame, I will need to develop the appropriate transformations. I wish to use the new system of coordinates associated with the moving, or dynamic, grid. The grid is allowed to move in the z direction to facilitate driving waves from the boundary as well as to allow the points to be concentrated in regions of interest. Although there are occasions when it may be useful to allow the grid to move in the x and y directions as well, the grid may then become twisted, which adds an unacceptable level of complication. This sort of problem is associated with two and three dimensional Lagrangian methods, which attempt to use the co-moving coordinates.

Denote the laboratory coordinates as x_i ($\{x_i : x,y,z,t\}$) and the new frame coordinates as x_i' ($\{x_i' : x',y',z',t'\}$). The coordinate transform, $\tilde{N}$, which may be nonlinear, is assumed to be known

$$\tilde{\mathbf{N}} : \left\{ \begin{array}{l} x=x(x',y',z',t') \\ y=y(x',y',z',t') \\ z=z(x',y',z',t') \\ t=t(x',y',z',t') \end{array} \right\}$$

To transform the derivatives required by the equation, one needs the relation

$$\frac{\partial}{\partial x_i'} = \left(\frac{\partial x_j}{\partial x_i'}\right) \frac{\partial}{\partial x_j}$$

Define $\partial\tilde{N}$ as the matrix which transforms the derivatives in one frame to derivatives in the other. It is then given by

$$\partial\tilde{N}_{ij} = \left(\frac{\partial x_j}{\partial x_i'}\right)$$

therefore

$$\partial\tilde{N} = \begin{pmatrix} \frac{\partial x}{\partial x'} & \frac{\partial y}{\partial x'} & \frac{\partial z}{\partial x'} & \frac{\partial t}{\partial x'} \\ \frac{\partial x}{\partial y'} & \frac{\partial y}{\partial y'} & \frac{\partial z}{\partial y'} & \frac{\partial t}{\partial y'} \\ \frac{\partial x}{\partial z'} & \frac{\partial y}{\partial z'} & \frac{\partial z}{\partial z'} & \frac{\partial t}{\partial z'} \\ \frac{\partial x}{\partial t'} & \frac{\partial y}{\partial t'} & \frac{\partial z}{\partial t'} & \frac{\partial t}{\partial t'} \end{pmatrix}$$

This matrix is nonsingular if the coordinate transform can be inverted to give $x_i'(x_j)$. It is assumed that transforms which are useful here can be inverted. Thus

$$\frac{\partial}{\partial x_i} = (\partial\tilde{N})^{-1}_{ij} \frac{\partial}{\partial x_j}$$

This may be used to calculate the laboratory frame derivatives from any moving frame derivatives.

For this program, x, y, and t remain the same as the laboratory frame while the grid is allowed to move in the z direction. The new coordinates are {x,y,L,t} and the required coordinate transform is

$$\tilde{\mathbf{N}} : \left\{ \begin{array}{l} x = x' \\ y = y' \\ z = z(x', y', L, t') \\ t = t' \end{array} \right\}$$

This yields

$$\partial\tilde{N} = \begin{pmatrix} 1 & 0 & \frac{\partial z}{\partial x'} & 0 \\ 0 & 1 & \frac{\partial z}{\partial y'} & 0 \\ 0 & 0 & \frac{\partial z}{\partial L} & 0 \\ 0 & 0 & \frac{\partial z}{\partial t'} & 1 \end{pmatrix}$$

This matrix is easily inverted to give

$$(\partial\tilde{N})^{-1} = \begin{pmatrix} 1 & 0 & -\frac{\partial z}{\partial x'}/\frac{\partial z}{\partial L} & 0 \\ 0 & 1 & -\frac{\partial z}{\partial y'}/\frac{\partial z}{\partial L} & 0 \\ 0 & 0 & 1/\frac{\partial z}{\partial L} & 0 \\ 0 & 0 & -\frac{\partial z}{\partial t'}/\frac{\partial z}{\partial L} & 1 \end{pmatrix}$$

Therefore, one may evaluate the laboratory frame derivatives required in the equations with

$$\frac{\partial}{\partial x} = \frac{\partial}{\partial x'} - \frac{\partial z}{\partial x'}/\frac{\partial z}{\partial L}\,\frac{\partial}{\partial L} \tag{2.5a}$$

$$\frac{\partial}{\partial y} = \frac{\partial}{\partial y'} - \frac{\partial z}{\partial y'}/\frac{\partial z}{\partial L}\,\frac{\partial}{\partial L} \tag{2.5b}$$

$$\frac{\partial}{\partial z} = 1/\frac{\partial z}{\partial L}\,\frac{\partial}{\partial L} \tag{2.5c}$$

$$\frac{\partial}{\partial t} = \frac{\partial}{\partial t'} - \frac{\partial z}{\partial t'}/\frac{\partial z}{\partial L}\,\frac{\partial}{\partial L} \tag{2.5d}$$

using only derivatives in the moving frame.

2.4 Time Integration of the Equations

The equations from chapter 1 and numerical methods from 2.1-2.3 are used together to calculate the time rate of change of each of the eight gas variables. A numerical scheme is still required for the time integration. A number of explicit techniques were tried, including Euler's, predictor-corrector, and leapfrog methods. The final program uses the fourth order Runge Kutta like method outlined below. It was chosen because it permits larger time steps and, more importantly, gives a measure of the accumulating numerical error resulting from the time integration.

Using the equations of chapter 1 and the earlier derived numerical methods, I have written a subroutine which returns the value of dR/dt using the variables at the current time step so

$$\frac{dR}{dt} \equiv f'(R(t),t)$$

A second subroutine uses these derivatives to find R at t+Δt using the following fourth order Runge-Kutta formula (Abramowitz and Stegun,1964, §9.2)

$$k_1 = \Delta t\ f'(R(t),t)$$
$$k_2 = \Delta t\ f'(R(t)+k_1/2,t+\Delta t/2)$$
$$k_3 = \Delta t\ f'(R(t)+k_2/2,t+\Delta t/2)$$
$$k_4 = \Delta t\ f'(R(t)+k_3,t+\Delta t)$$
$$R(t+\Delta t) = R(t) + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \qquad (2.6)$$

The resulting error is of order $(\Delta t)^5$. By calculating a new value R_1 using one step of $2\Delta t$ and comparing the result with the value, R_2, obtain from two steps of Δt, one gets an indication of the size of the error, ϵ, as follows:

$$R_1 = R(t+2\Delta t) + \phi\,(2\Delta t)^5$$
$$R_2 = R(t+2\Delta t) + 2\phi(\Delta t)^5$$
$$\epsilon \equiv 2\phi(\Delta t)^5 = (R_1 - R_2)/15$$

Where Φ is a numerical constant (assumed independent of Δt) characteristic of the equations. This can be used to choose the largest time step which produces a desired accuracy, ϵ_0, (Press et al, 1986, §15.2). If the current step size results in an unacceptably large error, it can be repeated with a smaller time step chosen as follows:

$$\Delta t_{new} = 0.9 \Delta t_{old} \left(\frac{\varepsilon_0}{\varepsilon}\right)^{1/4} \tag{2.7}$$

The factor of 0.9 has been included as a safety factor to reduce the probability of repeating a step because Δt is slightly too large. The exponent, 1/4, comes from the desire to limit the error accumulated over all steps. If the error is acceptable, Δt can be increased with the following

$$\Delta t_{new} = 0.9 \Delta t_{old} \left(\frac{\varepsilon_0}{\varepsilon}\right)^{1/5} \tag{2.8}$$

This method provides a straightforward way to obtain the maximum Δt consistent with a desired accuracy; moreover, it produces large time steps for the slowly changing parts of problem and concentrates many steps during the rapidly varying portions. The cost is an extra three evaluations of dR for each two Δt. The derivative must be evaluated four times for each Δt. Thus the cost is about 3/8, which is generally made up by the larger time step.

2.5 Numerical Boundary Conditions

The eight equations describe the behavior of the fluid throughout its volume; however, the region treated numerically is restricted. Since the problems to be considered are not strictly outflow, numerical boundary conditions will be required to maintain numerical stability. Periodic boundary conditions are used for the x and y boundaries as is implicit in the use of pseudo-spectral methods. The lower z boundary is used to drive the waves into the fluid and so explicit conditions will be specified for most problems. The upper boundary may be open or reflecting. To produce an open boundary, I have used a modified Sommerfeld radiation condition similar to the method suggested by Orlanski (Orlanski, 1976, pp251-269).

It is assumed that a disturbance near the boundary obeys the wave equation

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial z} = 0 \qquad (2.9)$$

For each variable, f, this equation is used to determine the wave speed. If $c>\Delta z/\Delta t$, the flow is strongly outward and $f^{t+\Delta t}(NL)$ can be extrapolated from $f^{t}(NL-1)$. If $c<0$ then the wave propagation is into the region and the boundary conditions must be given explicitly. When $0<c<\Delta z/\Delta t$, there is a outward moving wave (assumed to be obeying equation 2.9). Any function of the form

$$f(z,t)=f(z-ct)$$

identically satisfies this equation so, the new value of $f^{t+\Delta t}(NL)$ is given by

$$\begin{aligned} f(z_{NL}^{t+\Delta t},t+\Delta t) &= f(z_{NL}^{t+\Delta t}-ct-c\Delta t) \\ &= f(z_{NL}^{t+\Delta t}-c\Delta t,t) \\ &\simeq f(z_{NL-1}^{t},t)+(z_{NL}^{t+\Delta t}-z_{NL-1}^{t}-c\Delta t)\left(\frac{\partial f}{\partial z}\right)_{NL-1}^{t} \end{aligned} \qquad (2.10)$$

The new value of R is calculated and the required value of dR/dt is returned by the subroutine calculating the time derivative. Note that each variable will have its own characteristic wave speed and thus its own extrapolation. In the program ρ, v_x, v_y, v_z, e, B_x, B_y, and B_z are chosen as the { f } rather than ρ, ρv_x, ρv_y, ρv_z, ρe, B_x, B_y, and B_z.

The dynamic grid greatly complicates the implementation of this method. Moreover, the time derivative used in equation (2.10) is evaluated only to second order, so the boundary may dominate in determining the time step. This becomes a problem when

a large amplitude wave (or shock) passes the boundary. If this occurs and is not the region of interest, the boundary may be "stepped" over the shock by suddenly truncating the region to exclude the shock. The number of grid points may be permanently reduced or extra points may be added between existing points by interpolation.

Another approach may be used for an open boundary when a single outgoing wave velocity can be determined. The values of the variables at an interior point are noted and the position of the point is propagated outward at the wave velocity. When this point passes the boundary, its position and its values of the variables replace the boundary values. A new interior point is chosen and the process repeated. This method worked well for acoustic waves, Alfvén waves, and other cases with a single well defined wave velocity.

To produce a reflecting boundary, two algorithms were used. The simplest was to continually set the variables at the boundary to the initial values. A better method was to set the values at the boundary to those of the next point towards the interior and to reverse the normal velocity; that is,

$$\rho_{NL} = \rho_{NL-1}$$
$$(v_x)_{NL} = (v_x)_{NL-1}$$
$$(v_z)_{NL} = -(v_z)_{NL-1}$$
$$e_{NL} = e_{NL-1}$$
$$\mathbf{B}_{NL} = \mathbf{B}_{NL-1}$$

These conditions were used on both boundaries for some problems in which the motion of interest was along the x direction.

Periodic boundary conditions were also occasionally used. In this case, I set

$$f_{NL} = f_4$$
$$f_{NL-1} = f_3$$
$$f_2 = f_{NL-2}$$
$$f_1 = f_{NL-3}$$

for each of the variables. These conditions were most useful for problems which were independent of z, such as the 1D (in x) test problems.

The effectiveness of these conditions is discussed in chapter 3 for several test problems.

2.6 Numerical Viscosity

Explicit numerical schemes for nonlinear problems are frequently subject to an instability arising from a transfer of energy from long wavelength to short wavelength oscillations. As a result, an oscillation with a wavelength equal to the grid spacing eventually grows to swamp the motions of interest. A similar difficulty arises from the presence of shocks. The standard way of handling these problems is to add artificially large viscosity to the equations. I have already included viscosity in the equations, so the these instabilities can be controlled by the proper choice of coefficients in the equation

$$\tau_{ij} = \xi_1\left[\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - \frac{2}{3}\delta_{ij}\frac{\partial v_k}{\partial x_k}\right] + \xi_2\,\delta_{ij}\frac{\partial v_k}{\partial x_k}$$

The second coefficient, ξ_2, is sufficient to control the short wavelength oscillation; however, ξ_1, may be needed in the presence of oblique shocks or transverse waves.

If the oscillations occur in the fourier (x) direction, they may be reduced or eliminated by removing the highest frequency components. This fourier smoothing has much the same effect as artificial viscosity.

As a result of the addition of this artificial viscosity, real shock waves are widened in profile. The solutions in front of and behind the shocks are unchanged but the shock itself is rounded. This is discussed in greater detail in section 3.2.

3
Test Problems

The program was tested on a number of problems for which the analytic solutions were known. Each of the following sections contains one class of test problem and the results of the computer simulations of that problem. The first few problems are one dimensional, the next few are plane-wave, and the last few are two dimensional.

3.1 Acoustic Waves

The first and simplest test problem was to drive low amplitude sound waves from the lower z boundary. First, a single sinusoidal (in time) pulse was started and followed as it moved in the positive z direction. A sample graph of the resulting wave is given figure 3.1. The wave traveled with the expected speed and pulse length. There was some raggedness at the trailing edge of the pulse resulting from difficulty in stopping the driving force at the correct time (which generally fell in the middle of a time step). There was a reflection from the upper boundary with an amplitude of about 5%.

A similar simulation was performed with an acoustic pulse traveling in the x-direction. Since the boundary conditions are periodic in x, the pulse was driven from an interior point by sinusoidally varying the density and pressure, but not the velocity. One pulse traveled in each direction. Numerical viscosity and fourier smoothing were used to minimize the difficulties arising from the spatial discontinuity at the driving point. A sample graph is given in figure 3.2. Once again, the wave traveled with the expected speed and pulse length. The ragged trailing edge is again apparent.

Next, a continuous acoustic wave was driven from the lower boundary. The results, as shown in figure 3.3, were as predicted by theory. After the initial transient reflections passed the upper boundary, the reflections decreased in amplitude to less than 2%. This improvement can be attributed to use of the the Sommerfeld wave boundary conditions. These boundary conditions should work best when the net outflow results from a wave with uniform and constant speed. They did not work as well for the abrupt beginning and ending of the acoustic pulse.

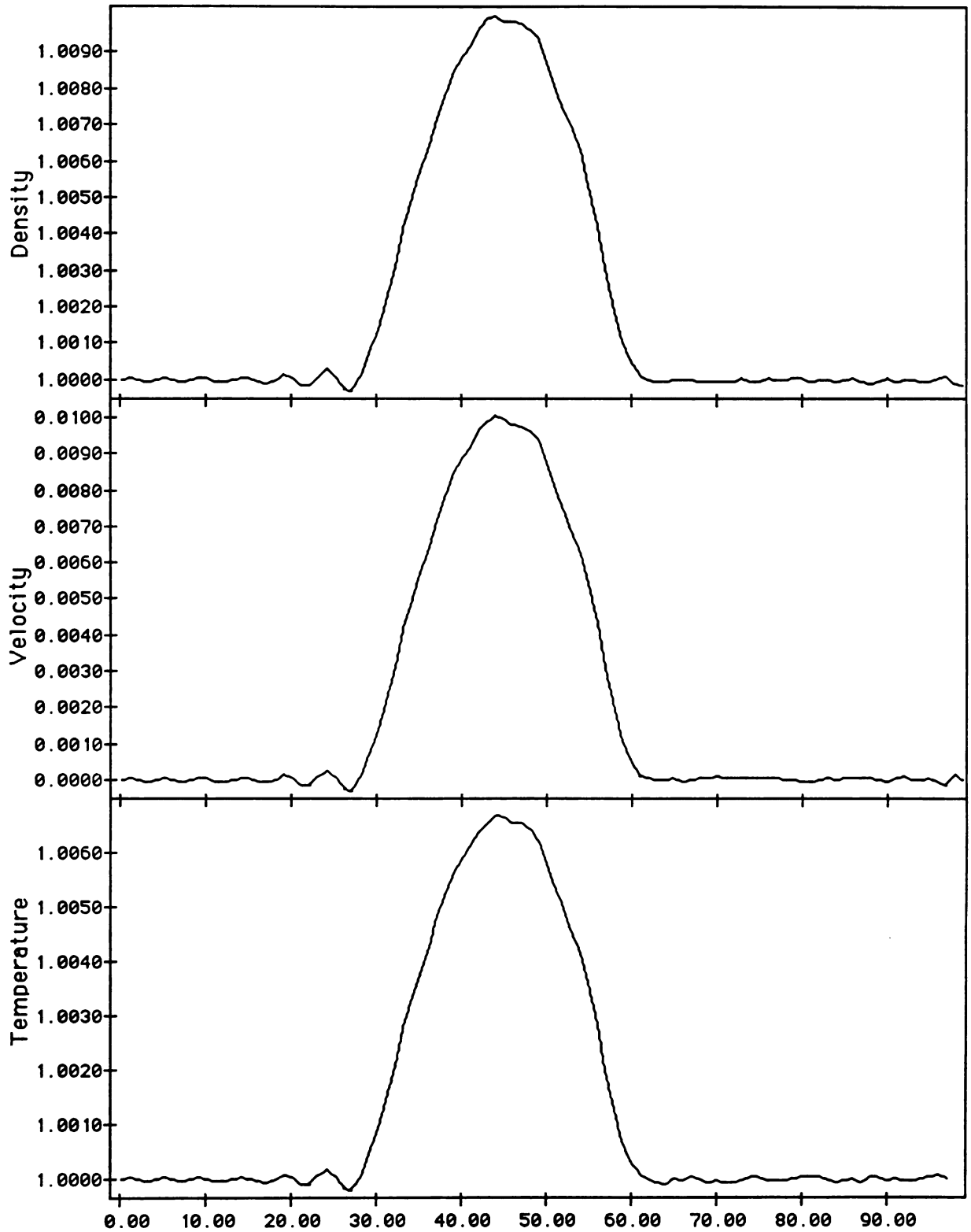


Figure 3.1: An acoustic pulse traveling in the z direction. T=60.00 Amplitude, v_p=0.01.

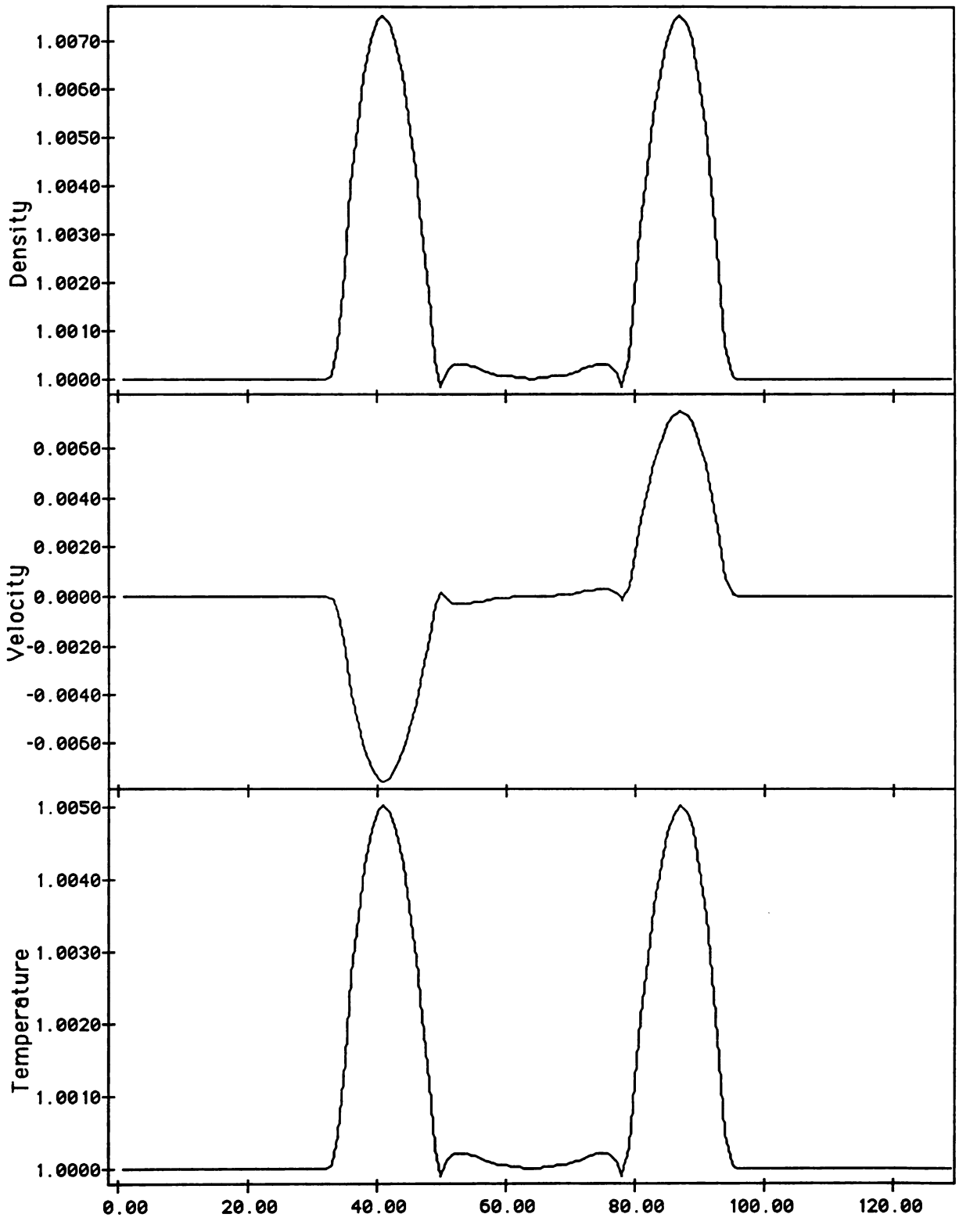


Figure 3.2: Acoustic pulses traveling in the x direction. T=30.00
Amplitude. v_p=0.008.

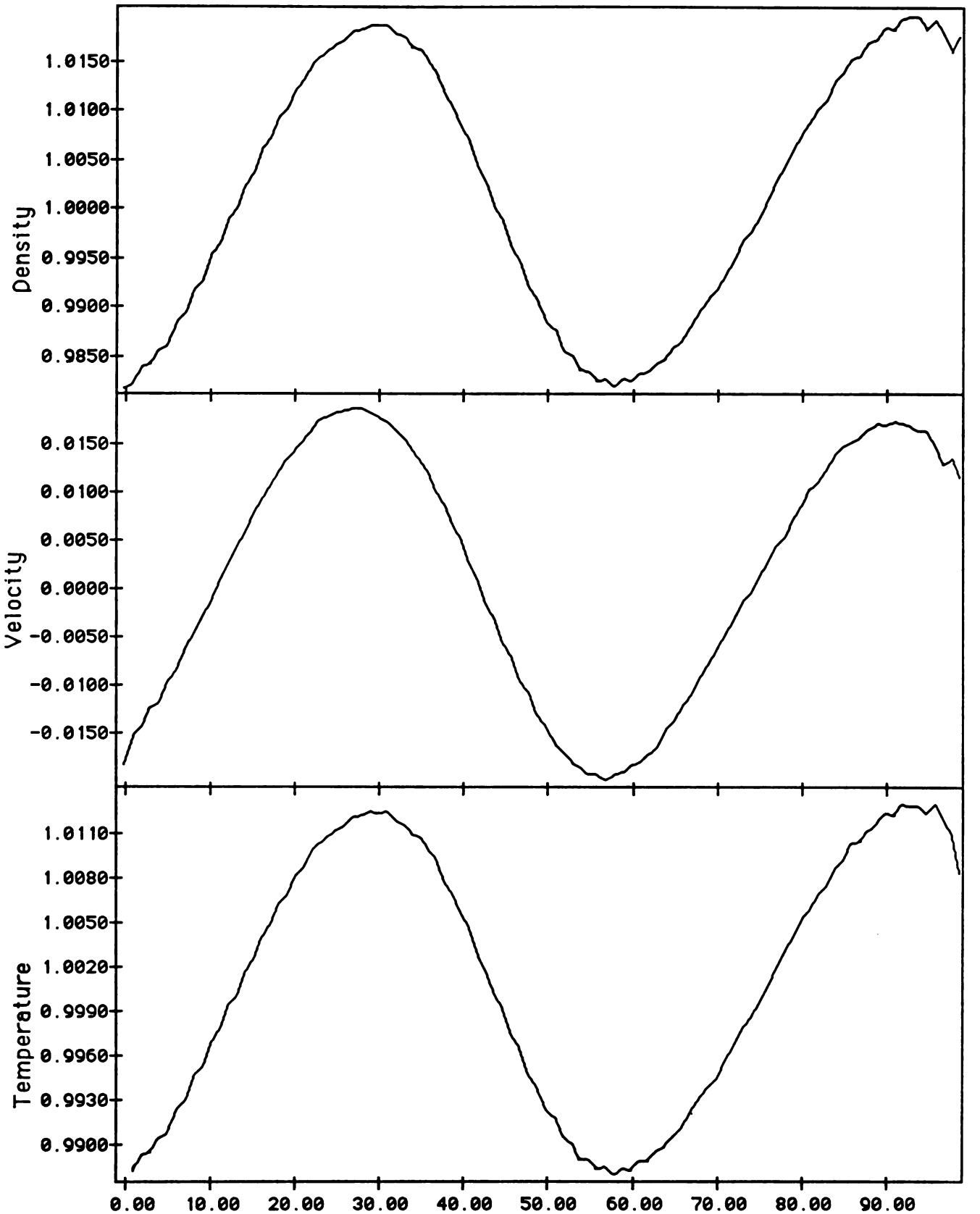


Figure 3.3: Acoustic wave propagating in the positive z direction. T=200.00
Amplitude, v_p=0.02.

3.2 One Dimensional Shock Wave

Another simple problem which is analytically soluble is that of a piston moving into or out of a uniform, non-magnetic fluid. If the piston moves inward, the fluid is divided into two regions separated by a shock as pictured in figure 3.4.

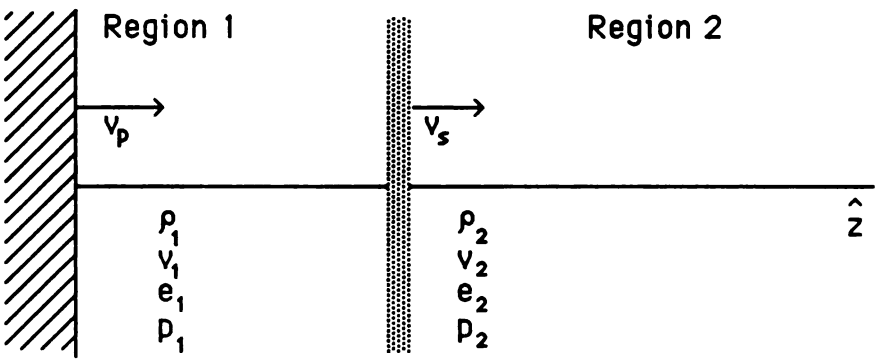


Figure 3.4: Adiabatic shock resulting from an inward moving piston.

To the right of the shock, the fluid parameters are all the same as the initial fluid at rest. To the left of the shock, the velocity of the fluid is that of the piston. The solutions for the remaining post-shock quantities are (see Appendix C)

$$\alpha \equiv \frac{\rho_1}{\rho_2} = \frac{1 + \frac{\gamma+1}{4}\frac{v_p^2}{c_2^2} + \sqrt{\frac{v_p^2}{c_2^2} + \left(\frac{\gamma+1}{4}\frac{v_p^2}{c_2^2}\right)^2}}{1 + \frac{\gamma-1}{2}\frac{v_p^2}{c_2^2}}$$

$$\frac{e_1}{e_2} = \left(1 + \frac{\gamma-1}{2}\frac{v_p^2}{c_1^2}\right)\frac{\alpha+1}{\alpha-1}$$

$$v_s = \frac{\alpha}{\alpha-1}v_p$$

The values which result from my choices of the initial fluid parameters and some values for the piston velocity are given in Table 3.1. When the code was used to simulate this problem using no viscosity, the post-shock instability alluded to in section 2.6 dominated. Sample output for such a case (v_p=0.1) is given in figure 3.5. This instability was eliminated by the addition of viscosity.

v_p	ρ_1	T_1	v_s
0.01	1.01003	1.00667	1.00667
0.05	1.05082	1.03363	1.03389
0.10	1.10321	1.06793	1.06889
0.25	1.26868	1.17591	1.18046
0.50	1.56343	1.37914	1.38743
1.00	2.15139	1.91234	1.86852
$v_p \rightarrow \infty$	4.00000	$3/4 v_p^2/\gamma$	$4/3\ v_p$

Table 3.1: Solution for piston problem, γ=5/3, ρ_2=1.0 , T_2=1.0

Sample output is given for such a case (v_p=0.1, ξ_2=0.15) in figure 3.6. The post-shock oscillations have been eliminated at the cost of widening the shock. For weak shocks, the shock assumes a profile given by a hyperbolic tangent. In this case, the thickness of the shock can be described by a single parameter δ where $\tanh(z/\delta)$ describes the shape of the profile. For weak shocks in an ideal gas, (Landau & Lifshitz, 1959, §92)

$$\delta \simeq \frac{2c}{\gamma^2 - 1} \frac{\xi_2}{\rho_2 e_2 - \rho_1 e_1}$$

The program was run for each of the first five piston speeds in table 3.1 with the viscosity chosen to give a shock thickness of about 4.0. The program worked well in each case. The average value of the gas variables behind the shock and the speed of the shock varied from the theoretical values by less than 1%. The amplitude of the reflection of weak shocks from the open boundary were of approximately the same size as those from the acoustic pulses; however, strong shocks were reflected with an amplitude of 10-15%. For problems in which this presents a problem, the program would have to step over the shock.

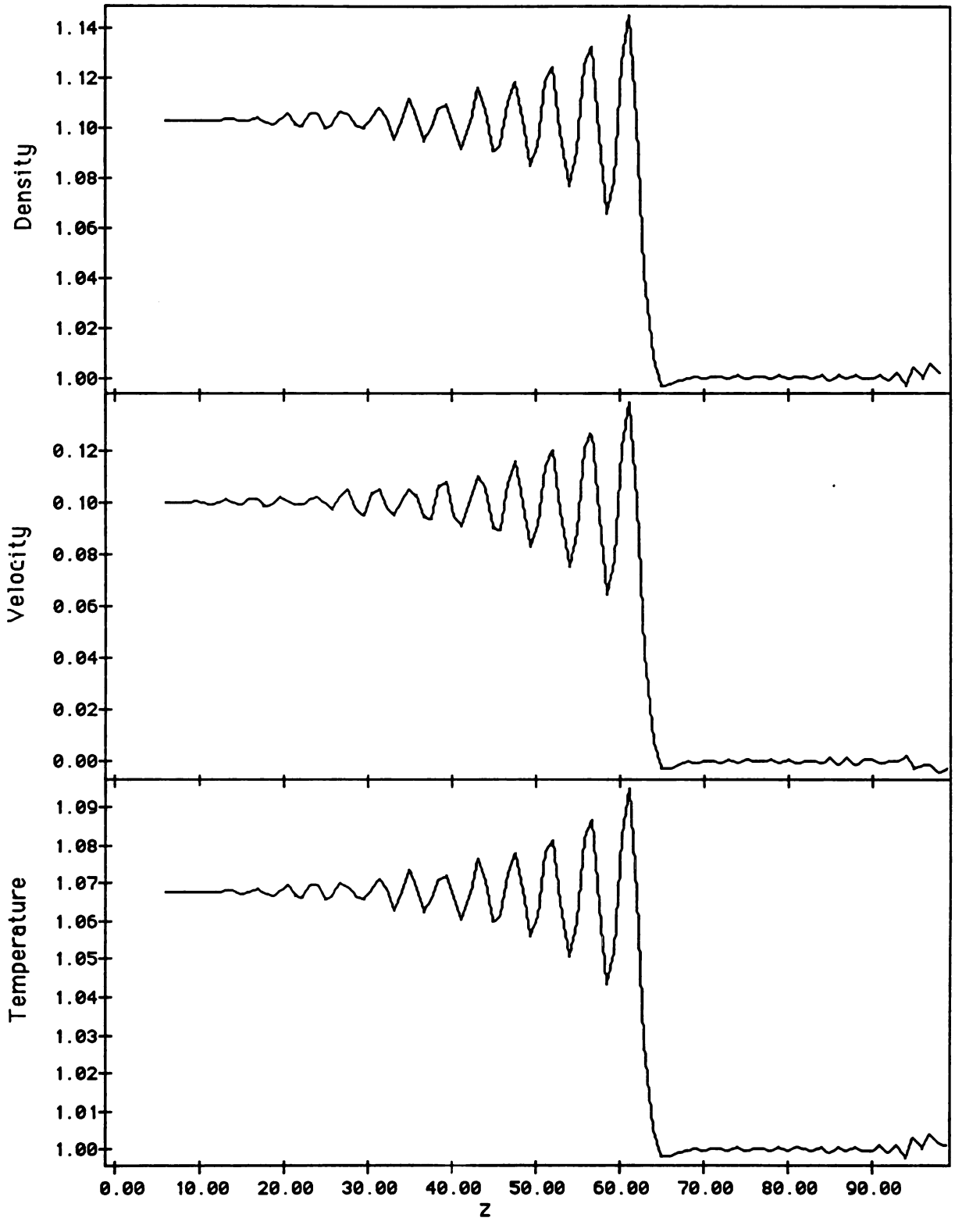


Figure 3.5 : Sample output for piston with v_p=0.1 and no viscosity.
t=60.0

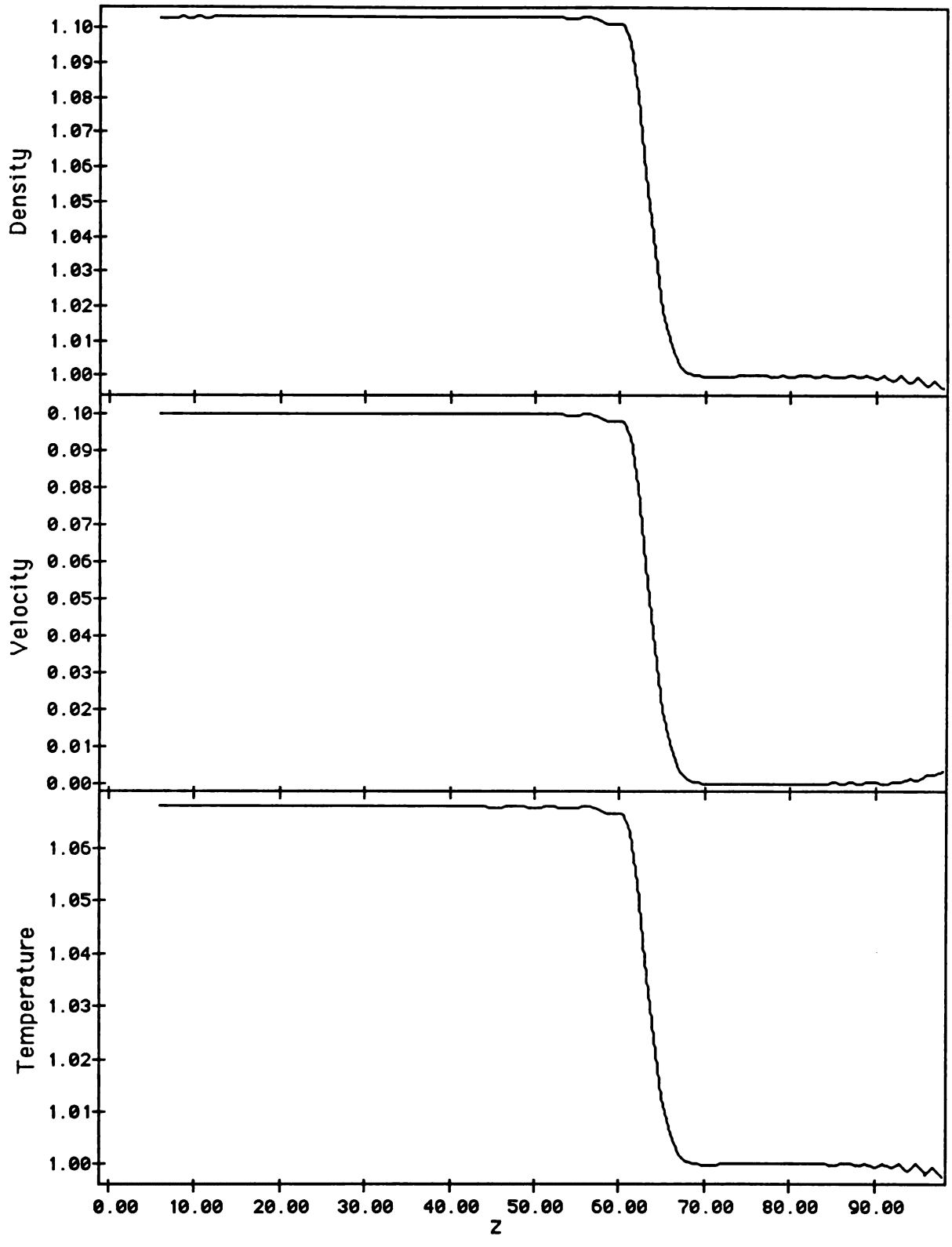


Figure 3.6: Computational results for shock wave in z direction.
v_p=0.1 , ξ_2=0.15 , and t=60.0

A similar problem was run in the spectral direction. Since the shock could not be driven from a boundary, an initial discontinuity was used to create the shock. The numerical domain was initially divided into two regions — one region had $v=v_p$ while the other had $v=-v_p$. Both regions had the same density and temperature. For $t>0$, there is a region in which the gas has come to rest as well as the initial two regions. A shock is driven into each of the initial regions. The solution is the same as the piston problem given in figure 3.4 except that region 1 is at rest rather than region 2. Because of the periodic boundary conditions, there is also an interface at which the gases are moving away from each other. Rarefaction waves are generated at this junction. These are considered in the next section.

The program was run for several of the piston speeds in table 3.1. Numerical viscosity and fourier smoothing were used to minimize the effect of the spatial discontinuity; however, very small time steps were still initially required. These small steps and the requirement for a large number of cells (only one fourth of the domain was useful for the shock problem) slowed the program considerably — each run took about eight hours on the Vax.

The post shock values again were within 1% of the theoretical values; however, the program had some difficulty with modeling the shock itself. This is most evident in the dip just behind the shock in figure 3.7. This was a remnant of the initial discontinuity and decreased with time.

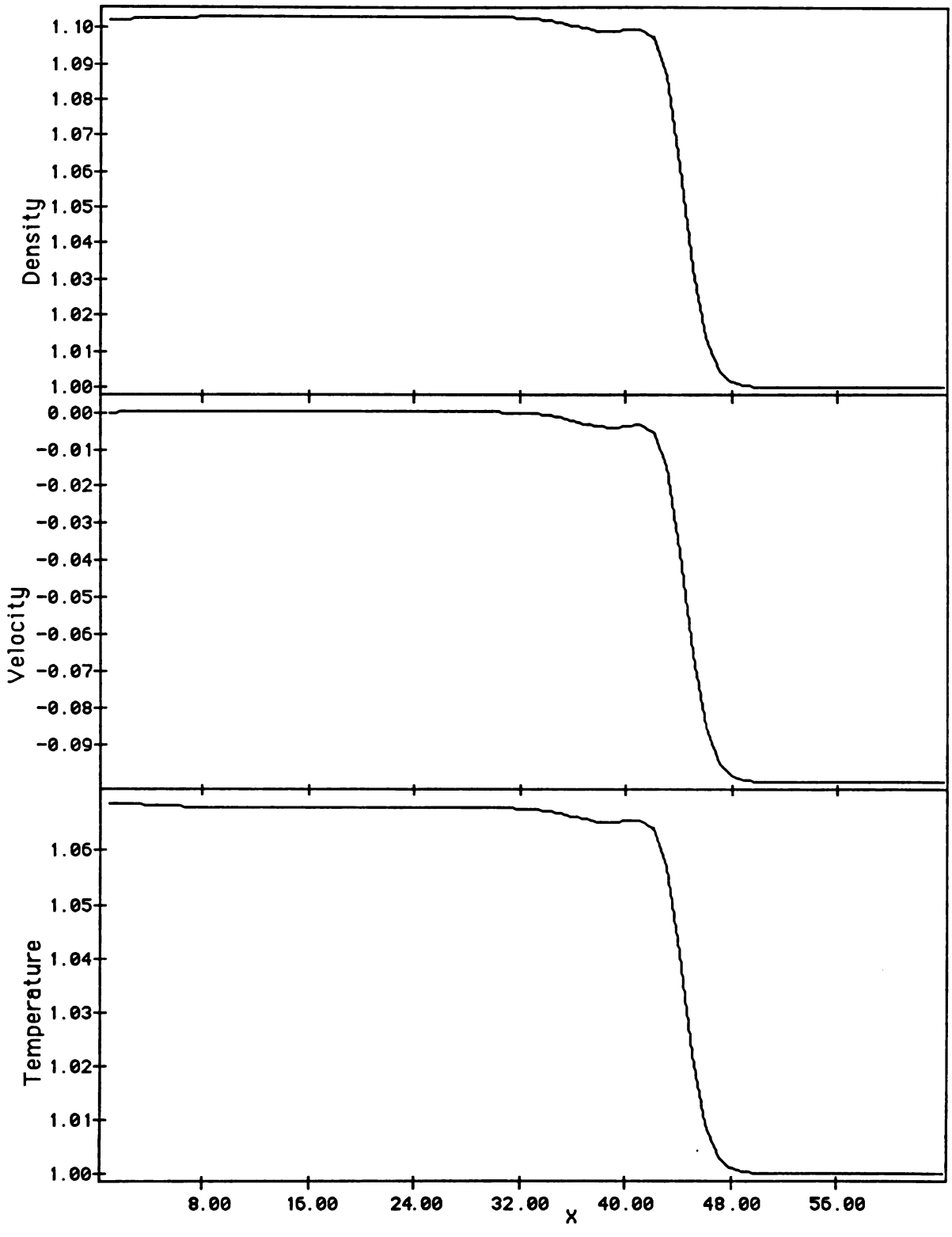


Figure 3.7: Computational results for shock wave in x direction.
v_p=0.1 , ξ_2=0.15 , and t=45.0.

3.3 One Dimensional Rarefaction

If a piston moves out of a uniform fluid, a rarefaction wave travels into fluid. For piston velocities less than the critical value, $2c/(\gamma-1)$, one expects the solution shown in figure 3.8 (Landau &Lifshitz,1959,§92.).

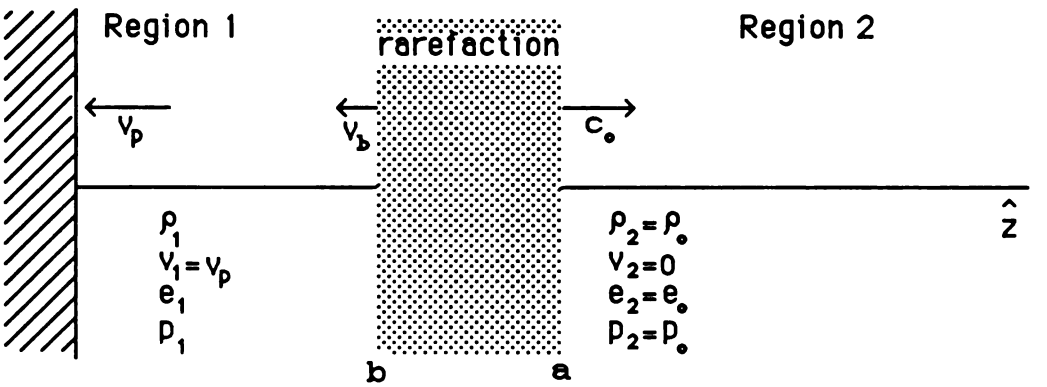


Figure 3.8: Theoretical solution for outward moving piston.

The fluid in region 1 has the same velocity as the piston. In region 2, the original conditions persist. The two regions are separated by a rarefaction wave; the leading edge of which moves with the velocity, $v_a=c_o$ (the initial sound speed). The velocity of the trailing edge, v_b, may be either to the right or to the left and depends on v_p as follows:

$$v_b = c_o - \frac{\gamma+1}{2}|v_p|$$

The value of the fluid variables in region 1 also depend on v_p and are given by

$$\rho_1 = \rho_o\left(1-\frac{\gamma-1}{2}\frac{|v_p|}{c_o}\right)^{\frac{2}{\gamma-1}}$$

$$P_1 = P_o\left(1-\frac{\gamma-1}{2}\frac{|v_p|}{c_o}\right)^{\frac{2\gamma}{\gamma-1}}$$

$$e_1 = \frac{P_1}{(\gamma-1)\rho_1}$$

Table 3.2 contains the values of the fluid variables which result from my choice of density and temperature and several piston speeds. Within the rarefaction wave itself, the speed of the gas is given by

$$|v| = \frac{2}{\gamma+1}\left(c_o - \frac{x}{t}\right)$$

where x is the distance from the front of the rarefaction and t is the time since the piston began moving outward.

The computer program was run for each of the piston speeds in table 3.2. Sample output is shown in figure 3.9. The average values for the gas variables in region 1 agreed with theory to within 1% for $v_p \leq 0.6$. For greater piston speeds, the numerical domain did not contain any of region 1. The rarefaction wave had the expected profile and traveled with the correct speed. There was a small wave packet which preceded the rarefaction wave which I believe was caused by the initial temporal discontinuity. It may be eliminated by increasing the numerical viscosity; however, this also changes the profile of the rarefaction. For purposes of this test, I felt the small oscillations were not significant and the viscosity was not increased.

v_p	ρ	p	e	v_b
0.00000	1.00000	0.60000	0.90000	1.00000
0.10000	0.90330	0.50645	0.84100	0.86667
0.20000	0.81304	0.42495	0.78400	0.73333
0.30000	0.72900	0.35429	0.72900	0.60000
0.40000	0.65096	0.29337	0.67600	0.46667
0.50000	0.57870	0.24113	0.62500	0.33333
0.60000	0.51200	0.19661	0.57600	0.20000
0.80000	0.39437	0.12725	0.48400	-0.06667
1.00000	0.29630	0.07901	0.40000	-0.33333
1.50000	0.12500	0.01875	0.22500	-1.00000
2.00000	0.03704	0.00247	0.10000	-1.66667

Table 3.2: Gas variables after the passage of the rarefaction cause by a piston moving at selected speeds.

The upper boundary performed fairly well in this test. It reflected the waves with an amplitude of about 5%. The reflected rarefaction eventually passed through the original rarefaction and continued to propagate through region 1.

A similar test was run in the spectral direction. Since the rarefaction could not be driven from the boundary, an initial discontinuity in velocity was used as explained in section 3.2. Because the numerical domain also included a shock, the viscosity could not be kept as low. The program was run for the lower piston speeds and the numerical results for region 1 were again within 1% of the theoretical values. Sample output is given in figure 3.10.

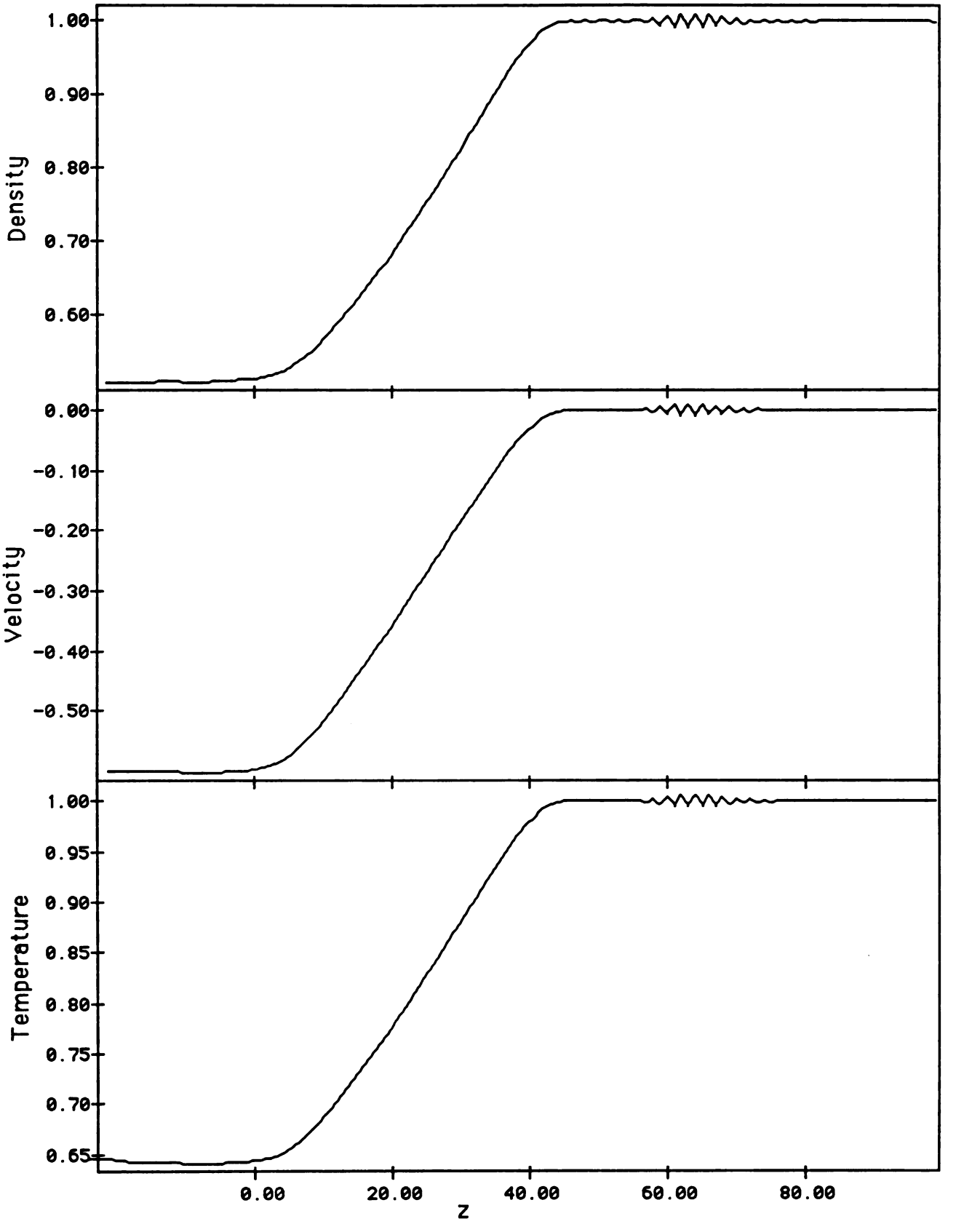


Figure 3.9: Computational results for rarefaction wave in the z direction. v_p= -0.60 , ζ_2=0.05 , and t=40.0.

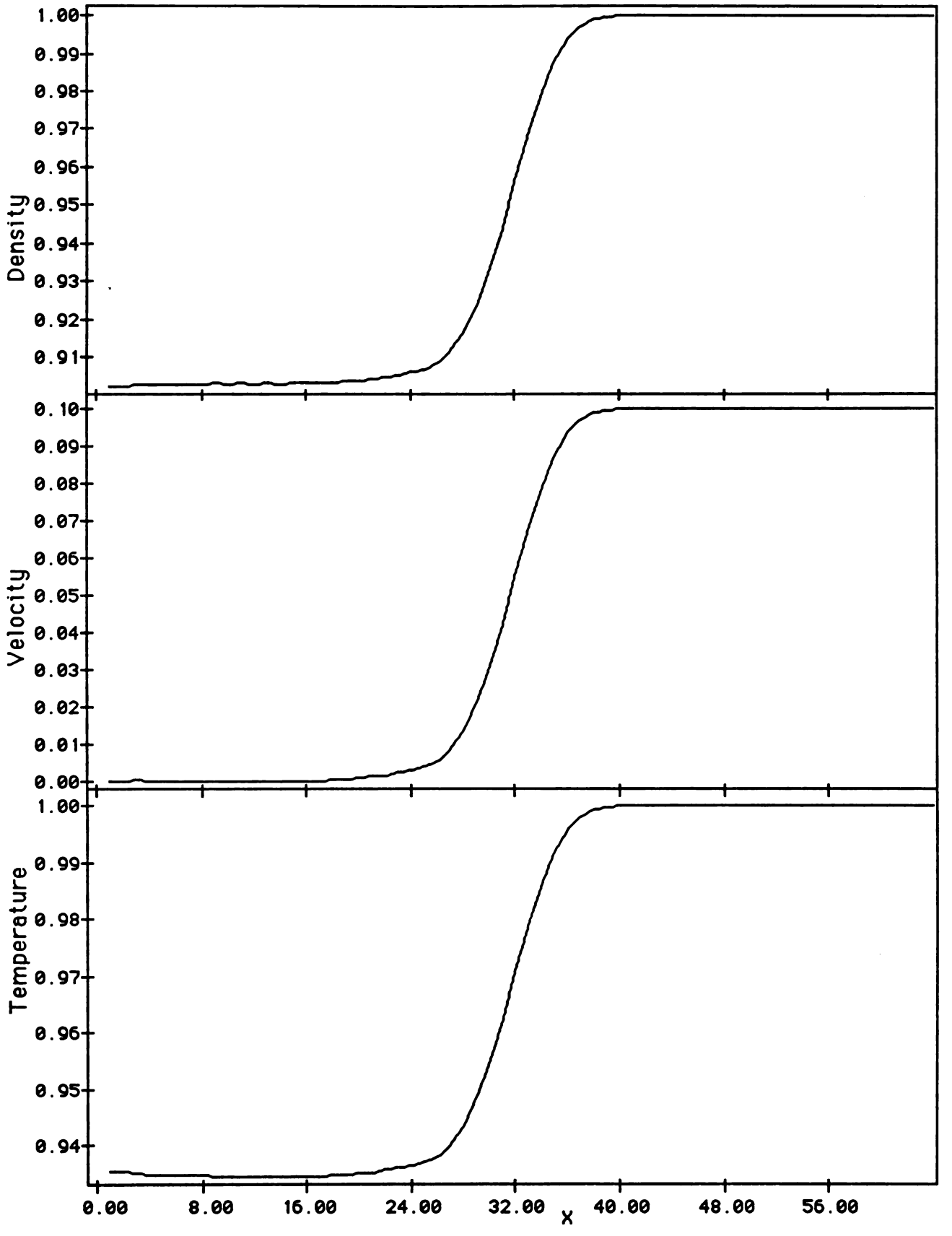


Figure 3.10: Computational results for rarefaction wave in the x direction.
v_p=0.1 , ξ_2=0.15 , and t=30.0.

3.4 Shock tube

Another one dimensional problem which was run in both directions consists of two semi-infinite regions of gas initially separated by a partition. The two regions are in thermal equilibrium but have different pressures. At t=0, the partition is broken and the gas begins to move due to the pressure gradient. For t>0, the gas is divided into four regions as shown in figure 3.11.

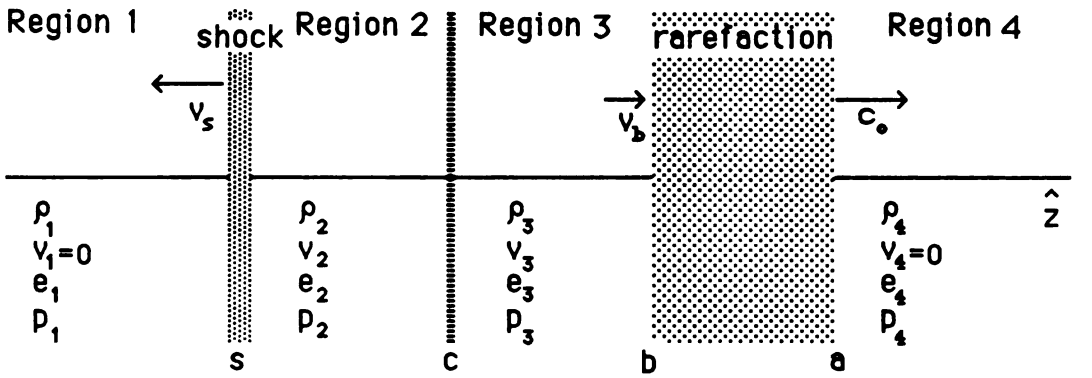


Figure 3.11: Theoretical solution for the shock tube.

Regions 1 and 4 are the remaining parts of the original regions and the initial conditions persist in them. Regions 2 and 3 share the same velocity (toward region1) and pressure but have different densities and temperatures. The regions are separated from each other by a shock wave moving into region 1, a contact discontinuity moving with the gas in regions 2 and 3, and a rarefaction wave moving into region 4. The leading edge of the rarefaction moves with initial sound speed. The speeds of the remaining interfaces, as well as the values of the pressure, density, velocity, and temperature in regions 2 and 3 depend on the initial pressure ratio. The theoretical solution for this problem follows directly from the combination

ratio	V_3	P_3	ρ_2	T_2	V_s	ρ_3	T_3	V_b
1.50000	.12158	1.21972	1.12628	1.08296	1.08433	1.32492	.92059	.83790
2.00000	.20764	1.39727	1.22082	1.14454	1.14796	1.61280	.86636	.72315
4.00000	.41361	1.90515	1.45984	1.30504	1.31306	2.56319	.74327	.44853
5.00000	.47931	2.09391	1.53853	1.36099	1.36935	2.96595	.70599	.36092
6.00000	.53272	2.25753	1.60289	1.40841	1.41634	3.33765	.67638	.28970
8.00000	.61643	2.53295	1.70395	1.48652	1.49210	4.01247	.63127	.17810
10.00000	.68081	2.76108	1.78142	1.54993	1.55205	4.62006	.59763	.09226

Table 3.3: Theoretical values for shock tubes.

of the shock and rarefaction wave solutions. The values which result for my choice of variables and several initial pressure ratios are given in table 3.3.

The program was used to simulate a shock tube with motion in the finite difference direction for each of the initial pressure ratios in table 3.3. Numerical viscosity was used because of the presence of the shock. The numerical results agreed with the theoretical values given in the table to within 1%. The program did have some difficulty modeling the contact discontinuity as can be seen in the temperature oscillations near the discontinuity in figure 3.12, sample output for a shock tube with an initial pressure ratio of 10. I believe this difficulty could be eliminated by using a non-zero thermal conductivity (k_T in eq 1.17a). I did not incorporate this change since contact discontinuities only result from initial discontinuities, which can be avoided when necessary.

The program was also used to simulate a shock tube with motion in the spectral direction for the smaller values of initial pressure ratios (ratio$\leq$5). For the larger values, the time steps became extremely small for the time just after t=0. Numerical viscosity and fourier smoothing were both used to minimize the difficulties arising from all the discontinuities. The numerical results were again within 1% of the theoretical values except for the value of gas velocity in regions 2 and 3. The average value there was still close to the theoretical, but there was a spatial variation in the velocity caused by the contact discontinuity which caused the average value to be uncertain by about 2%. Figure 3.13, which is sample output for an intitial ratio of 2, shows the difficulty in both the temperature and velocity graphs.

The difficulty experienced by both spatial methods in attempting to model the contact discontinuity makes it essential to avoid this type of discontinuity in the initial conditions. This program would need to be improved to give a useful simulation of problems in which contact discontinuities cannot be avoided.

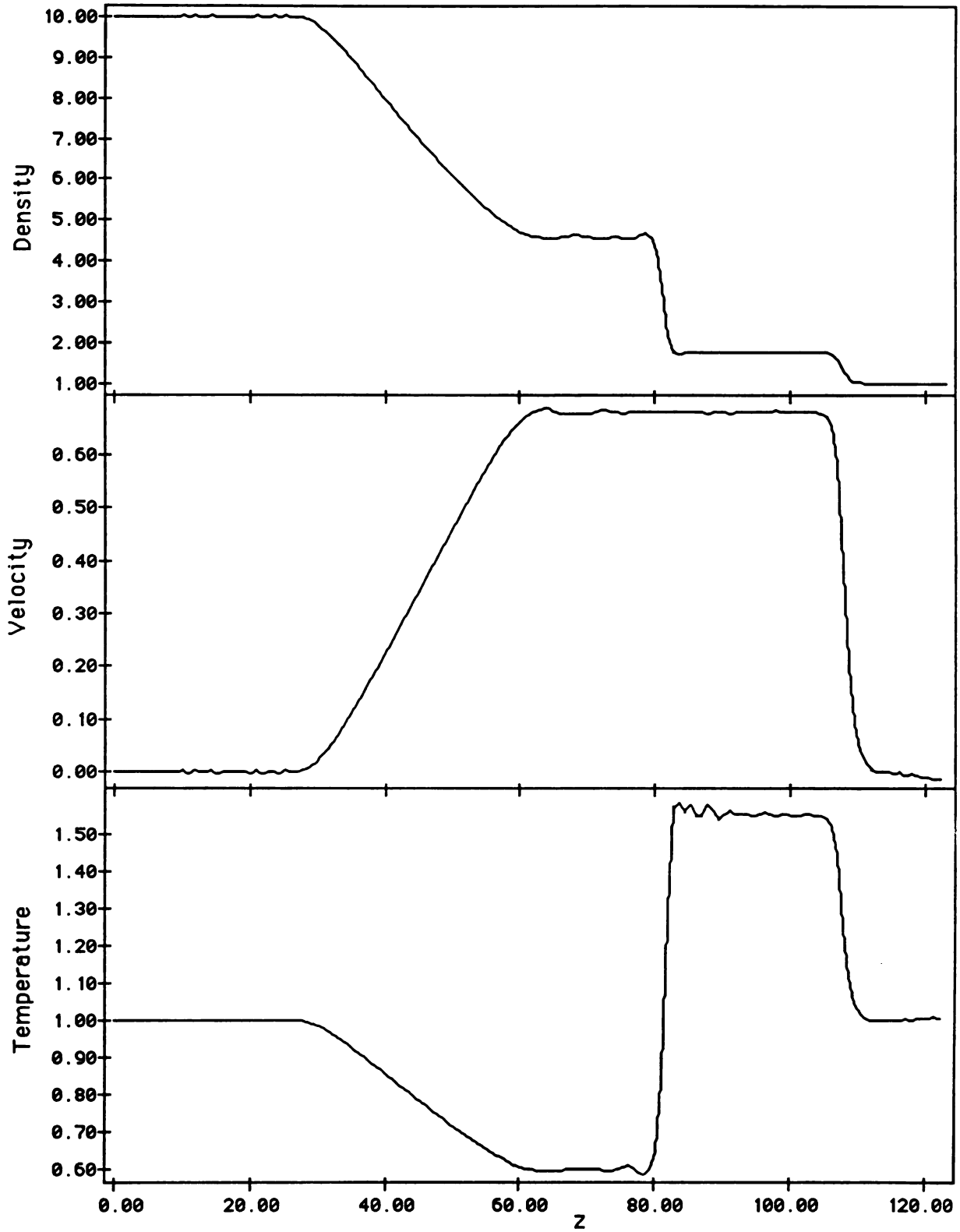


Figure 3.12: Computational results for shock tube in finite difference direction, t=30. Initial pressure ratio=10:1.

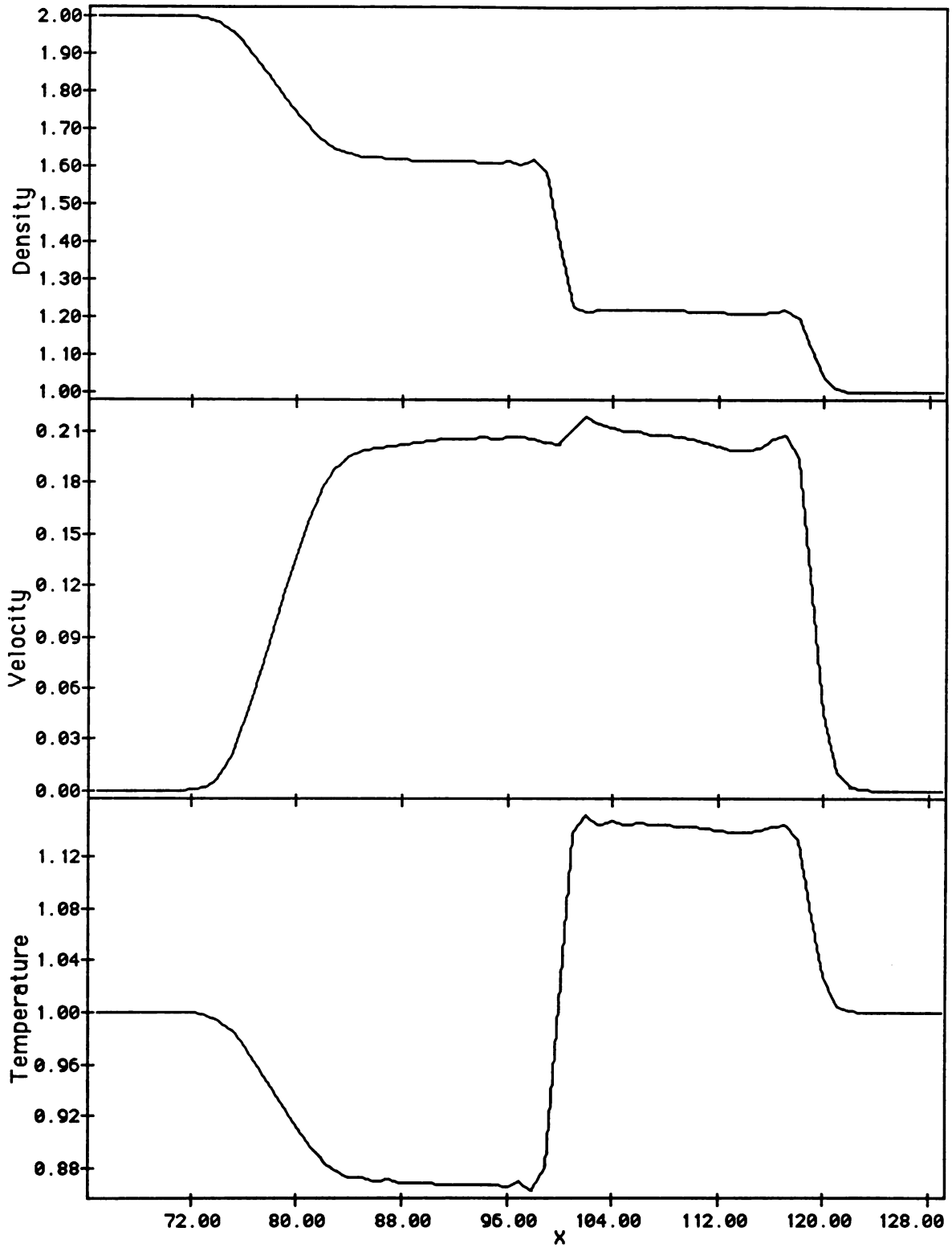


Figure 3.13: Computational results for shock tube in spectral direction, t=20.
Initial pressure ratio = 2:1.

3.5 Alfvén Waves

With the addition of a magnetic field, several new wave modes are possible. The simplest of these are Alfvén waves which propagate along the field lines with a velocity A, where

$$A = \sqrt{\frac{B_o^2}{\rho_o \mu_o}}$$

These waves are transverse oscillations in the velocity and magnetic field – the density, temperature, and longitudinal components of the velocity and magnetic field are undisturbed. In simulating this mode, I used the usual initial conditions and

$$B_z = B_{zo}$$
$$B_x = B_{xo} = 0.0$$

An Alfven wave was driven from the lower boundary by sinusoidally varying B_x and v_x as follows:

$$v_x = v_o \sin(\omega t)$$

and

$$B_x = -B_{zo}\, v_o \sin(\omega t)/A$$

The program was run for several choices of v_o, which effects only the amplitude of the wave, B_{zo},which effects the Alfvén speed, and ω. Sample output showing the spatial variation in B_x and v_x is given in figure 3.14. The remaining variables were constant (±0.05%) throughout the numerical domain and the wave propagated at the theoretical Alfvén speed to within 1%(the accuracy of determining the group velocity).

The program was then run with propagation in the x direction. The waves were generated by varying the transverse velocity. Both fourier smoothing and artificially large viscosity (ξ_2, in this case) were required to overcome the spatial discontinuity at the driving point. I obtained results equal in accuracy to that of the wave propagating in the z direction.

Next the program was used to simulate an Alfvén wave propagating at an angle to the magnetic field. The background magnetic field had both x and z components and the wave propagated in the x or z direction. The y components of the velocity and magnetic field varied while all the other variables remained constant. Sample output is given in figure 3.15. Once again only B_y and v_y varied – the other variables remained constant (±0.05%). The wave velocity was once again within uncertainty of the theoretical value(±1%). In this case, the coupling between the driving point and the wave was not as good as in the case of the propagation along the field lines. The

driving point in figure 3.15 oscillated with an amplitude of 0.020 in v_y. The wave, however, had an amplitude of only 0.017 (or 85% of expected). No other modes were excited and I am not sure of the cause for this decoupling.

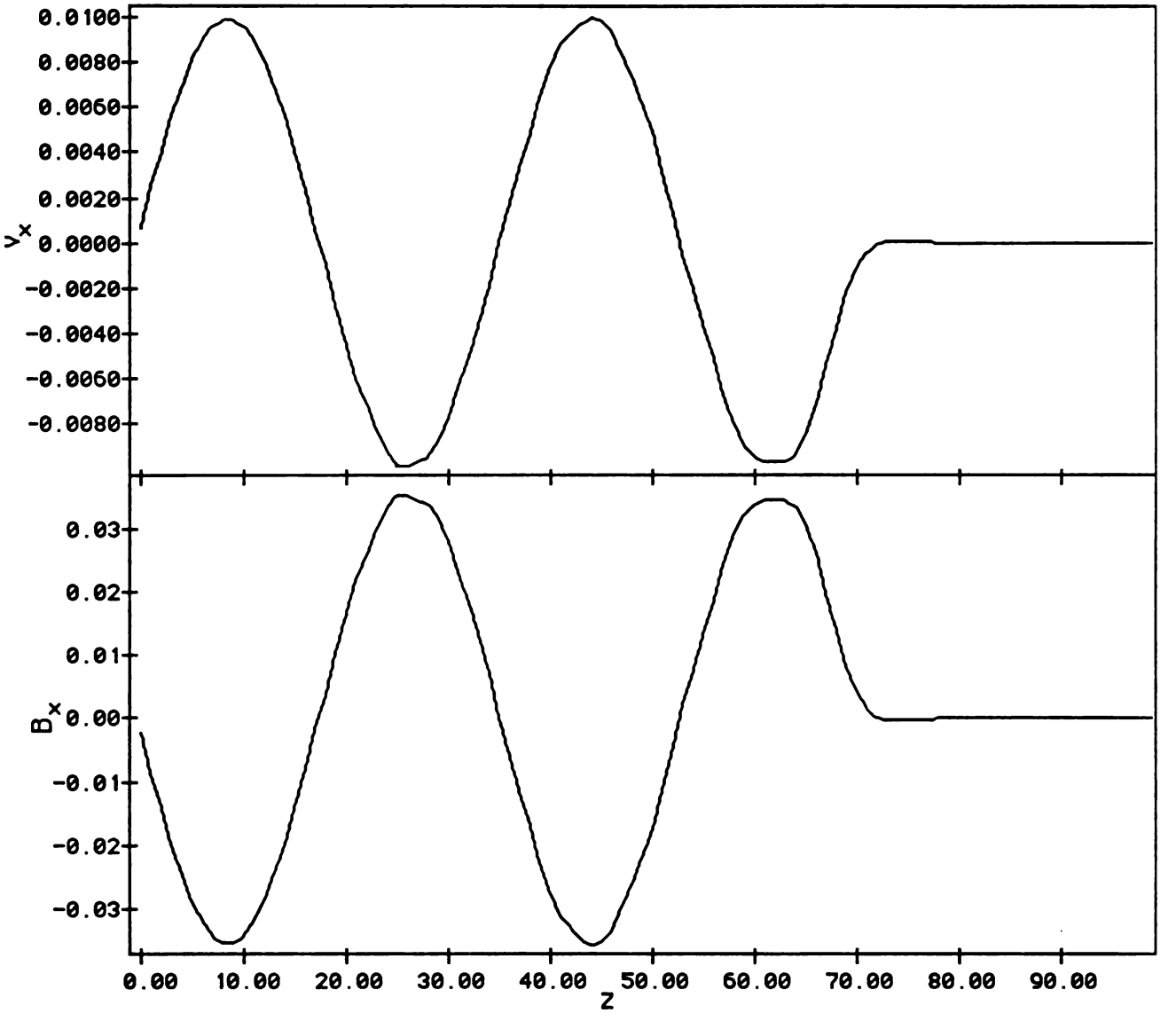


Figure 3.14: Alfvén wave propagating in the z direction. B_z=1.0 , A_z= 0.282 , t=150.0.
β= 15.08 A = 0.282

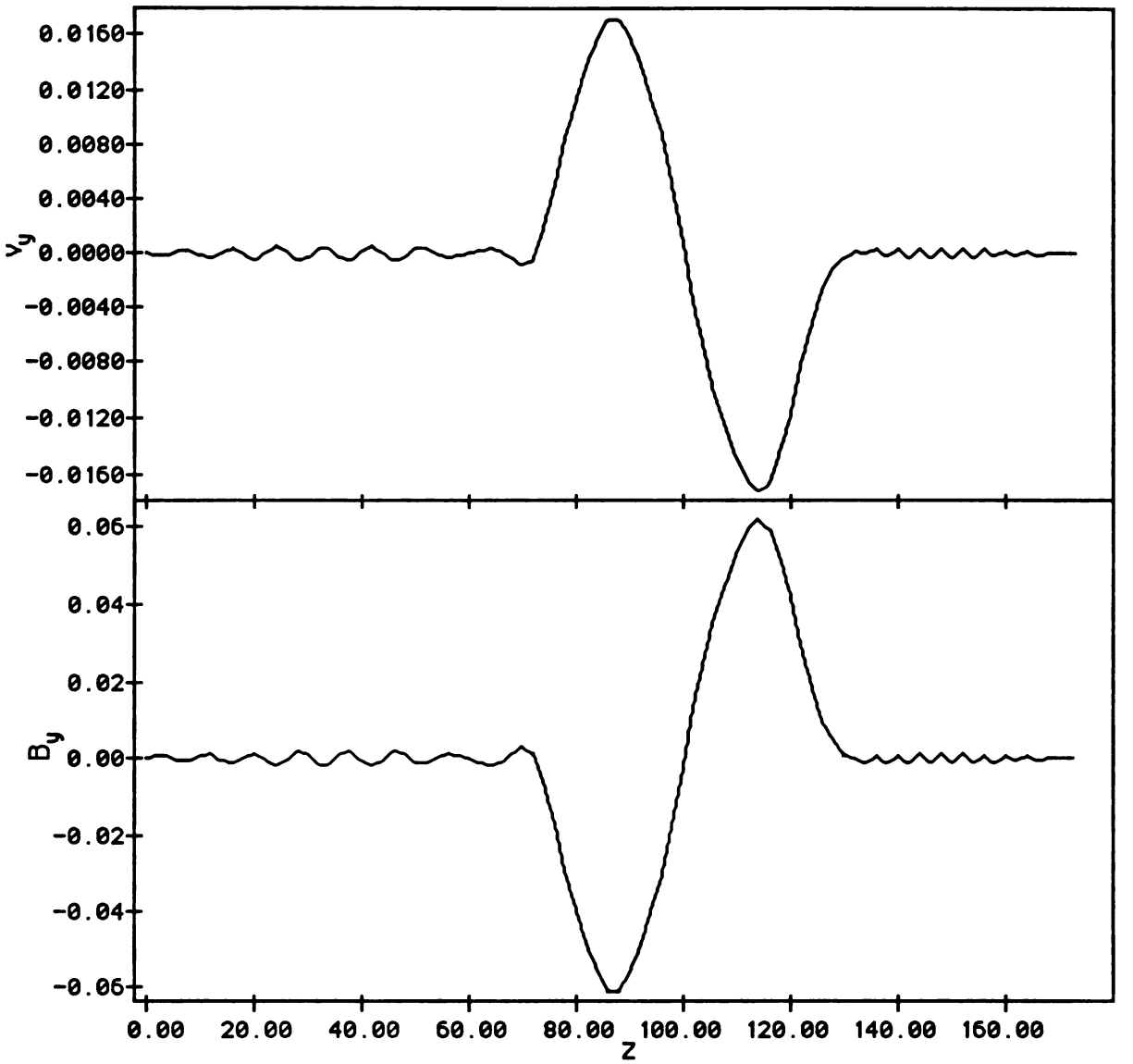


Figure 3.15: Alfvén wave pulse traveling at 45° angle to field lines. Bx=3.0 , Bz=3.0 , β= 1.676 , A= 1.197 , Az= 0.846 , t=150.0

3.6 Magnetoacoustic Waves

The introduction of a magnetic field causes the acoustic mode of oscillation to split into two modes. These two modes, called the fast mode and slow mode, consist of a coupled oscillation in all the variables except those tangential components which vary in the Alfvén mode of oscillation. The phase velocities of these two modes are given by (Hughes & Young,1966)

$$a_{slow} = \pm \left[\frac{A_x^2 + A_z^2 + c_o^2}{2} - \sqrt{\left(\frac{A_x^2 + A_z^2 + c_o^2}{2} \right)^2 - c_o^2 A_z^2} \right]^{\frac{1}{2}}$$

$$a_{fast} = \pm \left[\frac{A_x^2 + A_z^2 + c_o^2}{2} + \sqrt{\left(\frac{A_x^2 + A_z^2 + c_o^2}{2} \right)^2 - c_o^2 A_z^2} \right]^{\frac{1}{2}}$$

where c_o is the sonic speed, A_z is the Alfvén speed in the direction of propagation, and A_x is the Alfvén speed in the direction perpendicular to the direction of propagation. The coordinate axes have been chosen such that B_y=0.0 (without loss of generality).

For propagation along the magnetic field (A_x=0.0), there is only one mode of oscillation, in which $a=c_o$. This case was easily simulated by the program. The waves were driven exactly as they were for the acoustic wave in section 3.1.

For the general case, I tried several different ways of driving the waves. When the transverse magnetic field (B_x) and velocity were varied and the remaining variables held constant at the driving point, both modes were generated simultaneously. Figure 3.16 shows sample output for such a case. The relative strength of the modes depended on the relative amplitude of the oscillation in velocity and magnetic field. A considerable amount of "noise" was also generated which was reduced by using non-zero viscosity and magnetic diffusion.

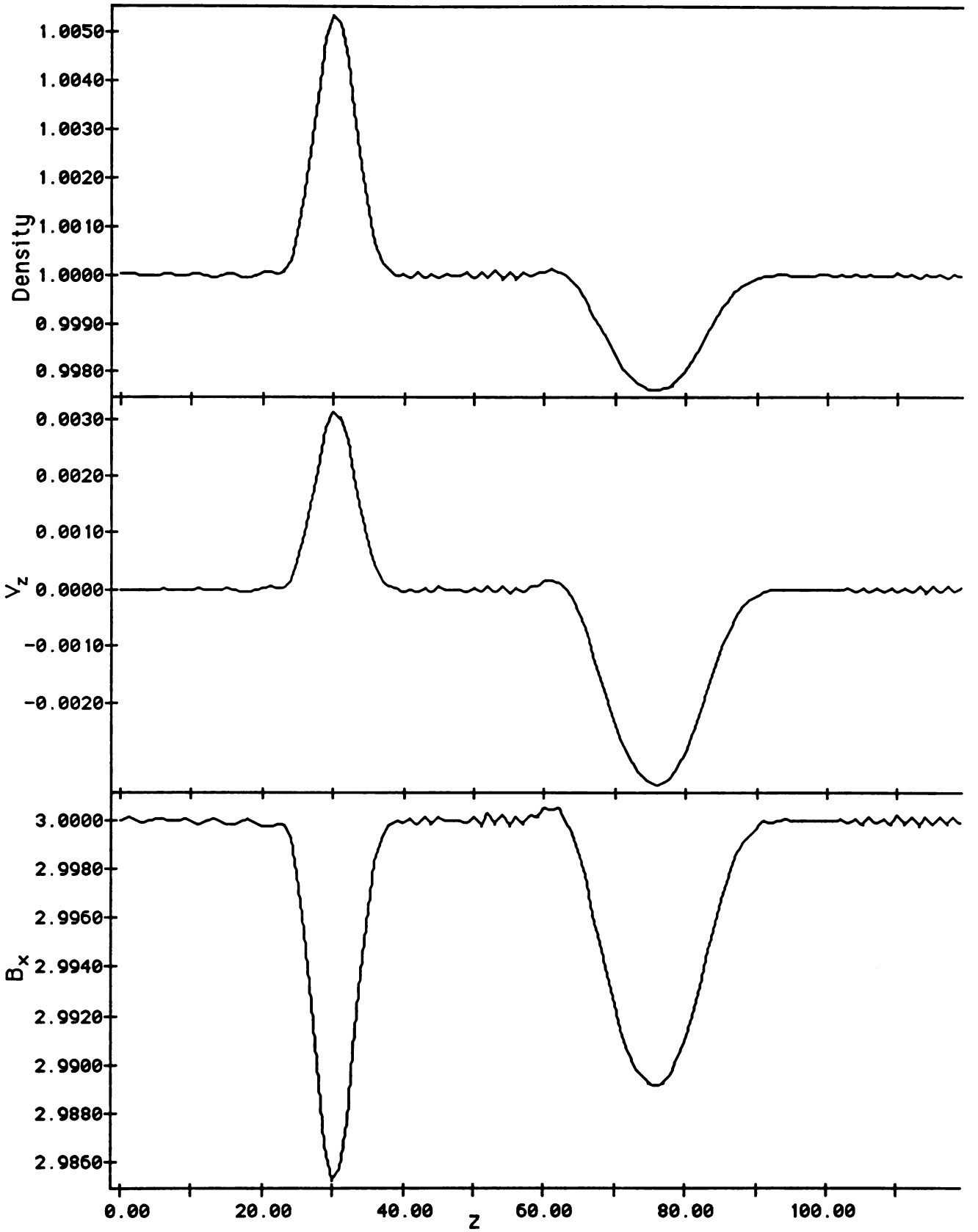


Figure 3.16: Simultaneously generated fast and slow mode magnetoacoustic wave pulses. t=60.0 , η=.01, ξ_1=.05 .

Next I solved the linearized equations to find the relations in the variation of ρ, v_x, v_y, e, and B_x to get

$$v_z = w \sin(\omega t)$$

$$\rho = \rho_o \left(1 + \frac{w}{a} \sin(\omega t)\right)$$

$$v_x = \frac{-A_x A_z}{a^2 - A_z^2} w \sin(\omega t)$$

$$e = e_o \left[1 + (\gamma - 1) \frac{w}{a} \sin(\omega t)\right]$$

$$B_x = B_{xo} \left(1 + \frac{a}{a^2 - A_z^2} w \sin(\omega t)\right)$$

where a is either a_{fast} or a_{slow}. When these relations were used to drive waves from the lower z boundary, only the chosen wave was generated. The program was used to simulate fast and slow mode magnetoacoustic waves propagating at a variety of angles to the magnetic field (by vary B_x and B_z). Figures 3.17 and 3.18 show some typical output (propagation at 45° to the field). A numerical instability developed in the propagation of the slow mode wave which was eliminated by using a non-zero magnetic diffusivity, η. The value used (η=0.01) still corresponded to a very low resistivity.

For propagation in the spectral direction, I used both of the methods outlined above. When just the tangential velocity and field were varied, the results were much the same as in the finite difference direction. There were fewer difficulties with the "noise", probably resulting from the fourier smoothing; therefore, I did not need to use a non-zero magnetic diffusivity.

When the driving point was oscillated using the theoretical relations above, both modes were generated. Since the relations could be correct for only one of the two directions in which the waves propagated, the waves propagating in the "wrong" directions were about equal in amplitude. In the forward direction, the chosen mode dominated. Sample output is shown in figure 3.19, which was driven as a fast wave. Notice that the slow mode waves appear symmetrically. This suggests to me that it may be possible to drive only the fast mode if the correct driving mode can be determined.

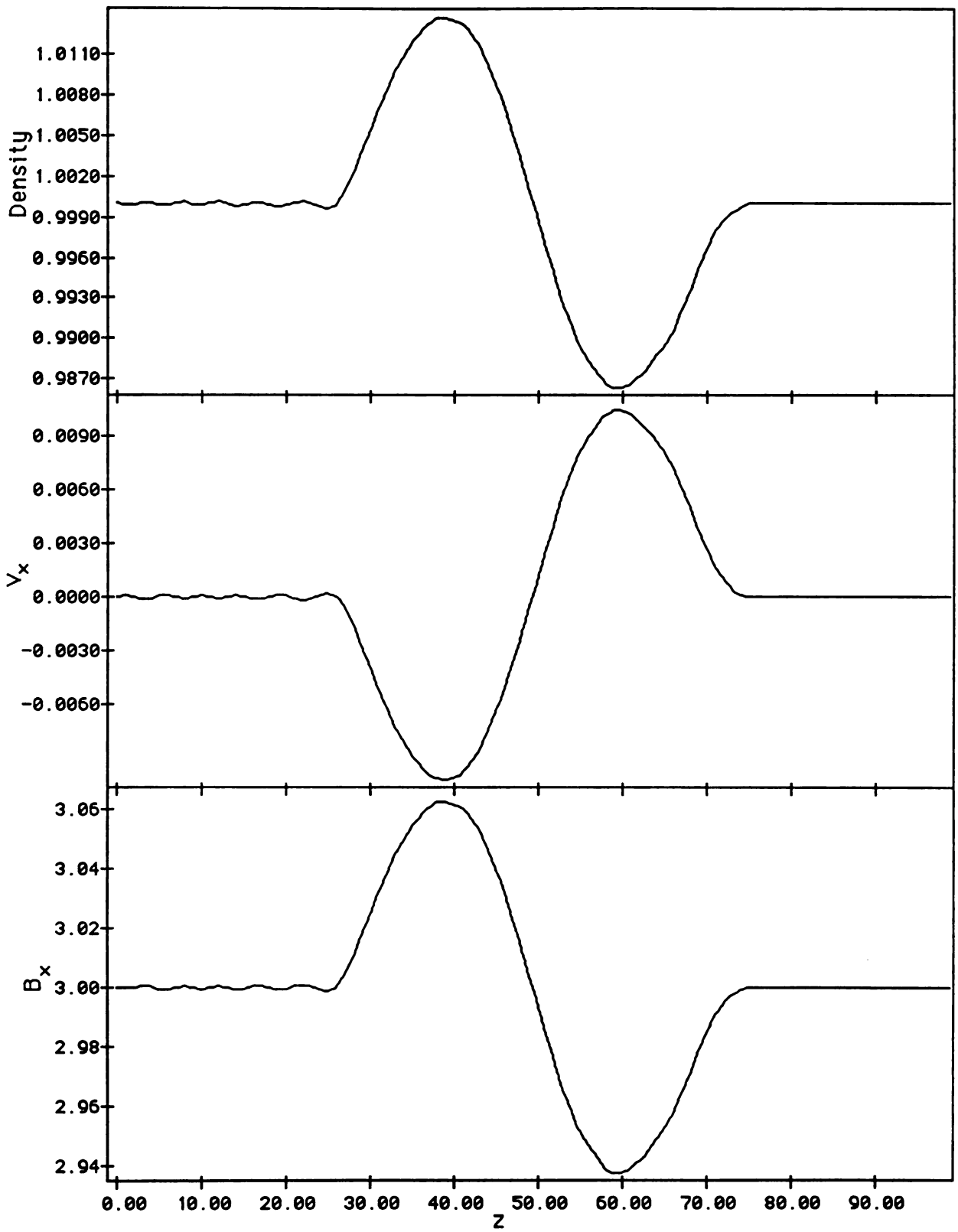


Figure 3.17: Fast mode magnetoacoustic wave. t=50.

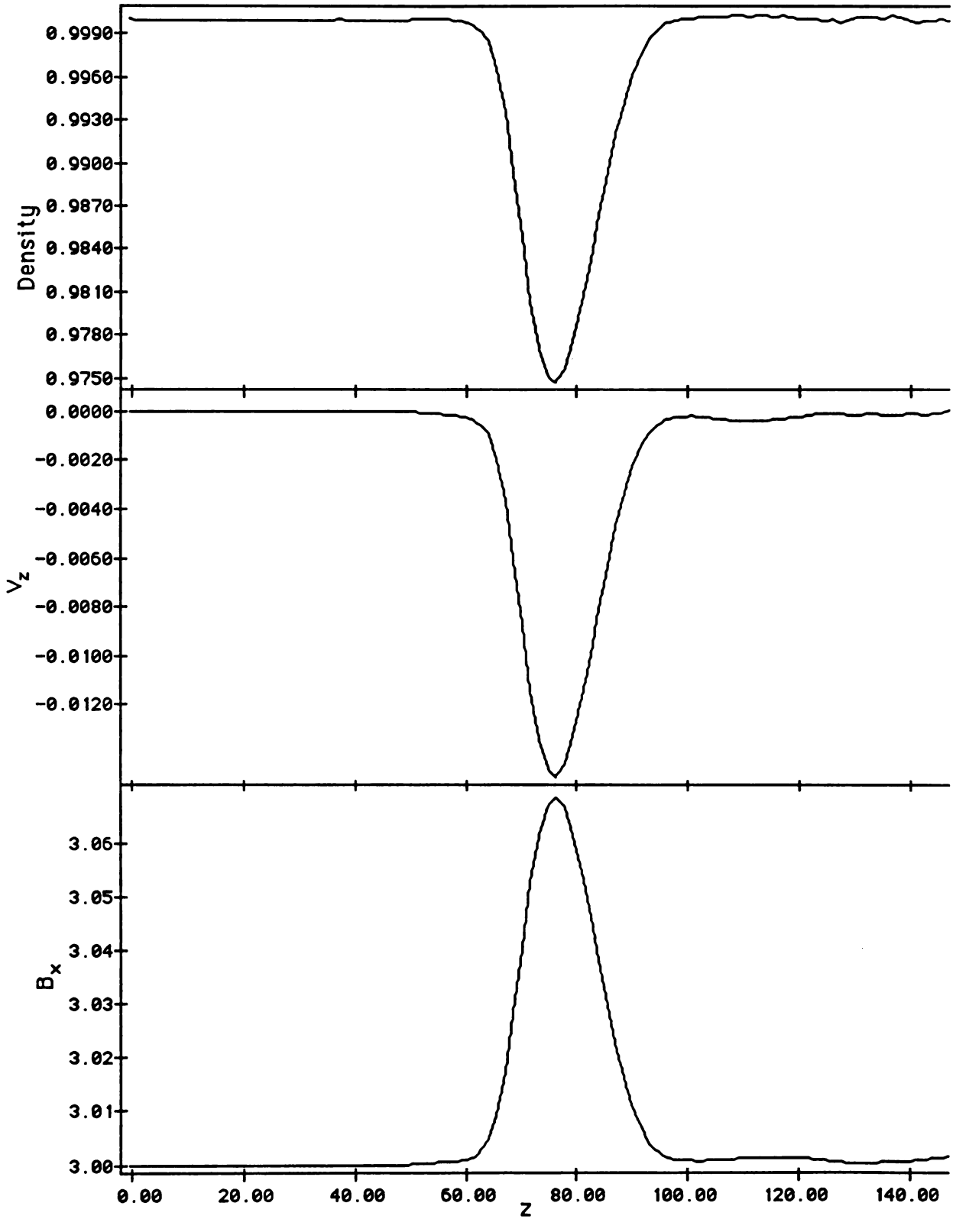


Figure 3.18: Slow mode magnetoacoustic wave pulse. t=150 , η=.01 .

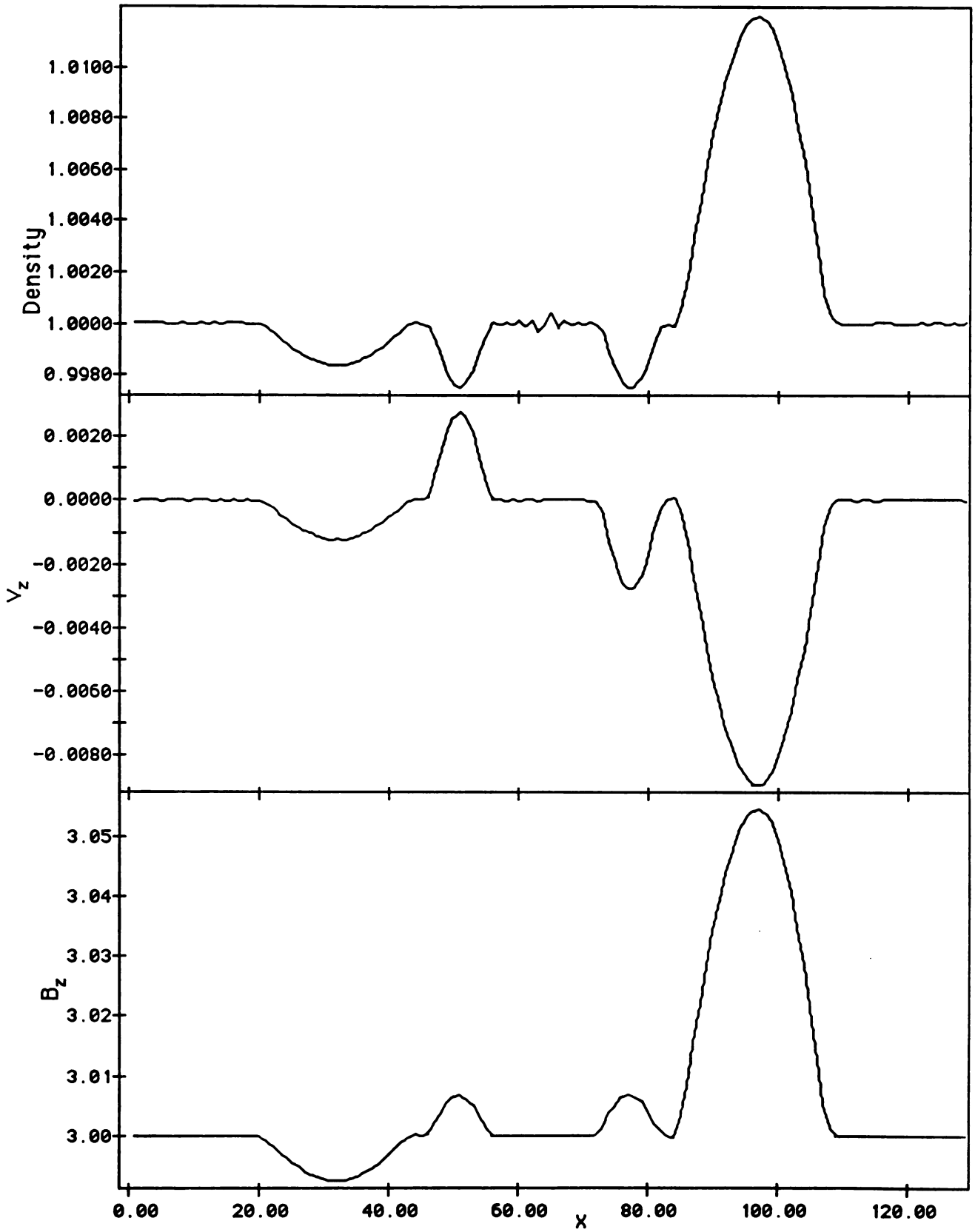


Figure 3.19: Magnetoacoustic wave pulses generated by driving with +x fast mode phase relations. t=30 , η=0.

3.7 Nonlinear Waves in Magnetic Atmosphere

The last class of test problems considered was that of shock waves and rarefaction waves propagating through a uniform magnetic atmosphere. Theoretical solutions are available for the simpler cases and the results for the cases for which theoretical solutions are not known will be needed for comparison in chapter 4. For the simplest case of waves propagating in the direction of the field, the problem may be solved theoretically. The waves propagate exactly as they would in the absence of the magnetic field and the the solutions from section 3.1 and 3.2 still pertain.

For the case where the waves propagate perpendicular to the field, theoretical solutions are also available. The solution to the piston driven shock given in appendix C, can be extended to include the magnetic field; however, the resulting equation in α, the ratio of densities, is cubic. Analysis of this equation revealed one real root which was found numerically for the chosen initial fluid parameters and several values for the piston speed. These values are listed in table 3.4. The results obtained from the simulation were in excellent agreement (±0.1%) with these values. Sample output is given in figure 3.20.

v_p/c	ρ_1	T_1	v_s	B_1
0.010	1.00765	1.00507	1.31539	3.02296
0.050	1.03852	1.02572	1.34818	3.11555
0.100	1.07799	1.05158	1.38227	3.23396
0.250	1.20061	1.13274	1.49617	3.60182
0.500	1.41553	1.28602	1.70328	4.24658
1.000	1.85403	1.70618	2.17092	5.56209

Table 3.4: Theoretical Solution for Shock— B_2=3.0

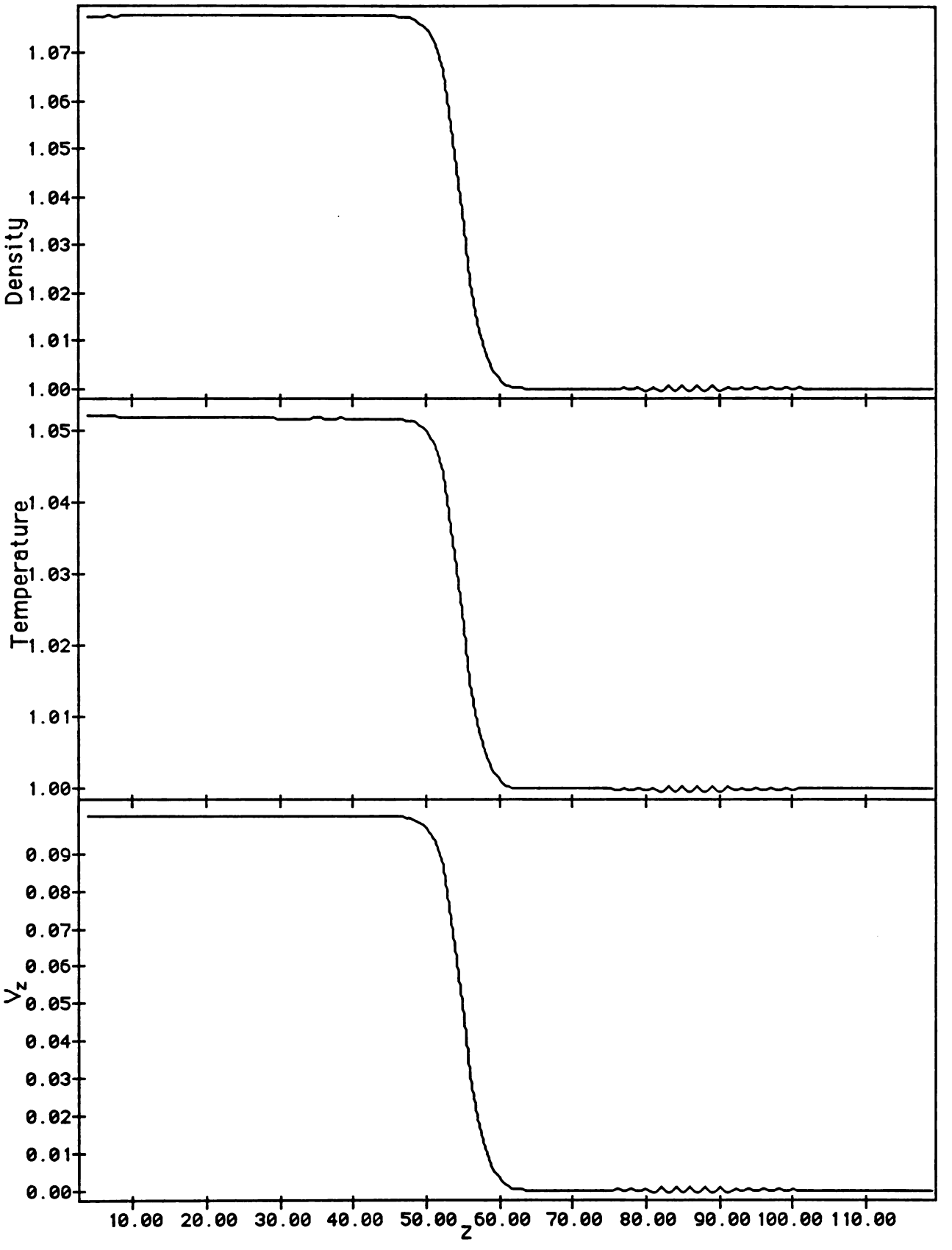


Figure 3.20a: Shock moving normal to a uniform magnetic field. V_p=0.1, B_2=3.0

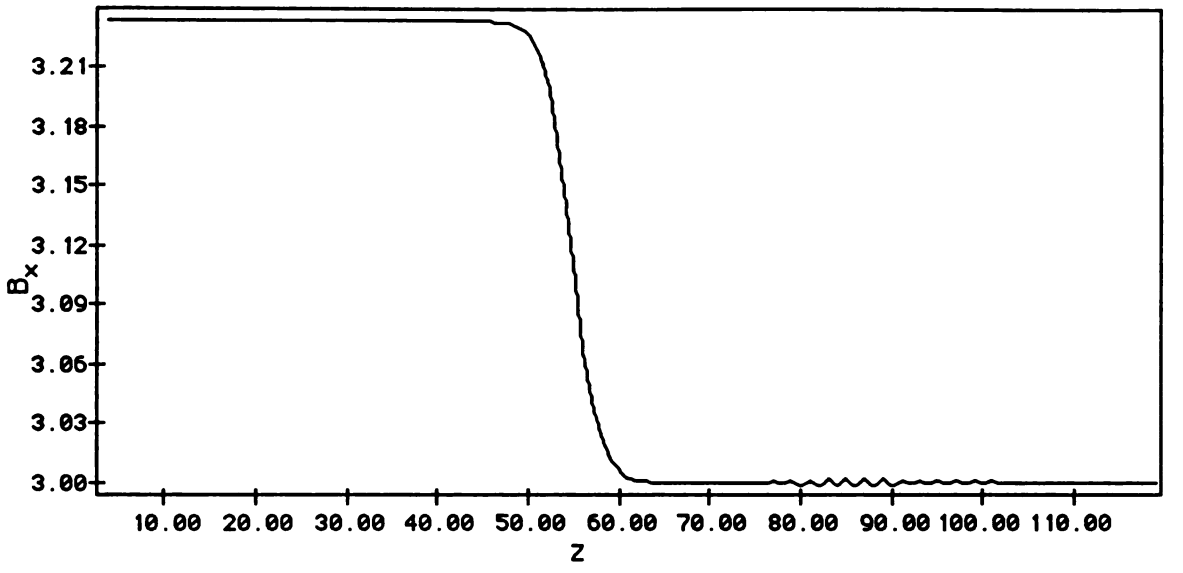


Figure 3.20b: Shock moving normal to a uniform magnetic field. V_p=0.1 B_2=-3.0

I was unable to find a solution in the literature for a rarefaction wave in a magnetic fluid. The solution used in section 3.3 (Landau & Lifshitz, 1959, §92) is a similarity solution with the acoustic speed as the characteristic velocity. I assumed that the presence of the magnetic field would not introduce a characteristic length and, therefore, the similarity solution could still be used. The characteristic velocity, in this case, would be the magnetoacoustic wave speed (only one mode is possible for propagation perpendicular to the field). Table 3.5 lists the values given by this solution for several piston speeds.

v_p/c	ρ_1	T_1	v_b	B_1
0.010	0.99239	0.99492	1.29670	2.97717
0.050	0.96232	0.97472	1.24340	2.88696
0.100	0.92559	0.94976	1.17670	2.77677
0.250	0.82105	0.87682	0.97670	2.46315
0.500	0.66483	0.76174	0.64337	1.99449

Table 3.5: Theoretical values for rarefactions— B_2=3.0

Once again the computer simulation of this problem yielded results in agreement with the analytic values (±0.1%). Sample output is given in figure 3.21.

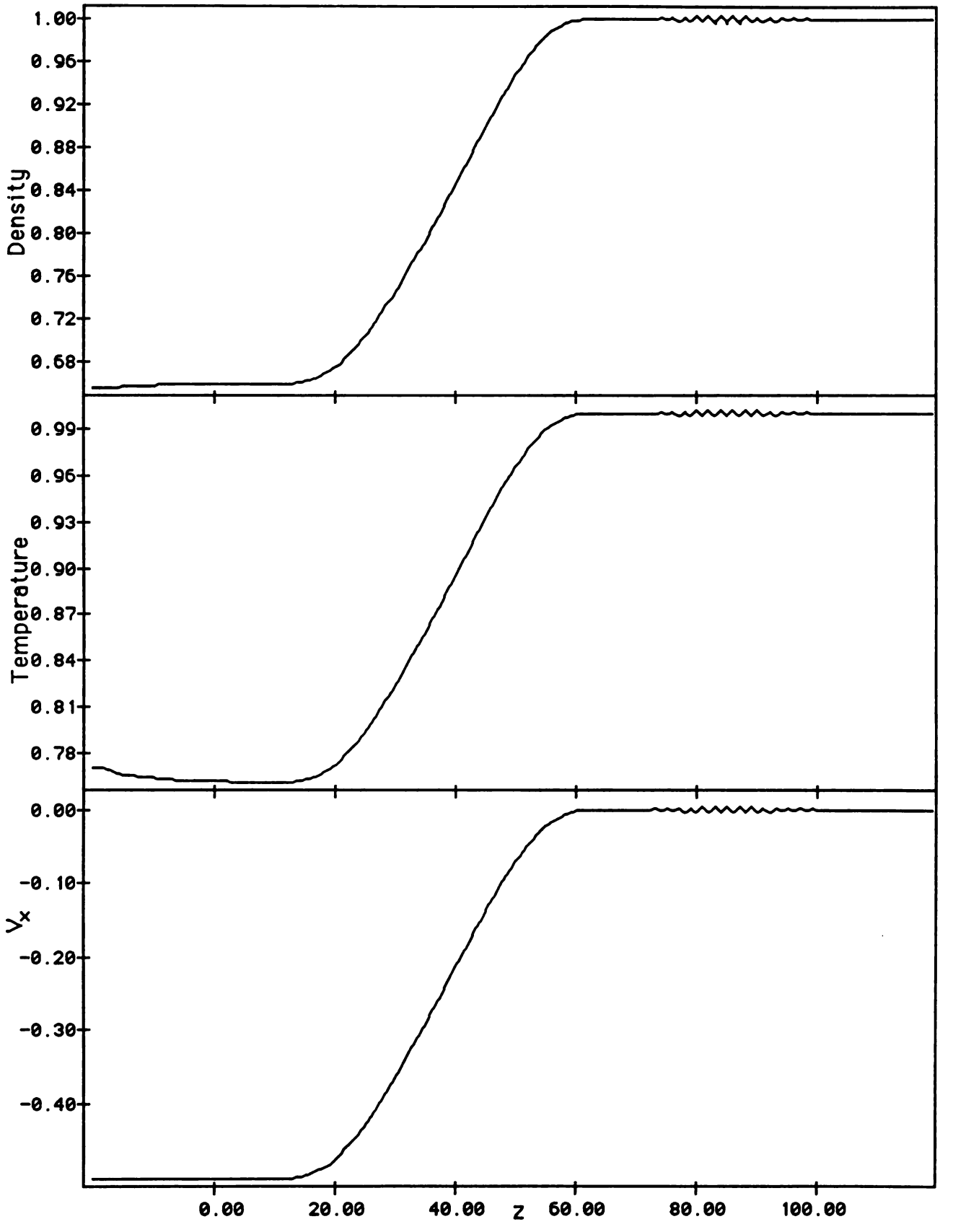


Figure 3.21a: Rarefaction moving normal to a uniform magnetic field. V_p=-0.5 B_3=3.0

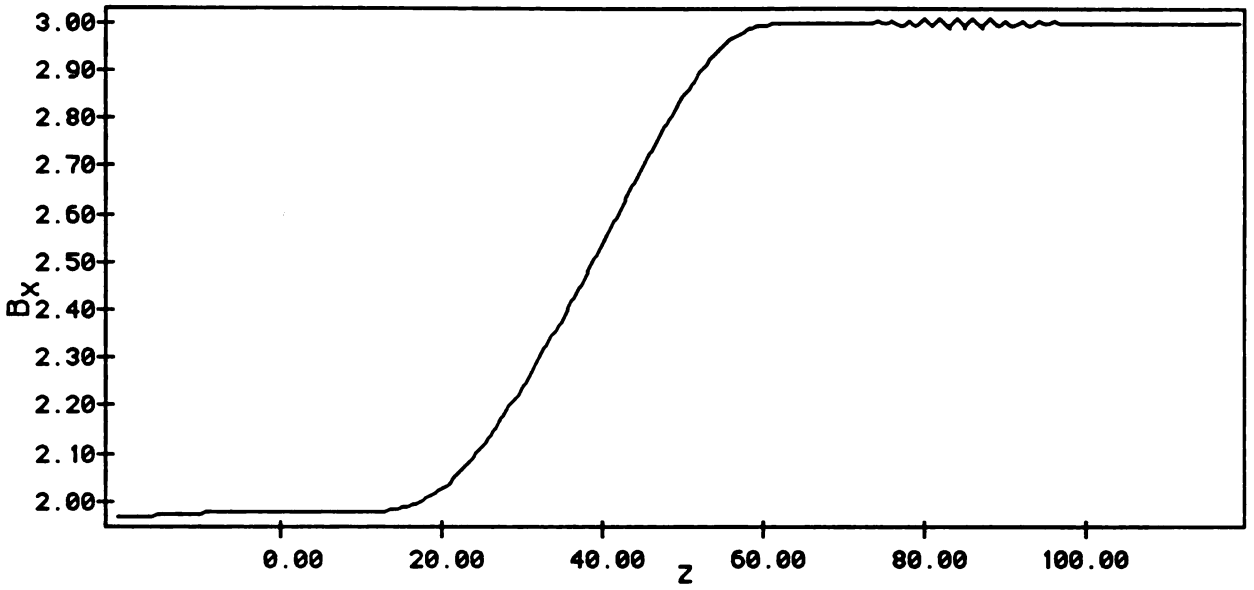


Figure 3.21b: Rarefaction moving normal to a uniform magnetic field.

For the most general case of a shock propagating at an angle to the magnetic field, the equation in α, the ratios of densities, is cubic in α^2 and cannot be solved analytically. An analysis given by Hughes and Young (Hughes & Young, 1966, §10.4) reveals that there are three real solutions for α and v_s. The first solution is called a slow shock since it reduces to a slow mode magnetoacoustic wave in the small amplitude limit. The second solution, called a fast shock, reduces to a fast mode magnetoacoustic wave. The third solution, call an intermediate shock, has a shock speed which approaches the Alfvén speed in the small amplitude limit.

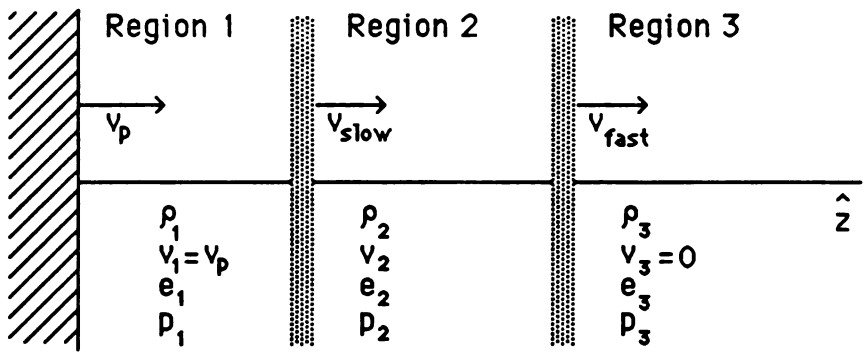


Figure 3.22: Shocks resulting from driving a piston at an angle to the magnetic field.

Figure 3.22 illustrates what happens when a piston is driven into the fluid at an angle to the magnetic field. The boundary values specified at the piston did not include the transverse velocity or magnetic field; therefore, a frictionless piston was being

modeled. In this case, any or all of these type of shocks may result from the piston. Output for a sample run is given in figure 4.4. As can be seen, two shocks result and propagate independently. The first and larger shock is a fast shock and the second is a slow shock. No intermediate shock was generated. The results for the case Bx=Bz=3.0 (θ=45°) are summarized in the following table.

V_p	ρ_1	v_{x1}	T_1	B_{x1}	ρ_2	v_{x2}	v_{z2}	T_2	B_{x2}
0.010	1.0081	-0.0025	1.0054	3.023	1.0062	-0.0046	0.0089	1.0041	3.028
0.050	1.0411	-0.0112	1.0273	3.112	1.0309	-0.0226	0.0443	1.0205	3.140
0.100	1.0837	-0.0208	1.0555	3.221	1.0624	-0.0443	0.0887	1.0414	3.281
0.250	1.218	-0.042	1.145	3.541	1.161	-0.1048	0.224	1.107	3.709
0.500	1.460	-0.063	1.315	4.062	1.338	-0.1908	0.454	1.235	4.438
1.000	1.98	-0.06	1.77	5.03	1.724	-0.309	0.935	1.604	5.89

Table 3.6: Shocks caused by driving a piston at θ=45° to a uniform magnetic field.

These results can be used to determine what transverse velocity must be specified to generate any fast shock. This was done, iteratively if necessary, to obtain α and v_s as a function of v_p for the fast shocks. Since the slow shock in figure 3.22 moves into a region whose fluid parameters were determined by the fast shock, it was more difficult to determine what transverse velocity was required to produce the desired slow shock moving into the undisturbed fluid.

v_p	ρ	v_x	T	B_x	v_s
0.000	1.0000	0.0000	1.0000	3.000	
0.010	1.0069	-0.0052	1.0046	3.032	1.453
0.050	1.0351	-0.025	1.0232	3.157	1.482
0.100	1.070	-0.049	1.0470	3.317	1.519
0.250	1.180	-0.116	1.122	3.794	1.634
0.500	1.377	-0.206	1.268	4.58	1.839

Table 3.7: Post shock fluid values for a oblique fast shock.

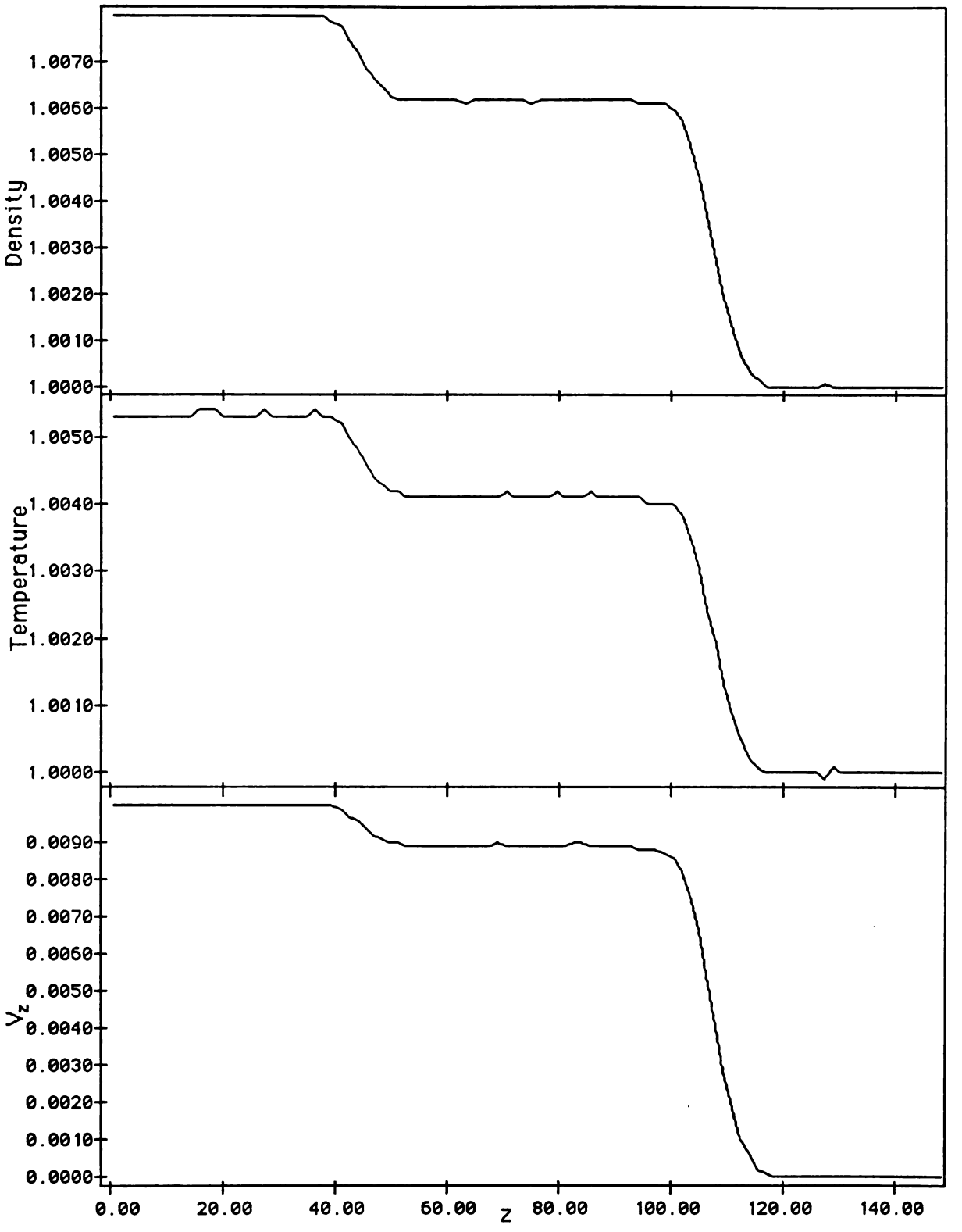


Figure 3.23a: Piston driven oblique shocks. V_p=0.01. B_{2x}=B_{2z}=3.0

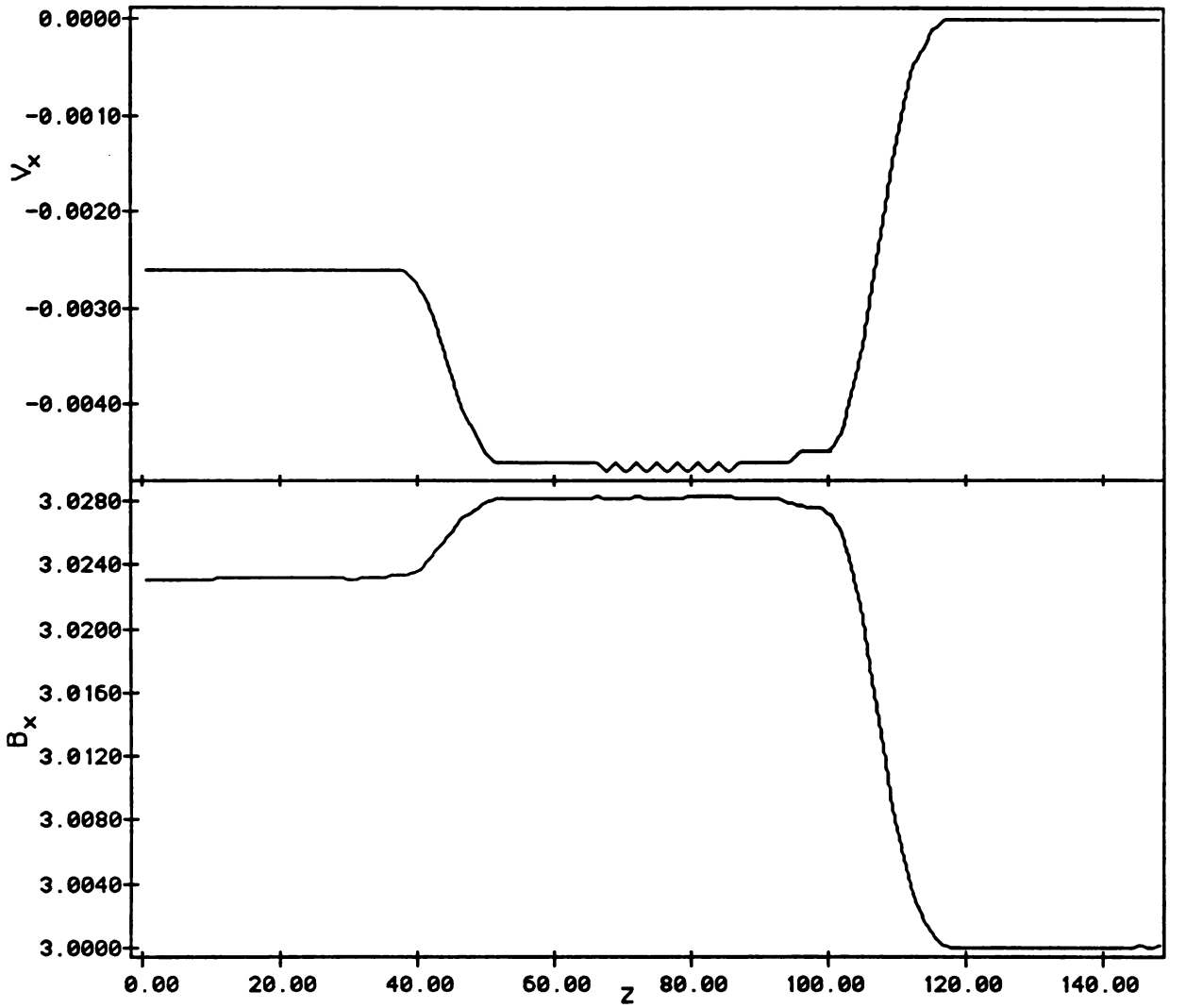


Figure 3.23b: Piston driven oblique shocks.

To isolate the dependence of slow and fast shocks on the angle between the shock and the magnetic field, I ran the piston at the same speed at different angles. The results are summarized in table 3.8.

θ	ρ_1	v_{x1}	T_1	θ_1	ρ_2	v_{x2}	v_{z2}	T_2	θ_2
0	1.1032	0.0000	1.0679	0.00					
15	1.1013	-0.0073	1.0668	16.30	1.0886	-0.0334	0.0914	1.0585	17.95
30	1.0967	-0.0123	1.0639	32.12	1.0823	-0.0351	0.0918	1.0544	33.28
45	1.0906	-0.0142	1.0600	47.30	1.0805	-0.0293	0.0955	1.0533	47.86
60	1.0849	-0.0126	1.0563	61.86	1.0794	-0.0206	0.0984	1.0525	62.08
75	1.080	-0.0077	1.0537	76.05	1.0784	-0.0105	0.0997	1.0519	76.07
90					1.0779	0.0000	0.1000	1.0517	90.00

Table 3.8: Post shock fluid values for different angles between piston and field.

When the piston was withdrawn from the fluid, two rarefactions resulted. This case is illustrated in figure 3.24. The two waves have leading edge velocities equal to the slow and fast magnetoacoustic wave speeds for the region, so I have called them fast and slow rarefactions. Sample output is given in figure 3.25.

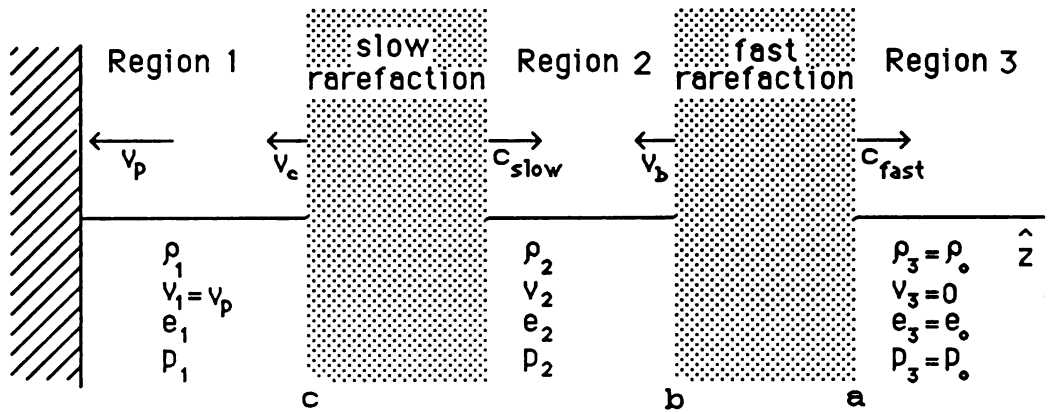


Figure 3.24: Rarefactions resulting from withdrawing a piston at an angle to the field.

The results of running the simulations for a number of different piston velocities are given in the following table.

V_p	ρ_1	v_{x1}	T_1	B_{x1}	ρ_2	v_{x2}	v_{z2}	T_2	B_{x2}
-0.010	0.9921	0.0028	0.9947	2.976	0.9938	0.0047	-0.0090	0.9958	2.972
-0.050	0.9607	0.0139	0.9736	2.883	0.9695	0.0238	-0.0446	0.9795	2.859
-0.100	0.9219	0.0275	0.9473	2.771	0.9399	0.0481	-0.0883	0.9596	2.722
-0.250	0.810	0.068	0.869	2.450	0.858	0.124	-0.213	0.904	2.329
-0.500	0.640	0.141	0.744	1.970	0.747	0.259	-0.395	0.824	1.735
-1.000	0.364	0.316	0.527	1.149	0.625	0.510	-0.617	0.732	0.868

Table 3.9: Rarefactions caused by driving a piston at θ=45° to a uniform magnetic field.

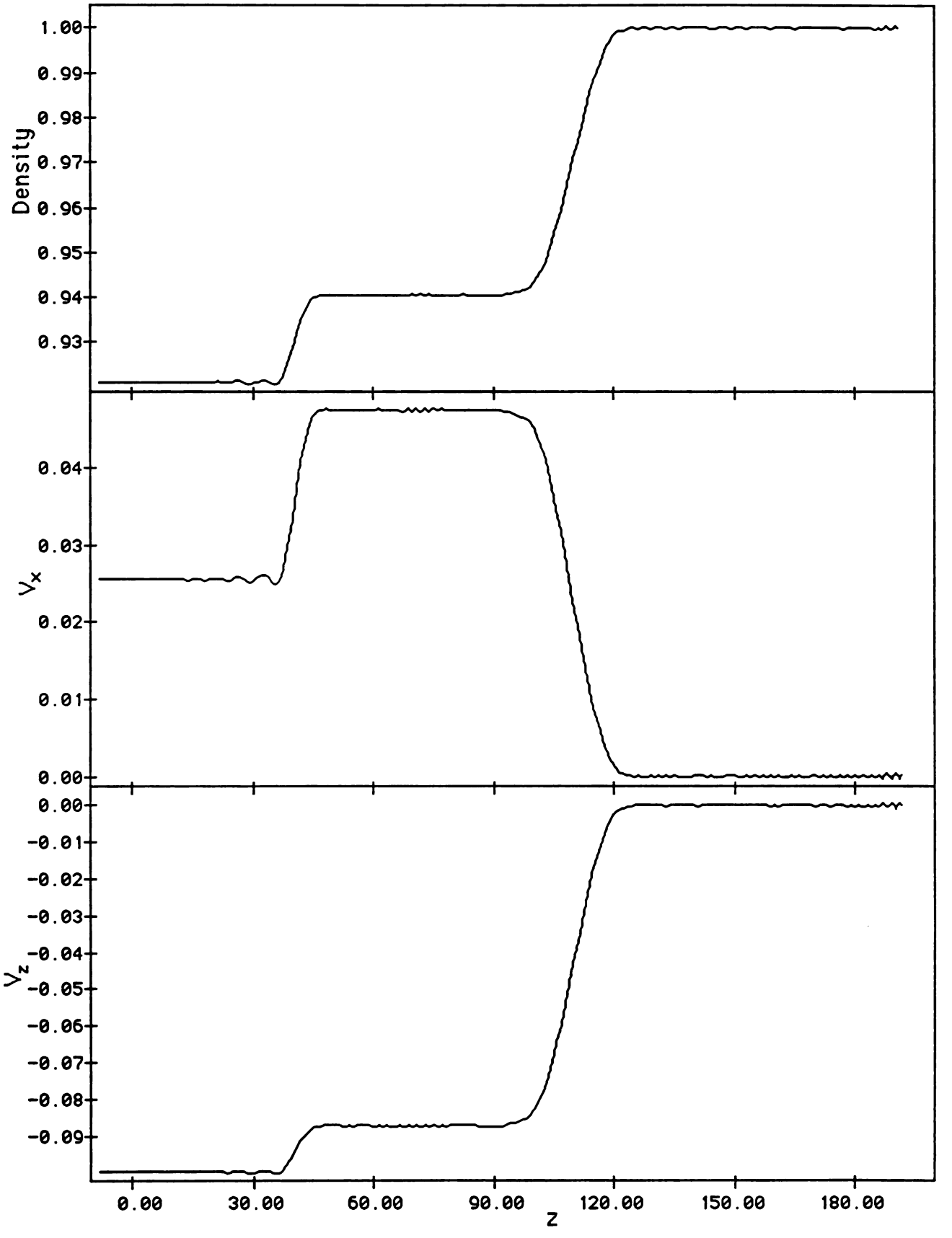


Figure 3.24a: Sample output for oblique rarefactions. v_p=0.1 , B_{3x}=B_{3z}=3.0

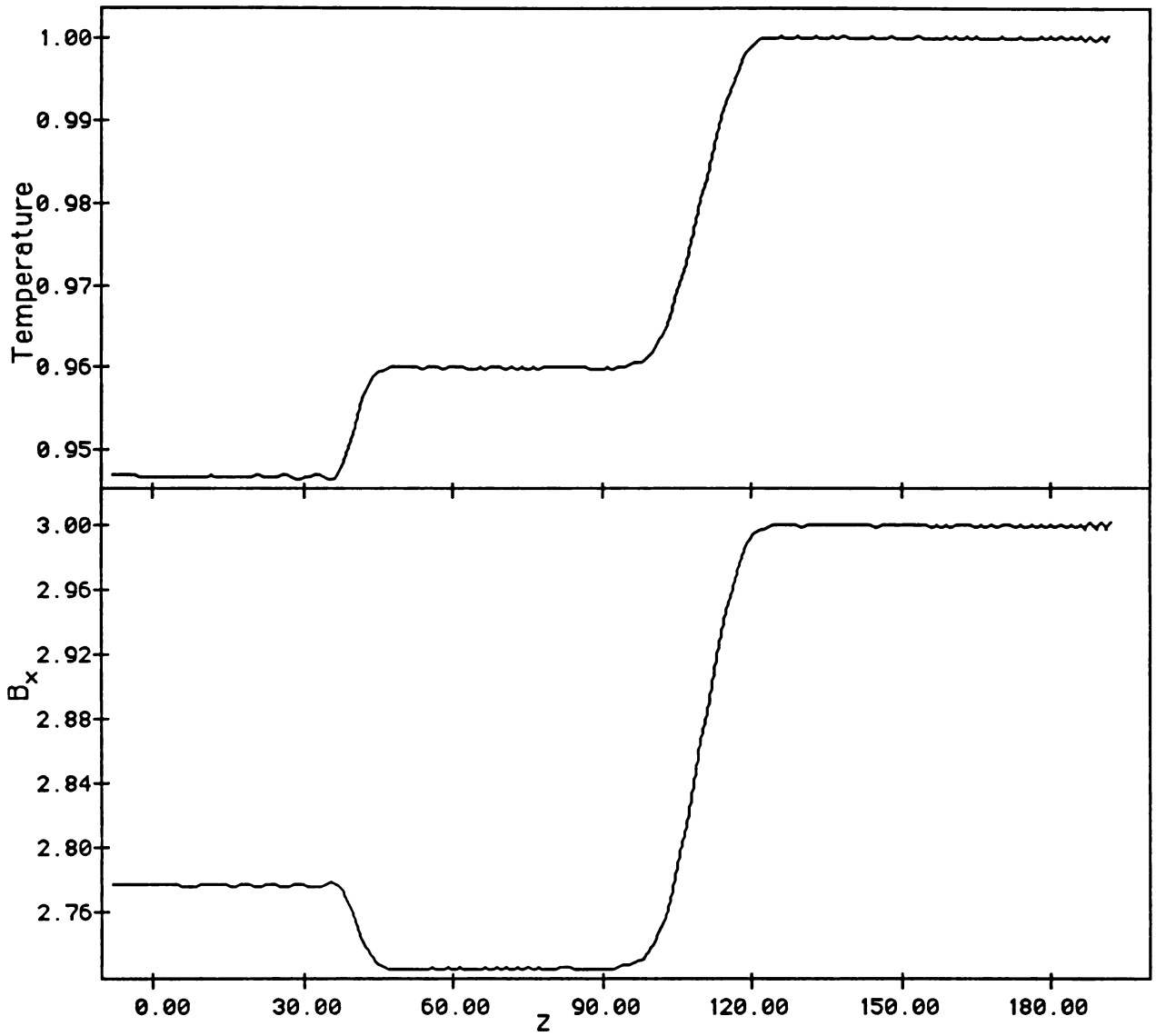


Figure 3.24b: Sample output for oblique rarefactions.

These results for nonlinear waves in a uniform magnetic field will be used for comparison when I consider the propagation of nonlinear waves through magnetically structured media.

4
Waves in Magnetically Structured Media

In the first three chapters, I have developed and tested a computer program for modeling nonlinear disturbances in magnetic fluids. The program was next used to model several problems for which thetheoretical solution are not known. These problems all consider waves in magnetically structured media.

4.1 Nonlinear Waves Incident Normal to a Magnetic Flux Slab

The program is able to model the propagation of nonlinear waves through a magnetically structured atmosphere. Theoretical solutions to problems of this type are generally unavailable.

The simplest problem of this type is the normal incidence of a nonlinear wave on a magnetic flux slab. This slab consists of region of high magnetic field embedded in an environment of weaker magnetic field. The simple slab considered here has a magnetic field given by

$$B_o(z) = \begin{cases} B_o\hat{x}, & |z-z_o| < a \\ 0, & |z-z_o| > a \end{cases}$$

The region is initially isothermal and the environment has a higher density to balance the total internal and external pressures. The wave, propagating in the z direction, is normal to the surface of the slab.

The computer program was used to model this problem for a variety of shock waves and magnetic field strengths. From the results of these simulations, I can draw the following conclusions:

1. Each time a wave crosses an interface, a reflection is possible. The magnitude of the reflection depends on the strength of the magnetic field within the slab. As a result of these reflections, the slab sends out wave pulses in both directions. The amplitude of these waves decline with each reflection and only the initial reflections were easily followed back through the environment. The amplitude of the pulses depend on the strength of the shock and the magnitude of the magnetic field. The width of the pulse also depends on the width of the slab.
2. After the shock passes and the slab reaches its final state, the slab is again in pressure equilibrium with the environment. It is narrower and the magnetic field strength is increased. The slab is no longer in thermal equilibrium with the environment. For all cases tried the slab was cooler than the environment. The density of the slab is increased by a lower ratio than that of the environment.
3. The slab is still at rest relative to the environment.
4. The final state of the environment is the same as it would have been if the slab had not been present. The two regions of the environment, one on each

side of the slab, are still identical.

5. The final state of the slab is denser, hotter, and has a stronger magnetic field than if a shock had passed through a uniform region with the same initial conditions.

The results of the simulations are given on the following table. The ratio of densities, α_{in}, is also the ratio of final magnetic field to initial magnetic field. The last column is the ratio of final to initial width for the slab.

B_{slab}	v_p	α_{in}	α_{out}	T_{slab}/T_{env}	W_f/W_i
0.10	0.100	1.1031	1.1032	0.9996	0.905
0.50	0.010	1.010	1.010	0.9999	0.990
	0.050	1.051	1.051	0.9998	0.951
	0.100	1.102	1.103	0.9993	0.907
	0.250	1.266	1.268	0.998	0.789
1.00	0.010	1.0099	1.0099	0.9999	0.990
	0.050	1.050	1.050	0.9997	0.953
	0.100	1.101	1.102	0.9992	0.908
	0.250	1.261	1.268	0.998	0.796
3.00	0.010	1.0093	1.010	0.9995	0.991
	0.050	1.047	1.051	0.9976	0.954
	0.100	1.094	1.103	0.994	0.912
	0.250	1.246	1.269	0.991	0.801
	0.500	1.519	1.564	0.987	0.656
10.00	0.010	1.0086	1.010	0.999	0.991
	0.050	1.043	1.051	0.995	0.957
	0.100	1.087	1.103	0.991	0.917
	0.250	1.226	1.270	0.981	0.810

Table 4.1: Results of simulating a shock incident normal to a slab.

Next, I used an outward moving piston to generate rarefactions which were normal to the slab. From the results, I can make the following conclusions.

1. Wave pulses are generated similar to those produced by the shocks—though opposite in magnitude.
2. After the rarefaction passes and the slab reaches its final state, the slab is again in pressure equilibrium with the environment. It is wider and the magnetic field strength is decreased. The slab is no longer in thermal

equilibrium with the environment. For all cases the slab was hotter than the environment.

3. The slab is still at rest relative to the environment.
4. The final state of the environment is the same as it would have been if the slab had not been present. The two regions of the environment, one on each side of the slab, are still identical.
5. The final state of the slab is less dense, cooler, and has a weaker magnetic field than if a shock had passed through a uniform region with the same initial conditions.

The results of the simulations are summarized in the following table.

B_{slab}	v_p	α_{in}	α_{out}	T_{slab}/T_{env}	W_f/W_i
0.10	-0.010	0.9900	0.9900	1.000	1.010
	-0.050	0.9508	0.9507	1.000	1.053
	-0.100	0.9033	0.9032	1.000	1.108
	-0.250	0.7700	0.7698	1.000	1.300
0.50	-0.010	0.9900	0.9900	1.000	1.010
	-0.050	0.9510	0.9507	1.000	1.053
	-0.100	0.9036	0.9032	1.000	1.109
	-0.250	0.7708	0.7701	1.000	1.305
1.00	-0.010	0.990	0.990	1.000	1.010
	-0.050	0.951	0.951	1.000	1.052
	-0.100	0.904	0.903	1.000	1.107
	-0.250	0.772	0.770	1.002	1.297
3.00	-0.010	0.991	0.990	1.000	1.010
	-0.050	0.954	0.951	1.002	1.050
	-0.100	0.915	0.903	1.008	1.093
	-0.250	0.779	0.770	1.007	1.284
	-0.500	0.600	0.578	1.024	1.667
10.00	-0.010	0.992	0.990	1.001	1.009
	-0.050	0.958	0.951	1.005	1.046
	-0.100	0.917	0.903	1.009	1.095
	-0.250	0.800	0.770	1.027	1.253

Table 4.2: Results of simulating a rarefaction normal to a slab.

It is of interest to note that if a shock is followed by a rarefaction driven by the same piston speed, the slab and environment each return to its initial state. Therefore a

source of shocks and rarefactions, such as a convection zone, would not serve to strengthen or weaken a flux slab provided the source produced identical distributions of shock and rarefaction strengths.

4.2 Nonlinear Waves Incident Normal to a Magnetic Flux Tube

As a second magnetic structure, I considered a magnetic flux tube whose magnetic field was given by

$$\mathbf{B}_o = \begin{cases} B_o\hat{y}\,, & x^2+z^2<r^2 \\ 0\,, & x^2+z^2>r^2 \end{cases}$$

Since the symmetry of the tube is not the same as that of the rectangular grid, I used a gradual boundary described by a hyperbolic tangent:

$$\mathbf{B}_o=\frac{1}{2}B_o\hat{y}\left[1+\tanh\left(r-\sqrt{x^2+y^2}\right)\right]$$

The numerical region was initially isothermal and the density varied to maintain the pressure balance.

The shock generated by the moving piston propagated in the +z direction and was therefore incident normal to tube boundary. Figure 4.1 shows a series of density plots as a shock passes the tube. Also shown in the figure, are contour plots of the temperature and speed after the shock has passed. In addition to noise, one can clearly see the difference in temperature and speed between the tube and its environment. The noise pictured is due to fluctuations less than the accuracy requested for the simulation. They are visible because of the near uniformity of the atmosphere (note the small contour spacings).

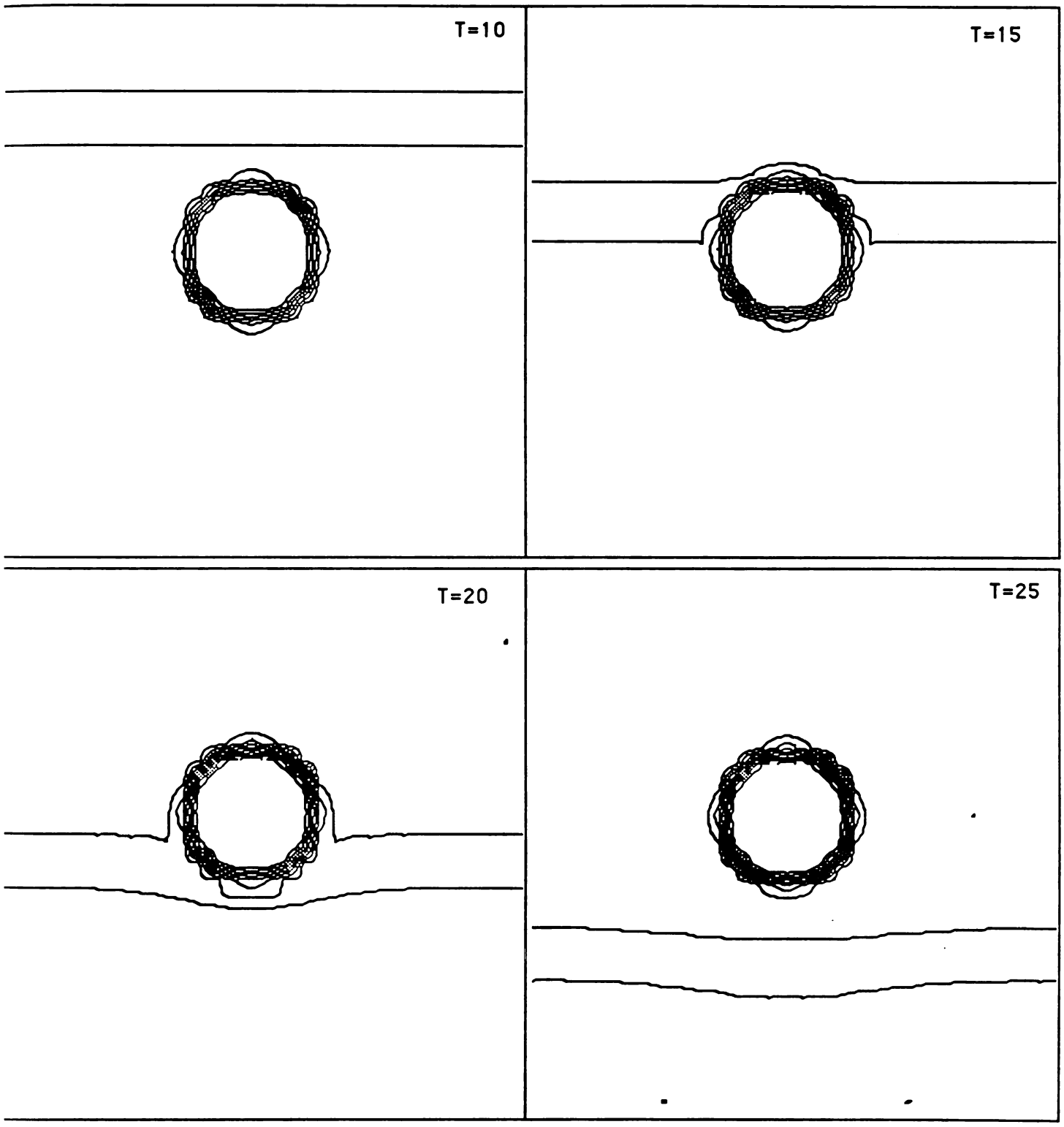


Figure 4.1a: Contour plot of density as a shock wave passes a flux tube.
$B_o = 3.0$, $\gamma_p = 0.1$

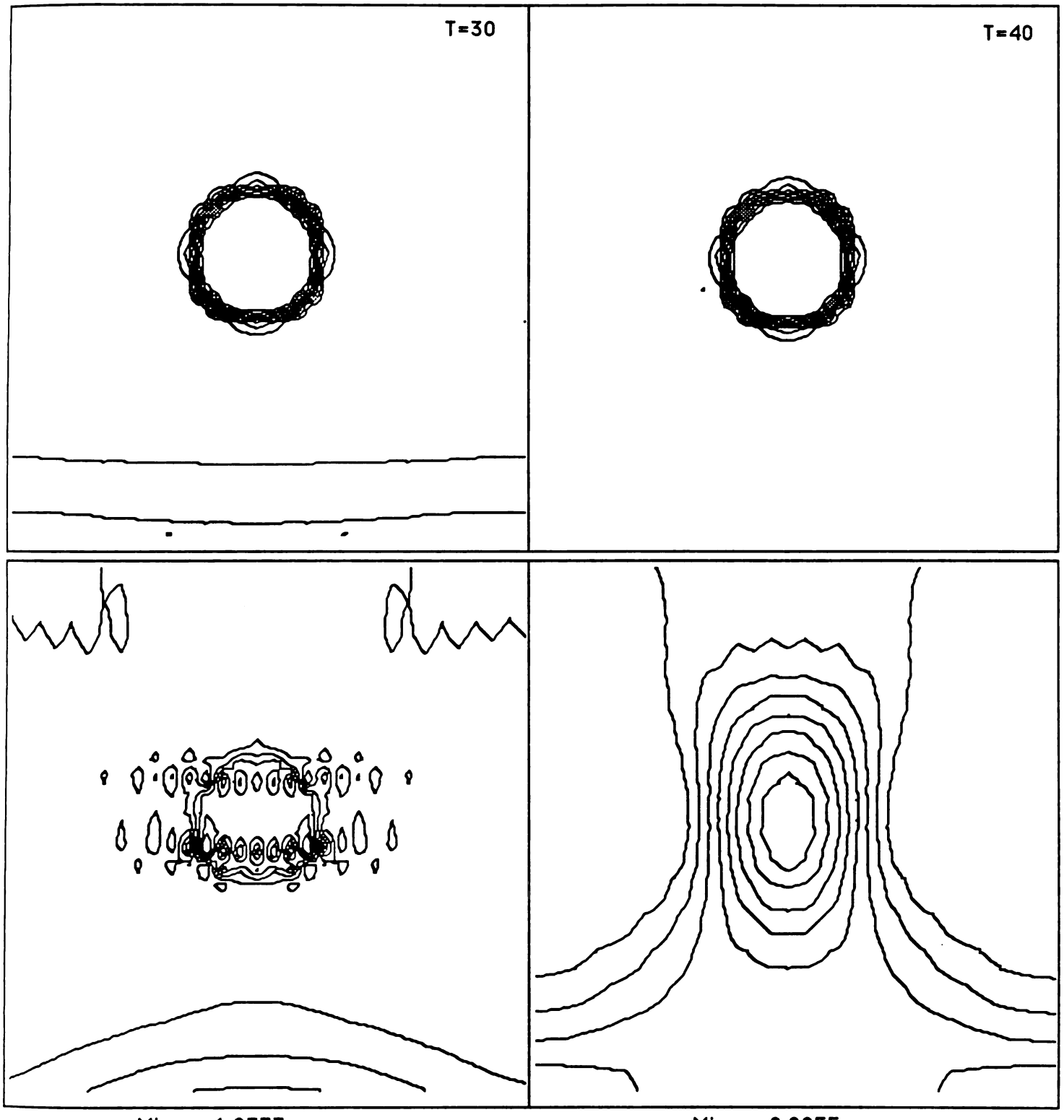


Figure 4.1b: Contour plots of density as a shock wave passes. Also showns are contour plots of temperature and speed after the shock.

Because of the amount of time required for each run, I did not use as large a variety of shock strengths and field strengths as for the slab. From the results of these simulations I can draw the following conclusions:

1. After the shock passes and the tube reaches its final state, it is in pressure equilibrium with the environment. The tube is no longer in thermal equilibrium with the environment. For all cases, the tube was cooler than the environment.
2. The density of the tube has increased by a lower ratio than the environment.
3. The tube is no longer cylindrical in cross section. The z dimension has been decreased by a fraction similar to that of the slab. The x dimension increases slightly.
4. The tube is not at rest relative to the environment but rather has a relative velocity in the +z direction. The fluid in the environment moves around the tube as it passes.
5. Far from the tube, the environment is the same as it would have been if the tube had not been present. Near the tube, there is a "wake" region where the velocity is higher and the temperature and density are slightly lower. To the sides (±x) of the tube, the z velocity is slightly slower. This effect decreases farther from the tube.
6. The final state of the tube is denser, hotter, and has a stronger magnetic field than if the shock had passed through a uniform region with the tube's internal conditions. The average final density and temperature of the tube were slightly less than those of a slab with the same internal initial conditions.
7. There was some evidence that the tube was not as uniform after the shock as before. The center was less dense, cooler, moving faster. The magnetic field was greatest near the center of the tube. To quantify these results would require a larger grid than is practical on the Vax.

The results of the simulations are given on the following table.

B_{slab}	v_p	v_{in}	α_{in}	α_{out}	T_{slab}/T_{env}	$W_{\|}/W_i$	$W_{\perp}/W_i$
0.10	0.100	0.1001	1.1030	1.1032	1.000	1.000	0.907
0.30	0.100	0.1002	1.1029	1.1032	0.999	1.000	0.908
1.00	0.100	0.1013	1.1017	1.1032	0.998	1.001	0.908
3.00	0.010	0.012	1.009	1.010	0.999	1.000	0.991
	0.050	0.055	1.045	1.051	0.997	1.002	0.954
	0.100	0.11	1.09	1.10	0.994	1.004	0.907
	0.250	0.264	1.25	1.27	0.989	1.008	0.792
	0.500	0.53	1.50	1.56	0.986	1.039	0.637

Table 4.2: Results of simulating a shock incident normally on a flux tube.

Next rarefaction waves were generated normal to the tube. From the results of the numerical simulation I can make the following conclusions:

1. After the rarefaction passes and the tube reaches its final state, it is in pressure equilibrium with the environment. The tube is no longer in thermal equilibrium with the environment. For all cases, the tube was hotter than the environment.
2. The density of the tube has decreased less than the environment.
3. The tube is no longer cylindrical in cross section. The z dimension has been increased by a fraction similar to that of the slab and the x dimension decreases slightly.
4. The tube is not at rest relative to the environment but rather has a relative velocity in the -z direction. The fluid in the environment moves around the tube as it passes.
5. Far from the tube, the environment is the same as it would have been if the tube had not been present. Near the tube, there is a "wake" region where the velocity is higher and the temperature and density are slightly greater. To the sides (±x) of the tube, the z velocity is slightly slower. This effect decreases farther from the tube.
6. The final state of the tube is less dense, cooler, and has a weaker magnetic field than if the shock had passed through a uniform region with the tube's internal conditions. The average final density and temperature of the tube were slightly greater than those of a slab with the same internal initial conditions.

The results of the simulations are given on the following table.

B_{slab}	v_p	v_{in}	α_{in}	α_{out}	T_{slab}/T_{env}	$W_{\|\|}/W_i$	$W_\perp/W_i$
0.10	-0.100	-0.1005	0.9033	0.9033	1.000	1.000	1.107
0.30	-0.100	-0.1006	0.9034	0.9033	1.000	1.000	1.107
1.00	-0.100	-0.1014	0.9045	0.9033	1.001	0.999	1.105
3.00	-0.010	-0.0115	0.991	0.990	1.000	1.000	1.109
	-0.050	-0.055	0.956	0.951	1.002	0.998	1.045
	-0.100	-0.108	0.912	0.903	1.005	0.996	1.095
	-0.250	-0.261	0.788	0.770	1.013	0.993	1.27
	-0.500	-0.51	0.60	0.579	1.038	0.971	1.66

Table 4.2: Results of simulating a rarefaction incident normally on a flux tube.

Because of the coarse grid and final non-uniformity, I am unable to make a conclusion regarding the effect of alternating shocks and rarefactions. A larger grid and greater time would be required than is practical on the Vax.

4.3 Acoustic Waves Incident Obliquely to a Magnetic Interface

As the last and most interesting problem, I considered acoustic waves incident obliquely on a plane parallel magnetic interface. The results of this simulation have immediate application to several outstanding problems in the solar atmosphere. The most important of these is the reported absorption of solar p-mode oscillations by sun spots (Braun et al, 1987). The details of this problem and the application of the results of this simulation to this problem are considered in the discussion section of chapter 5.

The magnetic structure for this problem was a plane interface between a field free region and a region of uniform magnetic field. The numerical domain was originally isothermal and the density in the the field free region was increased to maintain pressure equilibrium. The number and range of controllable parameters in this problem, as well as the large amount of CPU time required for each run, made it impossible to simulate the full range of problems possible. Therefore, the internal magnetic field strength was chosen to give a magnetic beta similar to that of a sun spot and the direction of the field was chosen normal to both the interface and the direction of wave propagation. These conditions are illustrated in figure 4.2. Several different magnetic field profiles were used for the interface. None of the profiles used was based on a sun spot model.

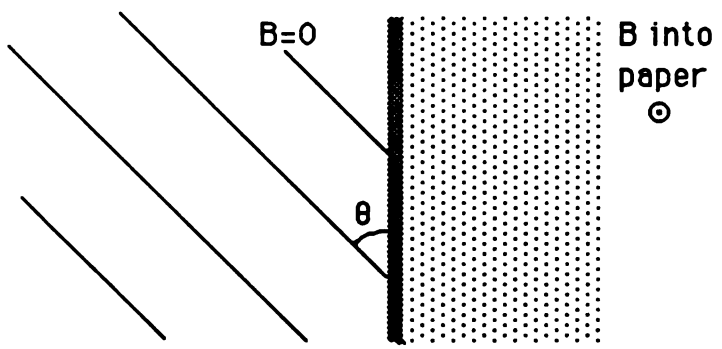


Figure 4.2: Acoustic wave incident at angle θ to the interface

To study the large scale effects and the angular dependences, I chose the following profile:

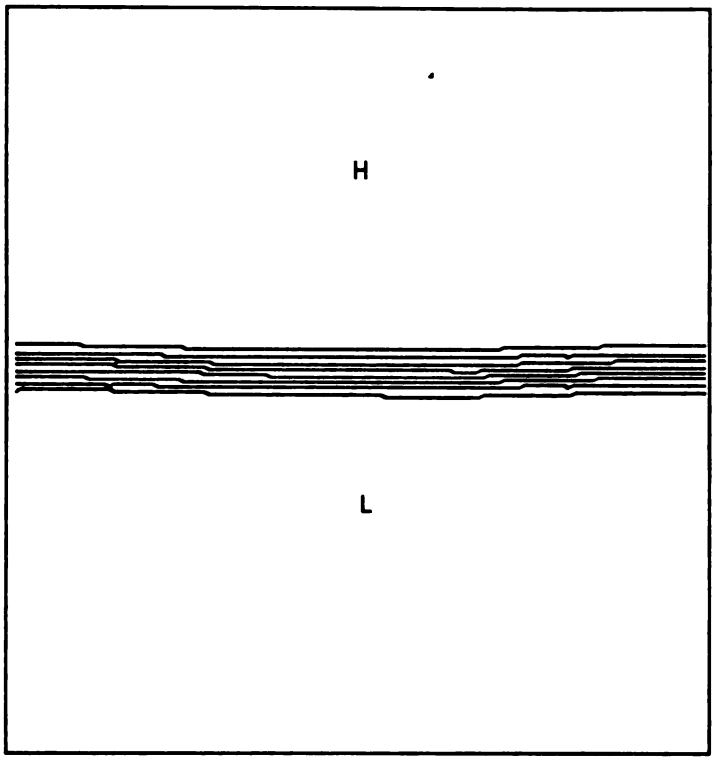
$$B_y=\frac{B_o}{2}\left[1+\tanh\left(\frac{z-z_o}{s}\right)\right]$$

The parameter s determined the width of the interface and B_o was the interior magnetic field. The cells were uniform in size. A variety of values was tried for s without effecting the large scale results. A value $s=2L_z$ was chosen for the remaining simulations with this profile.

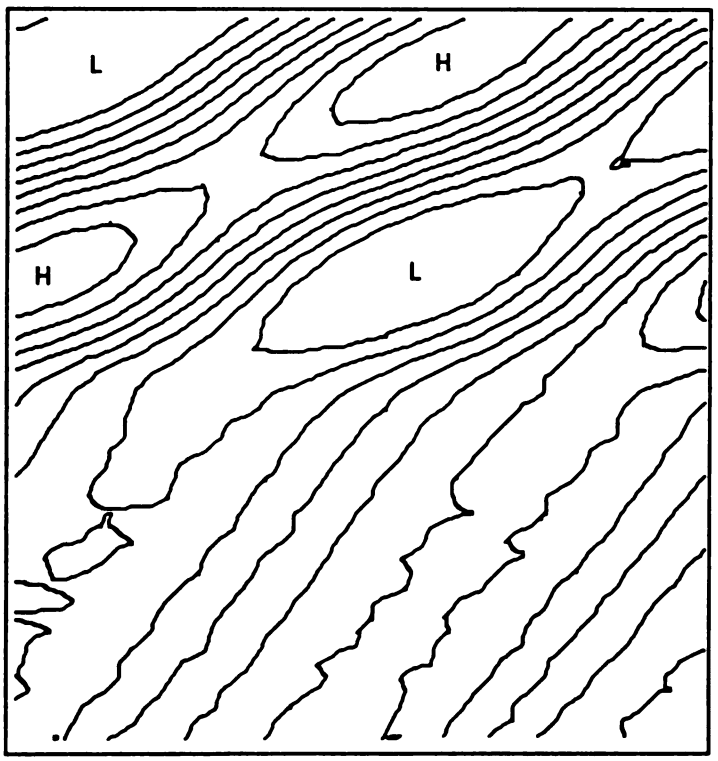
A piston moving with velocity v_p was used to drive an acoustic wave from the lower boundary at an angle θ to the interface. The wavelength, λ, was specified and the width of the cells in the spectral direction was redefined to meet the periodic boundary conditions. The required angular frequency for the driving force was calculated from the wavelength and the speed of sound. The simulation was run long enough for the transient affects to disappear— at least 8 periods for all frequencies used. The simulation was run for a variety of wavelengths and angles. The large scale results were the same for all choices of wavelength within the range $10Lz \leq \lambda \leq 100Lz$. The angle of incidence was central in determining whether the wave would be transmitted or reflected. The results can summarized as follows:

1. For small angles of incidence, the wave is mostly transmitted. For $\theta \leq 15°$, the energy flux in the z direction leaving the interface was greater than 95% of the incident flux. The transmitted wave was refracted away from the normal as expected.
2. For angles greater than 30°, the wave was mostly reflected. For $30° \leq \theta \leq 60°$, the energy flux in the z direction leaving the interface was less than 5% of the incident flux. A standing wave resulted from the interference of the incident and reflected waves.

These results are illustrated in the following figures. Figure 4.3 shows contour plots of the flow resulting for $\theta=15°$ and $v_p=0.01$. The refraction of the wave can be clearly seen. The noise in the temperature graph is smaller in magnitude than the accuracy requested of the simulation and is apparent because of the nearly uniform temperature. Figure 4.4 illustrates the results for $\theta=45°$. The standing wave pattern can be clearly seen in the region above the interface. The distance between the piston and the interface is not an integral number of half-wavelengths and the results are similar for the range of wavelengths simulated.

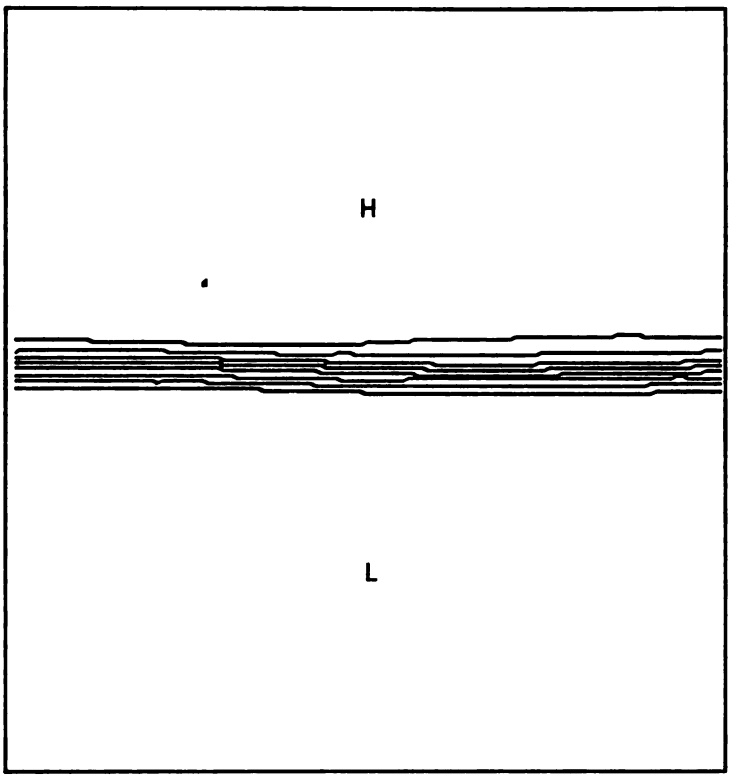


Min = 0.9947
max= 4.7818
contours= 0.37871
DENSITY AT T= 200.00

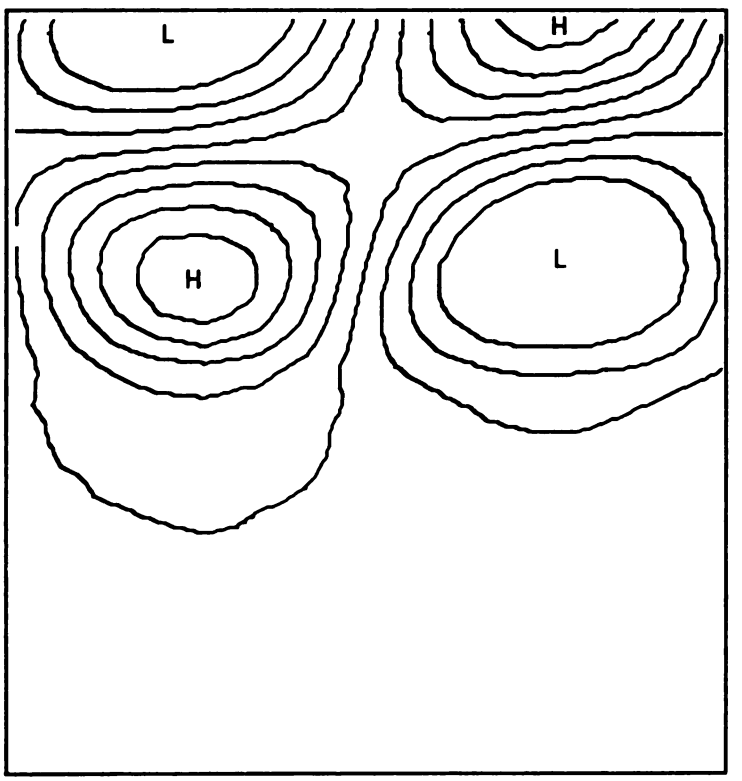


Min = 0.9928
max= 1.0073
contours= 0.00145
TEMPERATURE AT T= 200.00

Figure 4.3a: Contour plots of density and temperature for θ=15°, and v_p=0.01.

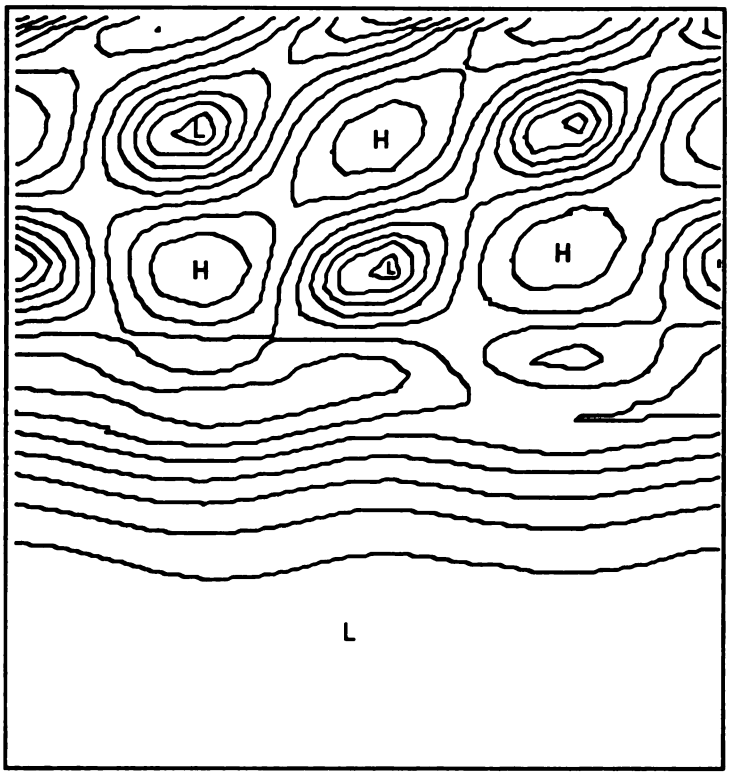


Min = 0.9977
max= 4.9097
contours= 0.39119
DENSITY AT T= 200.00

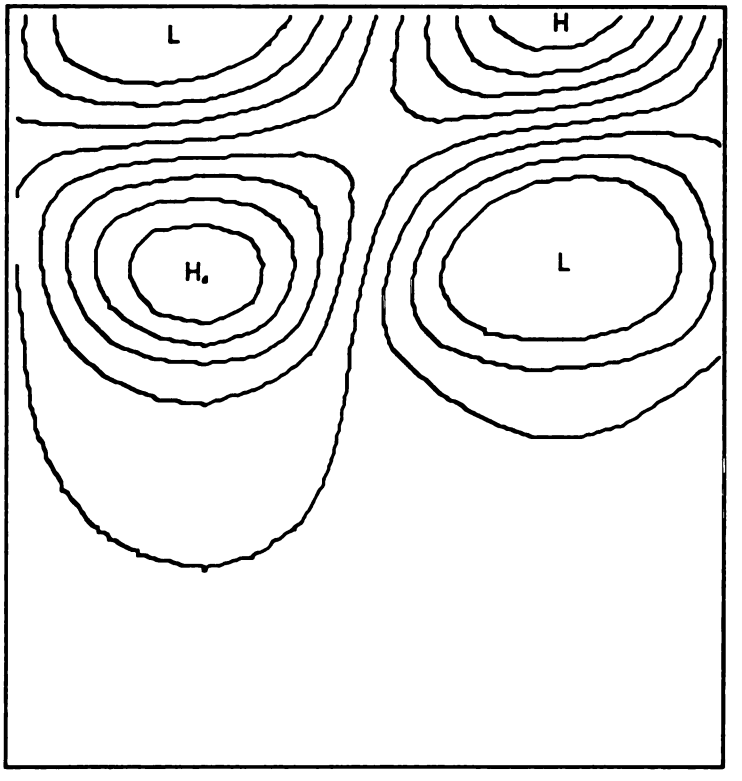


Min = 0.9768
max= 1.0264
contours= 0.00496
TEMPERATURE AT T= 200.00

Figure 4.4: Contour plots of density and temperature for θ=45° and v_p=0.01.



Min = 0.0000
max= 0.0275
contours= 0.00275
SPEED AT T= 200.00



Min = 4.0048
max= 4.5329
contours= 0.05281
TOTAL ENERGY AT T= 200.00

Figure 4.4b: Contour plots of speed and total energy density for θ=45° and v_p=0.01.

A second interface profile was chosen to study the small scale details. Since the interior Alfvén speed exceeds ω/k_x for $\theta \geq 30°$, a zone of resonant absorption was expected within the interface (Steinolfson, 1984) (Tataronis, 1975). To concentrate on this region, I chose the following profile:

$$B_y(z)=\begin{cases} 0 & z<z_b \\ a_0+a_1(z-z_o)+a_2(z-z_o)^2+a_3(z-z_o)^3 & z_b \leq z \leq z_o \\ c_0+c_1(z-z_o)+c_2(z-z_o)^2+c_3(z-z_o)^3 & z_o \leq z \leq z_t \\ B_o & z_t<z \end{cases}$$

The coefficients were chosen such that the Alfvén speed equaled ω/k_x at z_o for $\theta=45°$ and both B and $\partial B/\partial z$ were continuous everywhere. The value of z_t and z_b were chosen to determine the width of the interface and nonuniform cell spacing was used. Half of the cells were within the interface while one quarter was above and one quarter below the interface. The values used for the majority of the simulations gave an interface width about one quarter the thickness of the tanh profile and a cell size one tenth as large. This profile is illustrated in figure 4.5. In addition to the usual output, I had the program calculate and print the *average* density, velocities, and temperature for a full cycle as a function of z; therefore, small differences in these variables could be differentiated from the periodic variation of the wave.

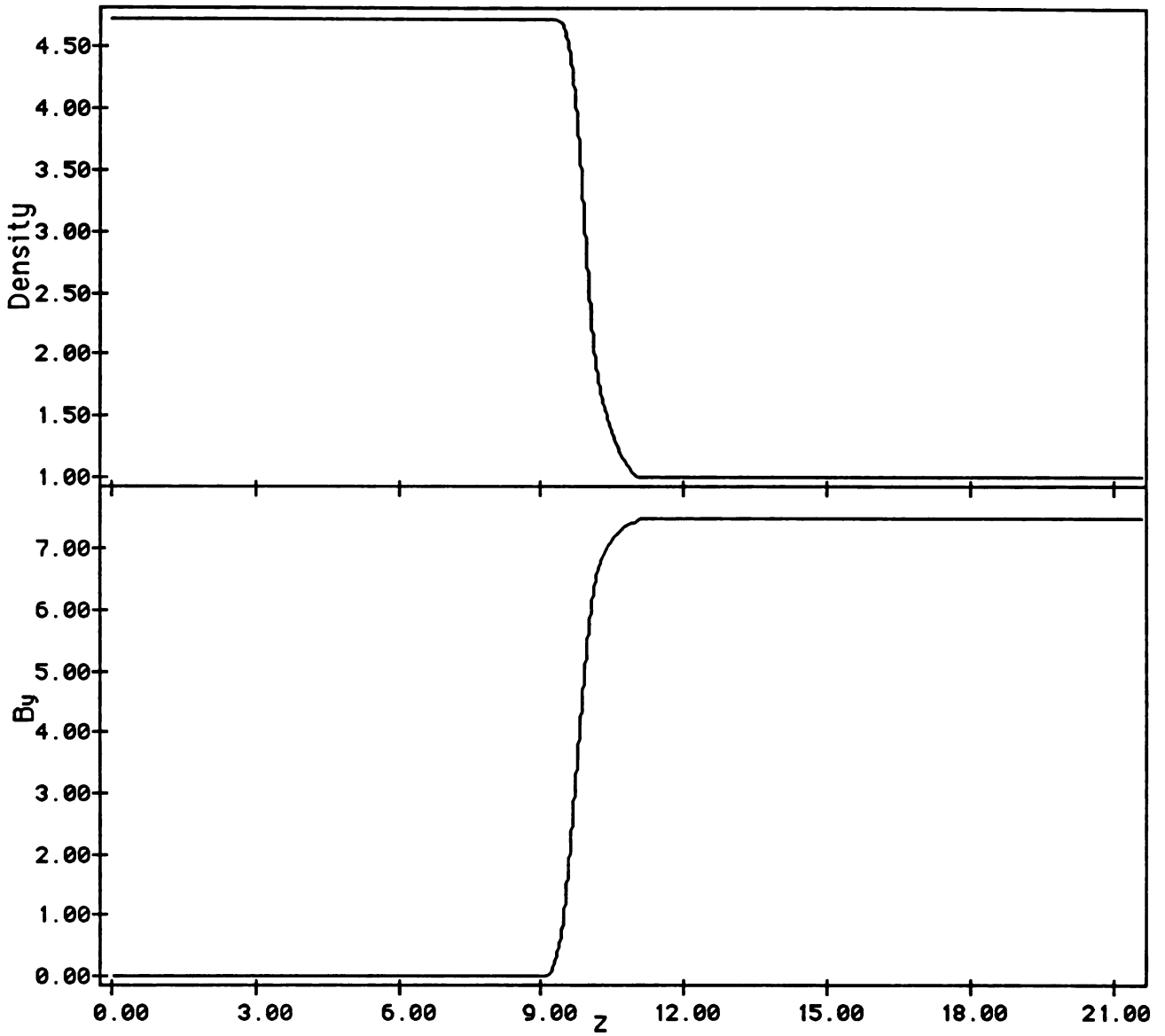
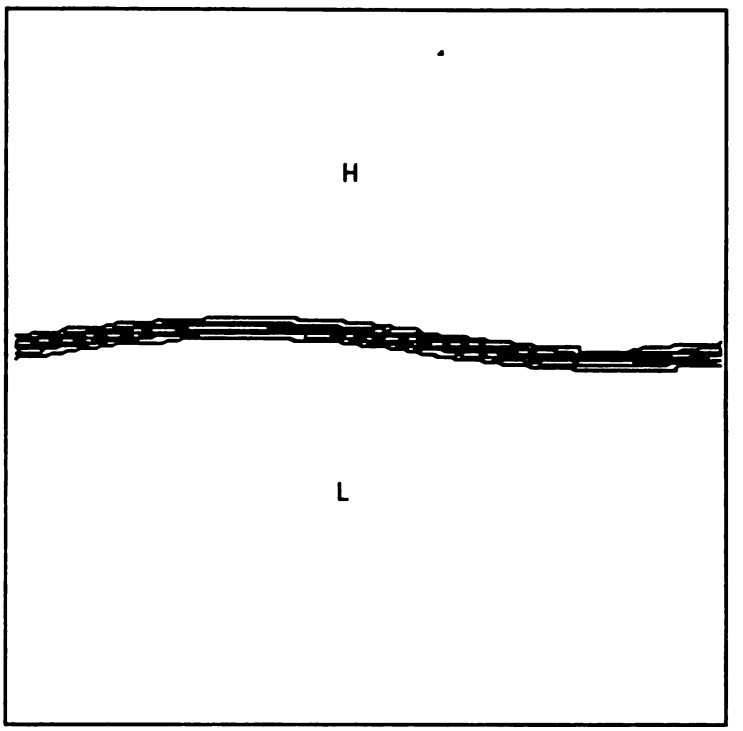
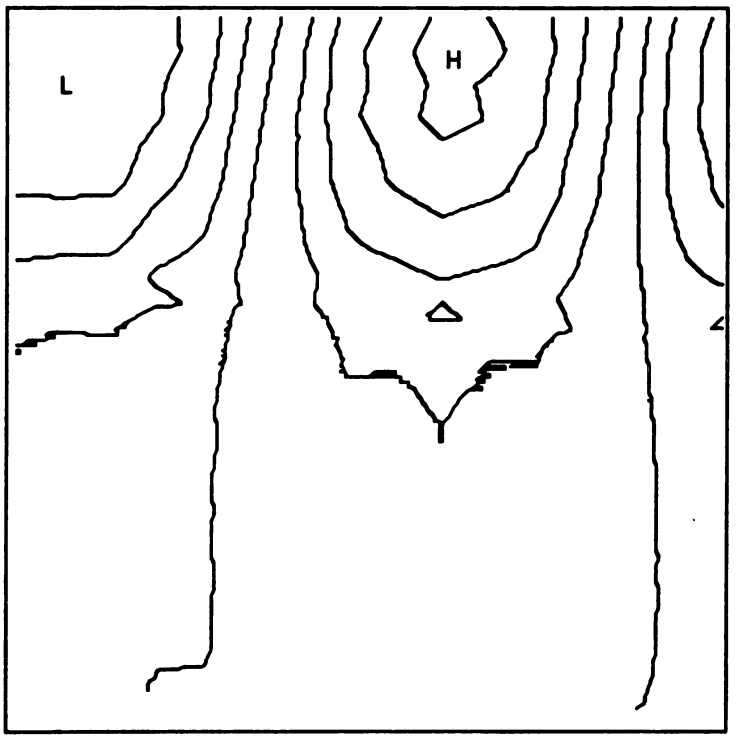


Figure 4.5: Profile of density and By. z_o=10.00, z_b=9.00, and z_t=11.00.

The finer grid required a smaller time step and thus considerably more CPU time. Each simulation required 50-60 hours of CPU time and thus only a few trials were run. For θ=15°, the results were consistent with the earlier simulations. No resonant heating was observed. For θ=45°, resonant heating was clearly evident in cells near z_o. Figure 4.5 shows contour plots of density and total energy for this simulation. The region shown is only 40% as large in z as figure 4.4. Although the total energy plot is shifted horizontally because of a difference in phase, the results are clearly consistent with the earlier simulation. There is only a hint of the resonance zone in this graph. The heated zone is clearly visible in figure 4.7 which shows temperature versus z for the central (in x) zone.



Min = 0.9966
max= 4.8002
contours= 0.38035
DENSITY AT T= 200.00



Min = 4.1367
max= 4.3750
contours= 0.02383
TOTAL ENERGY AT T= 200.00

Figure 4.6: Contour plot of density and total energy for the fine grid.

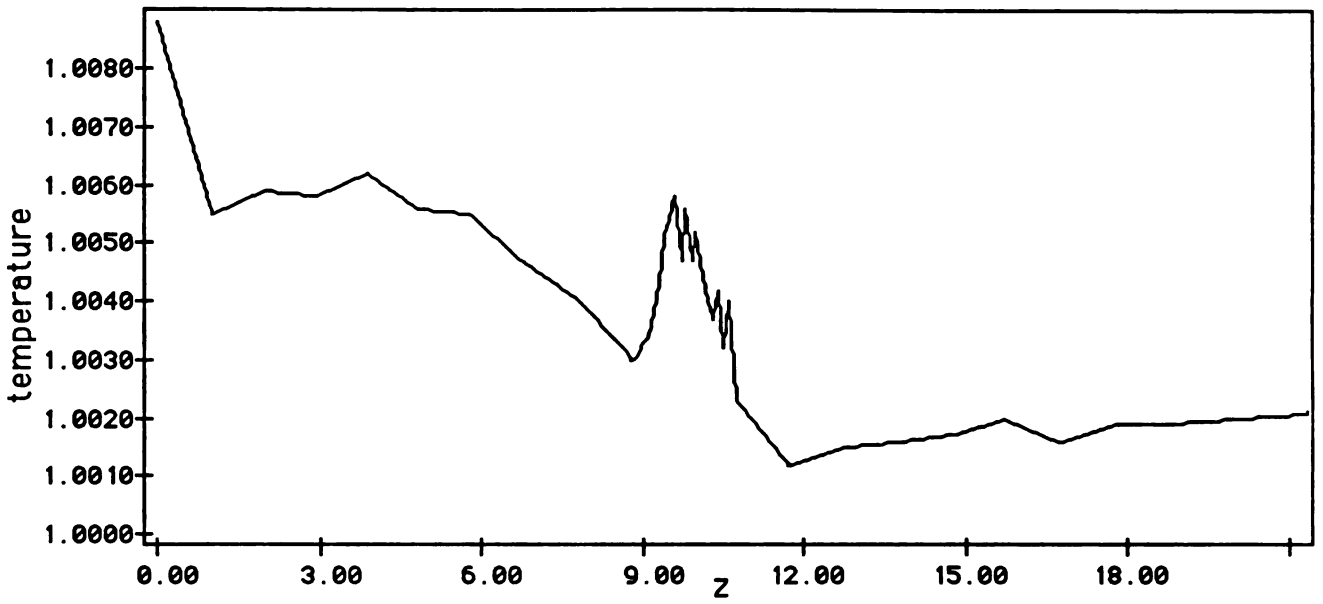


Figure 4.7: Temperature versus z for the central x zone. T=200.

The resonant absorption region can be seen even more clearly in figure 4.8 which shows the temperature average over a cycle versus z. The region begins at or just before z_o and continues to z_t. I ran shorter simulations with half the cell size and compared the results. The temperature rise was similar to that observed for larger cells and the region was same size; therefore, I concluded that smaller cell size would not improve the resolution of this resonant heating zone.

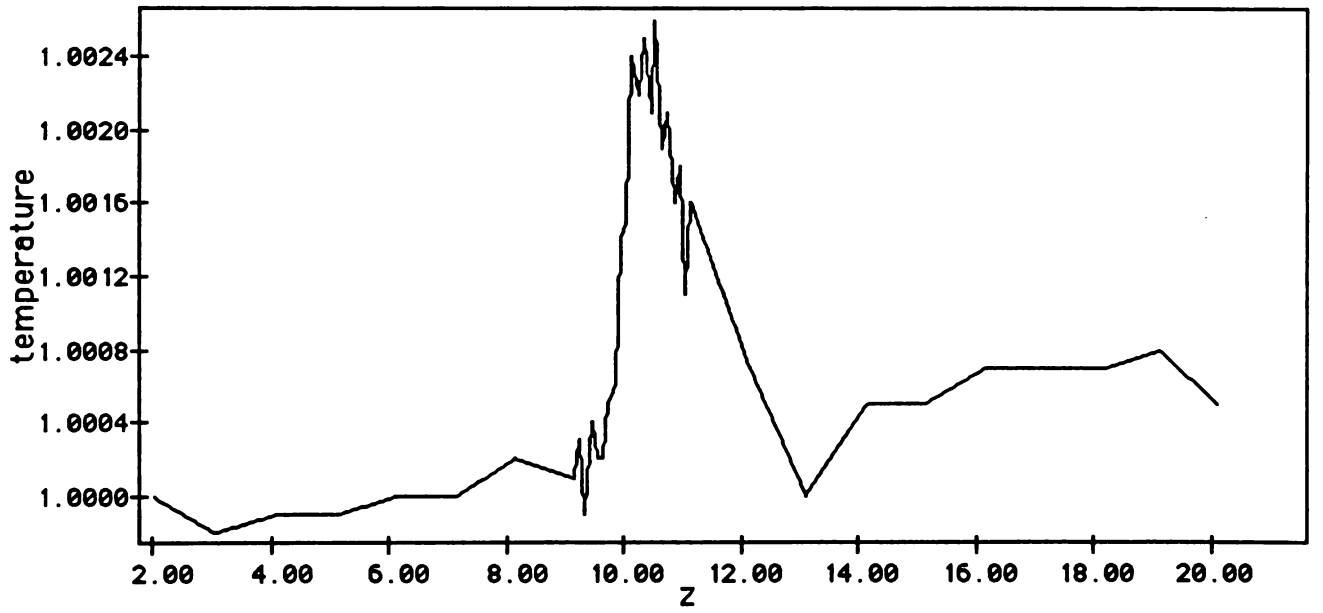


Figure 4.8: Temperature averaged over one cycle versus z. T=200.

I calculated the amount of surplus thermal energy in the resonant zone to be (8.9±1.9)% of energy available from the wave. Since the region remained in pressure equilibrium, the magnetic field, and thus the energy stored in the field, decreased. Considering both thermal energy gain and magnetic energy loss, I found the resonant zone had absorbed (3.0±0.6)% of the energy available from the wave. An additional (4.9±0.4)% of the energy was transmitted. I could not directly calculate the fraction of the energy reflected by the interface; however, the standing wave pattern was consistent with earlier simulations having reflections greater than 90%.

5
Conclusion and Discussion

5.1 General Conclusions

The results of the simulations run in chapter 3 and 4 allow me to make several conclusions regarding the general approach used. First and foremost, the program was able to model all of the physical problems attempted. When accepted values were known, the computational results agreed within the precision requested. The only exception was in modeling contact discontinuities. I believe this shortcoming could have been overcome by introducing non-zero thermal conductivity or another smoothing mechanism. This was not done, however, because contact discontinuities did not develop in the physical problems.

Several features of the program make it suitable for a wide variety of problems. First, it is an explicit code. Since no linearization or iterative calculations are required, it is straightforward to include any forces or other extra terms required for a particular problem. I also believe that explicit methods are easier to understand and implement.

Second, the dynamic grid allow the program to concentrate on the region of interest. Small zones, such as the resonant heating zone in section 4.3, could be studied without increasing the number of grid points beyond reason. Furthermore, fast moving features, such as shocks, could be followed without increasing the numerical domain. As another benefit, driving waves from the boundary was straightforward using the moving grid. For many problems, it would be beneficial to have the grid dynamic in two or three dimensions; however, this introduces the complication of grid twisting. More complicated algorithms for managing the grid motion may be able to overcome this difficulty. I used only the simplest grid motions.

Last, the Runge Kutta like method I used for the time integration proved to be very stable. It allowed the program to use the largest possible steps consistent with the requested precision with a minimum of overhead. It is straightforward to understand and implement. The method was clearly superior to using fixed time steps, either user chosen or calculated, or to using the Courant condition. Using the Courant condition becomes complicated with the addition of viscosity, conductivity, and other dispersive effects. Once implemented, the method was insensitive to other changes in the program. I would strongly recommend this method to others.

A number of the aspects of the program, although satisfactory, leave room for change and improvement. Foremost among these is the choice of numerical boundary conditions. From my work with this program, I would have to conclude that finding appropriate, stable, and easily implemented boundary conditions is the most difficult aspect of simulating magnetohydrodynamics. The method of following a grid point at the local wave velocity worked well for simulations of waves and other smooth variations but was not perfect for shocks and other discontinuities. Problem specific numerical boundary conditions may be required for these types of flow.

The use of a spectral method for taking spatial derivatives worked well for the problems on which I concentrated; however, there is some cost is using them. The accuracy of the derivatives obtained from the spectral method greatly exceeded that of the finite difference method for a given number of grid points, N. The finite difference method was faster, however, for a given N. This trade-off favored the spectral method only for $N \geq 32$. Thus for many problems with a small number of grid points, it would have been preferable to use finite difference methods in both directions. When computer power allows a larger grid or when using vector machines, the spectral method is clearly more powerful. For problems for which periodic boundary conditions are not appropriate spectral methods should not be used. The modular structure used in my program allowed me to easily substitute different routines for the spectral derivative subroutine.

Another aspect of this research which may require further work for future problems is the presentation and interpretation of the data. This type of program generates large volumes of data which can be difficult to interpret. The linegraph and contour plotting routines which I wrote were adequate for displaying the static characteristics of one and two dimensional problems; however, some form of animation would have been helpful in understanding some of the dynamic aspects of the problems. Additional display techniques will be required to adequately interpret three dimensional results. Both animation and three dimensional problems require more computing power than I had available.

In general, I would conclude that the algorithms, numerical methods, and the resulting program were successful in simulating the problems attempted within the constraints imposed by available computing power.

The conclusions which can be drawn from the simulation of non-linear waves

incident normal to a magnetic flux slab are quite straightforward. First the presence of the slab does not affect the final state of the fluid away from the slab. Second, the final state of the interior is different from that which would result if there were no magnetic structure. The details of the differences are given in section 4.1. Lastly, a source of shocks and rarefactions, such as the solar convection zone, would not concentrate the field.

Similar conclusions can be drawn for the simulation of non-linear waves incident normal to a flux tube. Once again, the magnetic structure effects the final state within the tube but not in the environment far from the slab. It was not as clear that the internal structure would be unaffected by repeated passes of shocks and rarefactions of equal amplitudes. A finer grid and more CPU time would be required to make a definite conclusion.

The conclusions which can be drawn from the last simulation are discussed in the next section.

5.2 Discussion of Acoustic Absorption by Sunspots

The recent observation of absorption of high-degree p-mode solar wave energy by sunspots (Braun et al, 1987) can be considered in light of the numerical results of section 4.3. The simulated problem can be interpreted as a simplified simulation of the interaction between solar p-mode acoustic waves and the sun spot boundary. From this simple approach, I should be able to address several questions. First, is the resonant absorption of wave energy of a sufficient magnitude to explain the observed results? If not, does a sufficient fraction of the incident wave energy pass through the interface into the sun spot interior where other mechanism could absorb or redirect the energy? Lastly, is the two dimensional magnetic structure of the sun spot simulation sufficient to explain the observed absorption or is three dimensional structure required?

First I must show that the dimensionless parameters chosen are appropriate for the physical problem. I used the following for the values of the sun spot (Allen,1973):

R_s=15Mm	radius of sunspot
B=2750 gauss	magnetic field
$P=8 \times 10^4$ dyn/cm^2	gas pressure
c=8 km/sec	speed of sound

The units were chosen to be consistent with those used by Braun et al. The choices for R_s, P, and c yielded the following scaling factors (see section 1.5):

$M=2 \times 10^{-7}$ grams/cm^3	density scale factor
L=0.1 Mm	length scale factor
T=12.5 sec	time scale factor

Using these scaling values and the parameters used in the simulation, I get the following for the physical parameters:

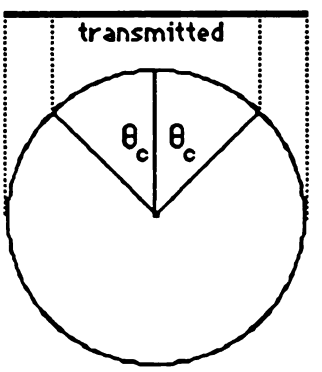
parameter	scaled	physical
λ	25	2.5 Mm
T_{max}	200	2500 sec
ω	0.25	0.02 rad/sec
B	7.5	2730 gauss
δ interface thickness	2	0.2 Mm
ρ	1	2×10^{-7} g/cm^3

These values are consistent with the sun spot values and the frequencies used by Braun et al. Note that for the radius and length scaling factor used, the plane

interface should have been an arc of about 10°. An additional simplification is the uniform temperature. The above scaling factors give a physical temperature of about 5500K while the actual temperature varies from 4240K on the interior of the sunspot to 6050K in the surrounding photosphere.

Addressing the first question above, I can consider the energy absorption by the resonant layer in the interface. Braun et al observed up to 50% absorption. Since the amount of energy absorbed for a simulation at any angle never exceeded 5% and was ~0% for angles less than 25°, I can conclude that the resonant absorption can not be the primary mechanism for the observed energy absorption.

To answer the second question, consider the following figure.



Incident wave

$$f = \frac{2R\sin\theta_c}{2R}\theta_c = \sin\theta_c$$

Figure 5.1: Wave incident on circular sunspot.

The critical angle, θ_c, is the angle at which the majority of the wave energy is reflected. If $\theta_c=(25°\pm4°)$ is used as the results of the simulation suggest, the fraction, f, of the incident energy which passes into the interior is (42±5)%. This result is consistent with observation if a mechanism can be found inside the sunspot which can absorb nearly all of the acoustic energy which enters.

Lastly, can any two dimensional magnetic structure account for the observed absorption? The most promising model for such a sunspot consists of a large number of narrow flux tubes. Incident acoustic waves would be partially transmitted and partially reflected repeatedly as they pass through this bundle of tubes. Since the results of the simulations (and observations) suggest that the absorption rate is largely independent of scaling effects, the results of the simulations may be applicable to each of these tubes. Using $\theta_c=25°$ and an absorption rate of 3% for the reflected part of the wave, one gets an average absorption of 1-2% for each tube

interaction. The observed absorption would require an average of 40-50 interactions. This value is not completely unreasonable. Since the critical region occurs where the magnetic field is relatively large, the magnetic field need not vanish between the narrow flux tubes. Three dimensional structures are even more promising since all is required is a mechanism for deflecting the energy downward.

In summary, I would conclude that the simulation shows that resonant absorption by a single two dimensional magnetic structure cannot account for the observed absorption. However, the numerical results are consistent with a mechanism inside the sunspot which absorbs or deflects the energy downward. Moreover, the results are consistent with absorption by a two dimension fine magnetic structure in which the magnetic field varies about a relatively large value.

5.3 Future Problems

There are a very large number of interesting problems which can be simulated by this program, by its three dimensional sibling, and by programs using similar algorithms written for different geometries. Most of these require more computing power than I currently have available; however, some of them do not require a supercomputer.

The current program should be used to simulate the normal incidence of non-linear waves on flux tubes with a much finer grid than was used here. This should resolve the question whether a source of small shocks and rarefactions can concentrate the field in a tube. The interaction of acoustic waves and sunspots could be further studied using the same circular flux tube. My experiences in section 4.3 suggest that a grid at least 10 times as fine as that used in section 4.2 would be required to study the absorption zone. The two dimensional code could also be used to study surface waves and solitons along flux slabs. I was able to take a preliminary look at these problems using the Vax but a detailed analysis required more grid points and CPU time than was practical.

A completely three dimensional Cartesian program would allow for the study of flux tube models of sunspots. By placing the z direction along the symmetry axis of the tube, gravity could be included. This would allow for the modeling of tapering flux tubes, sausage and kink mode tube waves, and a large number of similar problems. Even using a supercomputer, however, the total number of grid points would have to be kept fairly small. Thus problems such as the oblique incidence of acoustic waves on flux tubes could not be done with the resolution necessary to study small resonance zones.

Perhaps the greatest promise for simulating these flux tube problems lies in rewriting the program in cylindrical coordinates. Small cells could be used in the interface while larger cells could be used on both the inside and outside. The two dimensional program could be used to simulate normal incidence. This problem may be within the grasp of the Vax and will probably be the next problem I consider unless a larger computer becomes available. A three dimensional program would put the modeling of oblique incidence of acoustic waves on flux tubes within the grasp of supercomputers. Simulations of non-symmetric tube waves would also be

possible.

A two or three dimensional spherical algorithm would be possible using methods similar to those used in my current program. I have no plans at present to attempt this since no currently accessible astrophysical problems lend themselves to this symmetry. When sufficient computing power becomes available it may be possible to model proto-stellar collapse including magnetic fields. A Poisson solving algorithm would be required to handle the self-gravity. I include this aside because it was interest in this problem which originally started me down this line of research.

Appendix A
Glossary of Symbols

B— magnetic flux density
c— speed of light
D— electric flux density
E— electric field
F— total force
f— body force per unit volume
g— gravitational field
H— magnetic field
J— current density
P— pressure
R— array containing all eight fluid variables
q— electric charge
t— time
v— fluid velocity
α—ratio of densities
β—ratio of gas pressure to magnetic pressure
γ—ratio of specific heats
ε—permitivity
ζ—viscosity
η— magnetic diffusivity
μ—permeability
ρ— mass density
ρ_e— charge density
σ— electric conductivity
τ—viscous stress tensor

Appendix B
Summary of Equations

In chapter 2, I have developed the following equations which describe the time evolution of a magnetohydrodynamic fluid. The variables are defined in the laboratory frame, are dimensionless, and have the stratification divided out. Each equation is written below in vector form and summation form. The equations are presented in complete detail on the pages 104-105.

1. The continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0 \qquad (1.8)$$

becomes

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = \frac{\rho v_z}{H}$$

with the addition of stratification. In summation form

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x_j} \rho v_j = \frac{\rho v_z}{H}$$

2. The momentum equations

$$\frac{\partial}{\partial t} \rho \mathbf{v} + \nabla \cdot \left[\rho \mathbf{v}\mathbf{v} + \left(P + \frac{|\mathbf{B}|^2}{2\mu_o} \right) \underline{1} - \underline{\tau} \right] = \rho \mathbf{g} + \frac{(\mathbf{B} \cdot \nabla)\mathbf{B}}{\mu_o} \qquad (1.14)$$

becomes

$$\frac{\partial}{\partial t} \rho \mathbf{v} + \nabla \cdot \left[\rho \mathbf{v}\mathbf{v} + \left(P + \frac{|\mathbf{B}|^2}{2\mu_o} \right) \underline{1} - \underline{\tau} \right] = \rho \mathbf{g} + \frac{(\mathbf{B} \cdot \nabla)\mathbf{B}}{\mu_o} + \frac{\rho \mathbf{v} v_z}{H}$$

In summation form

$$\frac{\partial}{\partial t} \rho v_i + \frac{\partial}{\partial x_j} \left[\rho v_i v_j + \left(P + \frac{B_k B_k}{2\mu_o} \right) \delta_{ij} - \tau_{ij} \right] = \rho g_i + \left(B_k \frac{\partial}{\partial x_k} \right) \frac{B_i}{\mu_o} + \frac{\rho v_i v_z}{H}$$

3. The internal energy equations

$$\frac{\partial}{\partial t} \rho e + \nabla \cdot \left[\mathbf{v} (\rho e + P) \right] = \mathbf{v} \cdot \nabla P + \nabla \cdot k_T \nabla T + \tau_{ij} \frac{\partial v_i}{\partial x_j} + \frac{\eta}{\mu_o} |\nabla \times \mathbf{B}|^2 \qquad (1.17b)$$

becomes

$$\frac{\partial}{\partial t} \rho e + \nabla \cdot \left[\mathbf{v} (\rho e + P) \right] = \mathbf{v} \cdot \nabla P + \nabla \cdot k_T \nabla T + \tau_{ij} \frac{\partial v_i}{\partial x_j} + \frac{\eta}{\mu_o} |\nabla \times \mathbf{B}|^2 + \frac{\rho e v_z}{H}$$

In summation form

$$\frac{\partial}{\partial t}\rho e+\frac{\partial}{\partial x_j}[v_j(\rho e+p)]=v_j\frac{\partial P}{\partial x_j}+\frac{\partial}{\partial x_j}\left(k_T\frac{\partial T}{\partial x_j}\right)+\tau_{ij}\frac{\partial v_i}{\partial x_j}+\frac{\eta}{\mu_o}|\nabla\times\mathbf{B}|^2+\frac{\rho e v_z}{H}$$

4. The magnetic equations

$$\frac{\partial \mathbf{B}}{\partial t}=\nabla\times(\mathbf{V}\times\mathbf{B})-\nabla\times\eta\nabla\times\mathbf{B} \qquad (1.7a)$$

$$\frac{\partial B_i}{\partial t}+\frac{\partial}{\partial x_j}\left[B_i v_j-B_j v_i+\eta\left(\frac{\partial B_j}{\partial x_i}-\frac{\partial B_i}{\partial x_j}\right)\right]=0 \qquad (1.7b)$$

becomes

$$\frac{\partial \mathbf{B}}{\partial t}=\nabla\times(\mathbf{V}\times\mathbf{B})-\nabla\times\eta\nabla\times\mathbf{B}+\frac{\mathbf{B}v_z}{H}$$

$$\frac{\partial B_i}{\partial t}+\frac{\partial}{\partial x_j}\left[B_i v_j-B_j v_i+\eta\left(\frac{\partial B_j}{\partial x_i}-\frac{\partial B_i}{\partial x_j}\right)\right]=\frac{B_i v_z}{H}$$

The continuity equation

$$\frac{\partial \rho}{\partial t}+\frac{\partial}{\partial x}\rho v_x+\frac{\partial}{\partial y}\rho v_y+\frac{\partial}{\partial z}\rho v_z=\frac{\rho v_z}{H}$$

The momentum equations

$$\frac{\partial}{\partial t}\rho v_x+\frac{\partial}{\partial x}\left[\rho v_x v_x+\left(P+\frac{B_x^2+B_y^2+B_z^2}{2\mu_o}\right)-\tau_{xx}\right]+\frac{\partial}{\partial y}[\rho v_x v_y-\tau_{xy}]+\frac{\partial}{\partial z}[\rho v_x v_z-\tau_{xz}]$$
$$=\rho g_x+\frac{1}{\mu_o}\left(B_x\frac{\partial B_x}{\partial x}\right)+\frac{1}{\mu_o}\left(B_y\frac{\partial B_x}{\partial y}\right)+\frac{1}{\mu_o}\left(B_z\frac{\partial B_x}{\partial z}\right)+\frac{\rho v_x v_z}{H}$$

$$\frac{\partial}{\partial t}\rho v_y+\frac{\partial}{\partial x}[\rho v_y v_x-\tau_{yx}]+\frac{\partial}{\partial y}\left[\rho v_y v_y+\left(P+\frac{B_x^2+B_y^2+B_z^2}{2\mu_o}\right)-\tau_{yy}\right]+\frac{\partial}{\partial z}[\rho v_y v_z-\tau_{zy}]$$
$$=\rho g_y+\frac{1}{\mu_o}\left(B_x\frac{\partial B_y}{\partial x}\right)+\frac{1}{\mu_o}\left(B_y\frac{\partial B_y}{\partial y}\right)+\frac{1}{\mu_o}\left(B_z\frac{\partial B_y}{\partial z}\right)+\frac{\rho v_y v_z}{H}$$

$$\frac{\partial}{\partial t}\rho v_z+\frac{\partial}{\partial x}[\rho v_z v_x-\tau_{xz}]+\frac{\partial}{\partial y}[\rho v_z v_y-\tau_{yy}]+\frac{\partial}{\partial z}\left[\rho v_z v_z+\left(P+\frac{B_x^2+B_y^2+B_z^2}{2\mu_o}\right)-\tau_{zy}\right]$$
$$=\rho g_z+\frac{1}{\mu_o}\left(B_x\frac{\partial B_z}{\partial x}\right)+\frac{1}{\mu_o}\left(B_y\frac{\partial B_z}{\partial y}\right)+\frac{1}{\mu_o}\left(B_z\frac{\partial B_z}{\partial z}\right)+\frac{\rho v_z v_z}{H}$$

The energy equation

$$\frac{\partial}{\partial t}\rho e+\frac{\partial}{\partial x}[v_x(\rho e+p)]+\frac{\partial}{\partial y}[v_y(\rho e+p)]+\frac{\partial}{\partial z}[v_z(\rho e+p)]$$

$$=v_x\frac{\partial P}{\partial x}+v_y\frac{\partial P}{\partial y}+v_z\frac{\partial P}{\partial z}+\frac{\partial}{\partial x}\left(k_T\frac{\partial T}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_T\frac{\partial T}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_T\frac{\partial T}{\partial z}\right)+$$

$$\tau_{xx}\frac{\partial v_x}{\partial x}+\tau_{xy}\frac{\partial v_x}{\partial y}+\tau_{xz}\frac{\partial v_x}{\partial z}+\tau_{yx}\frac{\partial v_y}{\partial x}+\tau_{yy}\frac{\partial v_y}{\partial y}+\tau_{yz}\frac{\partial v_y}{\partial z}+\tau_{zx}\frac{\partial v_z}{\partial x}$$

$$+\tau_{zy}\frac{\partial v_z}{\partial y}+\tau_{zz}\frac{\partial v_z}{\partial z}+\frac{\eta}{\mu_o}\left[\left(\frac{\partial B_x}{\partial y}-\frac{\partial B_y}{\partial x}\right)^2+\left(\frac{\partial B_z}{\partial x}-\frac{\partial B_x}{\partial z}\right)^2+\left(\frac{\partial B_y}{\partial z}-\frac{\partial B_z}{\partial y}\right)^2\right]+\frac{\rho e v_z}{H}$$

The magnetic equations

$$\frac{\partial B_x}{\partial t}+\frac{\partial}{\partial y}\left[B_x v_y - B_y v_x + \eta\left(\frac{\partial B_y}{\partial x}-\frac{\partial B_x}{\partial y}\right)\right]+\frac{\partial}{\partial z}\left[B_x v_z - B_z v_x + \eta\left(\frac{\partial B_z}{\partial x}-\frac{\partial B_x}{\partial z}\right)\right]=\frac{B_x v_z}{H}$$

$$\frac{\partial B_y}{\partial t}+\frac{\partial}{\partial x}\left[B_y v_x - B_x v_y + \eta\left(\frac{\partial B_x}{\partial y}-\frac{\partial B_y}{\partial x}\right)\right]+\frac{\partial}{\partial z}\left[B_y v_z - B_z v_y + \eta\left(\frac{\partial B_z}{\partial y}-\frac{\partial B_y}{\partial z}\right)\right]=\frac{B_y v_z}{H}$$

$$\frac{\partial B_z}{\partial t}+\frac{\partial}{\partial x}\left[B_z v_x - B_x v_z + \eta\left(\frac{\partial B_x}{\partial z}-\frac{\partial B_z}{\partial x}\right)\right]+\frac{\partial}{\partial y}\left[B_z v_y - B_y v_y + \eta\left(\frac{\partial B_y}{\partial z}-\frac{\partial B_z}{\partial y}\right)\right]=\frac{B_z v_z}{H}$$

Appendix C
Analytic Solutions to Test Problems

Some of the analytic solutions used in chapter 3 were developed by me specifically for this research. None of these represent completely original work since they are based on well known results or published works. These derivations are include in the following section. References are included as appropriate.

C.1 Piston Driven Shock

The solution of the piston driven shock is shown in figure 3.4. It consists of two fluid regions separated by a shock. The shock moves at a constant speed and has zero thickness. In the frame of reference moving with the shock, fluid moves into the shock from the right with a constant speed of u_2 and fluid leaves the shock toward the left with a constant speed of u_1. Since the problem is steady state, all time derivatives vanish in the equations. Thus the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0 \tag{1.8}$$

shows that ρu must be conserved across the shock. Thus

$$\rho_1 u_1 = \rho_2 u_2 \tag{c.1}$$

Similarly, the momentum equation

$$\frac{\partial}{\partial t}\rho \mathbf{v} + \nabla \cdot (\rho \mathbf{v}\mathbf{v} + P\underset{\sim}{1} - \underset{\sim}{\tau}) = \mathbf{f} \tag{1.10}$$

with the idealization that viscosity and external forces are zero yields

$$\rho_1 u_1^2 + P_1 = \rho_2 u_2^2 + P_2 \tag{c.2}$$

If the same technique and idealizations are applied to the total energy equation

$$\frac{\partial}{\partial t}\left(\tfrac{1}{2}\rho v^2 + \rho e\right) + \nabla \cdot \left[\mathbf{v}\left(\tfrac{1}{2}\rho v^2 + \rho e + P\right) - \mathbf{v}\cdot\underset{\sim}{\tau} - k_T \nabla T\right] = \mathbf{v}\cdot\mathbf{f} + \frac{\eta}{\mu_o}|\nabla \times \mathbf{B}|^2 \tag{1.20b}$$

one gets

$$u_1\left(\tfrac{1}{2}\rho_1 u_1^2 + \rho_1 e_1 + P_1\right) = u_2\left(\tfrac{1}{2}\rho_2 u_2^2 + \rho_2 e_2 + P_2\right) \tag{c.3}$$

Equation (c.1) can be used in equation (c.3) to yield

$$\tfrac{1}{2}u_1^2 + e_1 + \frac{P_1}{\rho_1} = \tfrac{1}{2}u_2^2 + e_2 + \frac{P_2}{\rho_2}$$

These three equations may now be transformed, nonrelativistically, back to the laboratory frame of reference—

$$\rho_1(v_1 - v_s) = \rho_2(v_2 - v_s)$$

$$P_1 + \rho_1(v_1 - v_s)^2 = P_2 + \rho_2(v_1 - v_2)^2$$

$$\tfrac{1}{2}(u_1 - v_s)^2 + e_1 + \frac{P_1}{\rho_1} = \tfrac{1}{2}(u_2 - v_s)^2 + e_2 + \frac{P_2}{\rho_2}$$

In region 1, behind the shock, one knows $v_1=v_p$. In region 2, ahead of the shock one knows $v_2=0$. The values of ρ_2 and e_2 are those of the undisturbed gas and are specified by the initial conditions. In addition to these three equations, one has the equations of state

$$P_1=(\gamma-1)\rho_1 e_1 \qquad P_2=(\gamma-1)\rho_2$$

Therefore P_1 and P_2 are known in terms of $\rho_1 e_1$ and $\rho_2 e_2$; thus we have three equations and three unknowns. Using the known values and eliminating the pressures, one gets

$$\rho_1(v_p-v_s)=-\rho_2 v_s \quad \Rightarrow \quad v_s=\frac{\rho_1 v_p}{\rho_1-\rho_2} \tag{c.4}$$

$$\rho_1(v_p-v_s)^2+(\gamma-1)\rho_1 e_1 = \rho_2 v_s^2+(\gamma-1)\rho_2 e_2$$

$$\frac{1}{2}(v_p-v_s)^2+\gamma e_1 = \frac{1}{2}v_s^2+\gamma e_2$$

We can use the first equation, (c.4), to eliminate v_s from the other two equations getting

$$(\gamma-1)\rho_1 e_1-v_p^2\frac{\rho_1\rho_2}{\rho_1-\rho_2}=(\gamma-1)\rho_2 e_2$$

and

$$\gamma e_1-\frac{v_p^2}{2}\left(\frac{\rho_1+\rho_2}{\rho_1-\rho_2}\right)=\gamma e_2 \quad \Rightarrow \quad e_1=e_2+\frac{v_p^2}{2\gamma}\left(\frac{\rho_1+\rho_2}{\rho_1-\rho_2}\right) \tag{c.5}$$

This can then be used to eliminate e_1. Thus

$$(\gamma-1)\rho_1 e_2+\frac{(\gamma-1)}{2\gamma}\left(\frac{\rho_1+\rho_2}{\rho_1-\rho_2}\right)\rho_1 v_p^2-v_p^2\frac{\rho_1\rho_2}{\rho_1-\rho_2}=(\gamma-1)\rho_2 e_2$$

Using the value of the speed of sound in the undisturbed fluid

$$c_2^2=\gamma(\gamma-1)e_2$$

one may solve this equation for the ratio of densities. Since the equation is quadratic, there are two solutions. Discarding the negative, non-physical, solution one gets

$$\alpha \equiv \frac{\rho_1}{\rho_2} = \frac{1+\frac{\gamma+1}{4}\frac{v_p^2}{c_2^2} + \sqrt{\frac{v_p^2}{c_2^2} + \left(\frac{\gamma+1}{4}\frac{v_p^2}{c_2^2}\right)^2}}{1+\frac{\gamma-1}{2}\frac{v_p^2}{c_2^2}}$$

The ratio of densities, which I have called α here, can then be plugged back into equations (c.4) and (c.5) to yield

$$\frac{e_1}{e_2} = \left(1 + \frac{\gamma-1}{2}\frac{v_p^2}{c_2^2}\right)\frac{\alpha+1}{\alpha-1}$$

$$v_s = \frac{\alpha}{\alpha-1}v_p$$

C.2 Weak Shock Profile

The profile of a weak shock may be derived using a method given by Landau and Lifshitz. They determined the variation of pressure within a shock (Landau & Lifshitz, 1959, §87) as

$$\frac{1}{2}\left(\frac{\partial^2 V}{\partial p^2}\right)_s (p-p_1)(p-p_2) = -\frac{V^3}{c^3}\left\{\left(\frac{4}{3}\eta+\zeta\right)+\frac{\kappa}{T}\left(\frac{\partial T}{\partial p}\right)_s\left(\frac{\partial V}{\partial s}\right)_p c^2\rho^2\right\}\frac{dp}{dx}$$

where $V\equiv 1/\rho$, $h\equiv\xi_1 = 0$, $\zeta\equiv\xi_2$, and $\kappa\equiv K_T = 0$ in the symbols I am using. Substituting in these symbols and values yields

$$\frac{1}{2}\left(\frac{\partial^2 V}{\partial p^2}\right)_s (p-p_1)(p-p_2) = -\frac{V^3}{c^3}\xi_2\frac{dp}{dx}$$

One can use pV^γ = constant for an adiabatic process to evaluate the derivative on the right side of the equation:

$$\left(\frac{\partial^2 V}{\partial p^2}\right)_s = \frac{\gamma+1}{\gamma^2}\frac{V}{p^2}$$

Using this and $c^2 = \gamma pV$, the equation simplifies to

$$1 = \frac{-2c\xi_2}{\gamma+1}\frac{1}{(p-p_1)(p-p_2)}\frac{dp}{dx}$$

Integrating

$$\int_{-x}^{+x} dx = \int_{-x}^{+x}\frac{-2c\xi_2}{\gamma+1}\frac{1}{(p-p_1)(p-p_2)}\frac{dp}{dx}dx = \int_{p_1}^{p_2}\frac{-2c\xi_2}{\gamma+1}\frac{1}{(p-p_1)(p-p_2)}dp$$

yields

$$2x = \frac{c\xi_2}{\gamma+1}\frac{4}{p_2-p_1}\tanh^{-1}\left(\frac{p-\frac{1}{2}(p_2+p_1)}{\frac{1}{2}(p_2-p_1)}\right)$$

Therefore, one gets the following variation of pressure

$$p = \frac{1}{2}(p_2+p_1) + \frac{1}{2}(p_2-p_1)\tanh\left(\frac{x}{\delta}\right)$$

with

$$\delta = \frac{2c\xi_2}{\gamma+1}\frac{1}{p_2-p_1} = \frac{2c\xi_2}{\gamma^2-1}\frac{1}{p_2e_2-p_1e_1}$$

For a shock with this profile, over 90% of the pressure variation occurs with in a distance of δ.

Appendix D
Auxiliary Programs

I have written a number of programs to graph, analyze, or calculate data. In this section I have included the source codes for these. The FORTRAN routines were compiled and included in an object library. The Pascal programs were written on and for the Macintosh computer.

D.1 Graph6

```
program linegraph(input,output);
{This program uses the linegraph routines to read output from GMHD and draw a
graph for each of six variables.  It has options of connecting the points with
line segments or using a cubic spline.  The spline is slow and not needed for
n≥ 40.}

Uses MacIntf;
{This program uses the Macintosh interface}

type{global types}
   parray= array[0..500] of integer;
   glnarray= array[0..256] of real;

var{global variables}
   thefile:              text;
   n,i,ni:               integer;
   z,rho,vx,vz,bx,bz,t: glnarray;
   quit,done,
   splineflag,
   tickflag,
   pointflag:         Boolean;
   theWindow:         WindowPtr;
   SKIP:          CHAR;
   data:          array [1..80] of char;

procedure getn;{reads number of points from file}
   begin
      readln(thefile,n);
      writeln('Found table with ',n,' entries.');
   ni:=1;
   while ((not eoln(theFile)) and (ni < 80)) do begin
      read(theFile, data[ni]);
      ni:=ni+1
      end;{while}
   readln(thefile);
   end; {getn}

procedure getxy;{reads the points from file}
var
   i,q:  integer;
   zz,rr,vvx,vvz,tt,bbx,bbz:  real;
   begin
   readln(theFile);
       for i:=0 to n do begin
         readln(thefile,q,zz,rr,vvx,vvz,tt,bbx,bbz);
         z[i]:=zz;
         rho[i]:=rr;
         vx[i]:=vvx;
         vz[i]:=vvz;
         t[i]:=tt;
         bx[i]:=bbx;
         bz[i]:=bbz;
         end; {for}
   end;{getxy}


procedure graph(  x,y:     glnarray;
```

```
                n:       integer;
                tflag,
                pflag:   boolean);{the actual graphing program}

const
   hmin=60;
   hmax=505;
   numticks=8;
   vmin=25;
   vmax=225;
   grit=3;
type
   harray= array[hmin..hmax] of real;

var
   h,vo:        integer;
   ypl,ypn:     real;
   xmax,xmin:   real;
   ymax,ymin:   real;
   y2:          glnarray;
   xh,v:        harray;
   xp,yp:       parray;
   graphrect:   rect;
   log10:       real;
   q:           char;


procedure spline(ypl,ypn:  real;
           var  y2: glnarray);{determines the cubic spline}
{This procedure is based on Press et al.,1986}

var   i,k:           integer;
      p,qn,sig,un:   real;
      u:             glnarray;

begin {spline}
   if(ypl>0.99e30) then begin
      y2[1]:=0.0;
      u[1]:=0.0
      end {then}
   else begin
      y[2]:=-0.5;
      u[1]:=(3.0/(x[2]-x[1]))*((y[2]-y[1])/(x[2]-x[1])-ypl)
      end; {else}

   for i:= 2 to n-1 do begin
      sig:=(x[i]-x[i-1])/(x[i+1]-x[i-1]);
      p :=sig*y2[i-1]+2.0;
      y2[i]:=(sig-1.0)/p;
      u[i]:=(y[i+1]-y[i])/(x[i+1]-x[i])-(y[i]-y[i-1])/(x[i+1]-x[i]);
      u[i]:=(6.0*u[i]/(x[i+1]-x[i-1])-sig*u[i-1])/p
      end;{for}
   if (ypn>0.99e30) then begin
      qn := 0.0;
      un := 0.0
      end {then}
   else begin
      qn:= 0.5;
      un:= (3.0/(x[n]-x[n-1]))*(ypn-(y[n]-y[n-1])/(x[n]-x[n-1]))
      end;{else}
```

```
   y2[n]:= (un-qn*u[n-1])/(qn*y2[n-1]+1.0);
   for k:=n-1 downto 1 do begin
      y2[k]:= y2[k]*y2[k+1]+u[k]
      end;{for}
   end;{spline}

procedure splint( xa,ya,y2a:  glnarray;
                  n:          integer;
                  x:          real;
            var   y:          real);
var
   klo,khi,k:  integer;
   a,b,h:      real;

begin {splint}
   klo:=1;
   khi:=n;

   while (khi-klo>1) do begin
      k:=(khi+klo) div 2;
      if(xa[k]>x) then khi:=k else klo:=k
   end; {while}

   h:=xa[khi]-xa[klo];
   if (h = 0.0) then begin
      writeln(' Bad xa input');
      readln
      end; {if}

   a:=(xa[khi]-x)/h;
   b:=(x-xa[klo])/h;
   y:=a*ya[klo]+b*ya[khi]+((a*a*a-a)*y2a[klo]+(b*b*b-b)*y2a[khi])*(h*h)/6.0
end;{splint}

procedure splint2(   xa,ya,y2a:  glnarray;
                  n:          integer;
                  x:          real;
            var   y:          real;
            var   klo,khi:    integer);
var
   a,b,h:      real;

begin {splint2}

   while (x>xa[khi]) do begin
      khi:=khi+1;
      klo:=klo+1
   end; {while}

   h:=xa[khi]-xa[klo];
   a:=(xa[khi]-x)/h;
   b:=(x-xa[klo])/h;
   y:=a*ya[klo]+b*ya[khi]+((a*a*a-a)*y2a[klo]+(b*b*b-b)*y2a[khi])*(h*h)/6.0
end;{splint2}

procedure scale(  x:       glnarray;
                  n:       integer;
            var   max,min: real);{This procedure scales the data to fit}
var
```

```
   i: integer;

begin{scale}
   max:=x[0];
   min:=x[0];
   for i:=1 to n do begin
      if (x[i] > max) then max:=x[i];
      if (x[i] < min) then min:=x[i];
      end;{for}
 if (abs(max-min)<0.0005) then max:=min+1.0
end;{scale}

procedure calcxhspline;

var
   delta:   real;
   h,i:        integer;

begin{calcxhspline}
   delta:=(xmax-xmin)/(hmax-hmin);
   for h:=hmin to hmax do xh[h]:=xmin+delta*(h-hmin);
   for i:=1 to n do xp[i]:=round(hmin+(x[i]-xmin)/delta);
end;{calcxhspline}

procedure calcxpyp;
var
   deltax,
   deltay:     real;
   i:       integer;

begin{calcxpvp}
   deltax:=(xmax-xmin)/(hmax-hmin);
   deltay:=(vmax-vmin)/(ymax-ymin);
   for i:=1 to n do begin
      xp[i]:=round(hmin+(x[i]-xmin)/deltax);
      yp[i]:=round(vmax -deltay*(y[i]-ymin))
   end;{for}
end;{calcxpvp}

procedure calcvspline(y2:     glnarray);
var
   delta,yh:   real;
   h,i,klo,khi:            integer;

begin{calcvspline}
   khi:=2;
   klo:=1;
   delta:=(vmax-vmin)/(ymax-ymin);
   h:=hmin;
   while h<= hmax do begin
      splint2(x,y,y2,n,xh[h],yh,klo,khi);
      v[h]:=vmax -delta*(yh-ymin);
      h:=h+grit
      end;{while}
   khi:=2;
   klo:=1;
   for i:= 1 to n do begin
      splint2(x,y,y2,n,x[i],yh,klo,khi);
      yp[i]:=round(vmax -delta*(yh-ymin))
      end;{for}
```

```
   end;{calcvspline}

Procedure addline;
var
   h: integer;
begin
   pensize(1,1);
   penmode(PatOr);
   vo:=round(v[hmin]);
   moveto(hmin,vo);
   h:=hmin;
   while h<=hmax do begin
      lineto(h,round(v[h]));
      h:=h+grit
      end;{while}
   moveto(0,0);
   penNormal
   end;{addline}

Procedure connect;
var
   i: integer;

begin
   pensize(1,1);
   penmode(PatOr);
   moveto(xp[1],yp[1]);
   for i:=2 to n do lineto(xp[i],yp[i]);
   moveto(0,0);
   PenNormal
end;{connect}

function rounder(x:  real): real;
var
   d:       integer;
   q,
   tens:    real;
begin
   d:=trunc(ln(x)/log10);
   repeat
      tens:=exp(d*log10);
      q:=round(x/tens)*tens;
      d:=d-1;
   until (q <> 0.0);
   rounder:=q

end;{rounder}

procedure tickmark;{calculate and place the tickmarks}
var
   xinc,yinc,xtick,ytick:     real;
   deltay,deltax:             real;
   i,top,bot:                 integer;
   vtick,htick:               integer;
   w,d:                       integer;

begin
   deltay:=(vmax-vmin)/(ymax-ymin);
   deltax:=(xmax-xmin)/(hmax-hmin);
   xinc:=(xmax-xmin)/numticks;
```

```
    yinc:=(ymax-ymin)/numticks;
    xinc:=rounder(xinc);
    if xinc = 0.0 then xinc:=0.1;
    yinc:=rounder(yinc);
    if yinc = 0.0 then yinc := 0.1;
    top:=trunc(ymax/yinc);
    bot:=trunc(ymin/yinc+0.999);
    w:=7; d:=2;
    if yinc<0.01 then d:=4;
    for i:=bot to top do begin
       ytick:=i*yinc;
       vtick:=round(vmax -deltay*(ytick-ymin));
       moveto(hmin-8,vtick);
       lineto(hmin-2,vtick);
       moveto(10,vtick+4);
       writeln(ytick:w:d)
    end;{for}

    top:=trunc(xmax/xinc);
    bot:=trunc(xmin/xinc+0.999);
    w:=7; d:=2;
    if xinc<0.01 then d:=4;
    for i:=bot to top do begin
       xtick:=i*xinc;
       htick:=round(hmin+(xtick-xmin)/deltax);
       moveto(htick,vmax+8);
       lineto(htick,vmax+2);
       moveto(htick-25,vmax+20);
       writeln(xtick:w:d)
    end;{for}

    moveto(0,0)

end; {tickmark}

procedure addpoints;{add the points as triangles}
var
    i: integer;
begin
    for i:= 1 to n do begin
       moveto(xp[i],yp[i]);
       lineto(xp[i],yp[i]-2);
       lineto(xp[i]-2,yp[i]+2);
       lineto(xp[i]+2,yp[i]+2);
       lineto(xp[i],yp[i]-2)
       end;{for}
    moveto(0,0)
end;{addpoints}

begin {graph}
 moveto(0,10);
 for i:=1 to ni do write( data[i]);
 writeln;
    log10:=ln(10.0);
    scale(x,n,xmax,xmin);
    scale(y,n,ymax,ymin);
    SetRect(graphrect,hmin-5,vmin-5,hmax+3,vmax+5);
    if splineflag then begin
       spline(1e50,1e50,y2);
       calcxhspline;
```

```
      calcvspline(y2);
      addline
      end{then}
   else begin
      calcxpyp;
      connect;
      end;{else}
   FrameRect(graphrect);
   if tflag then tickmark;
   if pflag then addpoints;
   readln(q);
   if q='.' then quit:=true
   end; {graph}

procedure findnext;{finds the next table to graph}
var
   one,s,t,p:  char;
begin
   repeat
      readln(theFile,one,s,t,p);
      if eof(theFile) then begin
         quit:=true;
         exit(findnext);
      end;{if eof}
      if one ='@' then begin
         if s='F' then splineflag:=false else splineflag:=true;
         if t='F' then tickflag:=false else tickflag:=true;
         if p='T' then pointflag:=true else pointflag:=false;
         end;{then}
   until (one = '@') or quit
end;{findnext}

Procedure OpenFile;{opens a new file or quits if cancel button}
var
   reply:      SFreply;
   typeList:   SFTypeList;
   where:      point;
begin
   typeList[0]:='TEXT';
   where.h:=80;
   where.v:=100;
   initCursor;
   SFGetFile(where,'',NIL,1,TypeList,NIL,reply);
   done:= not reply.good;
   if reply.good
      then
         open(theFile,reply.fName);
end;{OpenFile}

begin{main}
   theWindow:=FrontWindow;setWTitle(theWindow,'Linegraph');
 MoveWindow(theWindow,1,35,true);
 SizeWindow(theWindow,510,348,true);
   done :=false;
   openFile;
   while not done do begin
      quit:=false;
      tickflag:=true;
      pointflag:=false;
      repeat
```

```
      page(output);
      setWTitle(theWindow,'Searching File');
      findnext;
      if not quit then begin
         getn;
   if (n>256)then begin
      writeln('n larger than 256, using n=256');
      n:=256
      end;{if n>256}
         writeln('reading data from file');
         getxy;
         page(output);
         HideCursor;
         setWTitle(theWindow,'Density vs Z');
         graph(z,rho,n,tickflag,pointflag);
         moveto(0,0);
         page(output);
         setWTitle(theWindow,'Vx vs Z');
         graph(z,vx,n,tickflag,pointflag);
         moveto(0,0);
         page(output);
         setWTitle(theWindow,'Vz vs Z');
         graph(z,vz,n,tickflag,pointflag);
         moveto(0,0);
         page(output);
         setWTitle(theWindow,'Temperature vs Z');
         graph(z,t,n,tickflag,pointflag);
         moveto(0,0);
         page(output);
         setWTitle(theWindow,'Bx vs Z');
         graph(z,bx,n,tickflag,pointflag);
         moveto(0,0);
         page(output);
         setWTitle(theWindow,'Bz vs Z');
         graph(z,bz,n,tickflag,pointflag);
         moveto(0,0);
      end;{if not quit}
   until quit;
   close(thefile);
   openFile;
end;{while not done}
end.{main}
```

D.2 Contour

```
PROGRAM Contour;
{This program produces contour plots on the Macintosh using encoded data
send by subroutine MAKCON (in FORTRAN) from the mainframe.  It supports the
full Macintosh interface!}
   CONST {menu stuff}
      AppleID = 1000;
      FileID = 1001;
      EditID = 1002;
      nextID = 1003;

   VAR
      done, doneWithFile, next : Boolean;
      theEvt : eventRecord;
      wRecord : windowRecord;
      myWindow, TheWindow : windowPtr;
      WindowRect, blankRect : Rect;
      appleMenu, fileMenu, editMenu, nextMenu : MenuHandle;
      thefile : text;
   PROCEDURE ProcessMenu_in (CodeWord : longint);
   forward;

   PROCEDURE plot;
      CONST
         hmin = 5;
         vmin = 4;
         space = 0;
         skip = 65;
         nextcol = 66;
         quit = 67;
      VAR
         c, t : char;
         d, h, i, ni, ii : integer;
         thru : boolean;
         plotrect : rect;
   BEGIN
      thru := false;
      moveto(hmin, vmin);
      h := hmin;
      REPEAT
         BEGIN
            WHILE ((NOT thru) AND (NOT eoln(theFile))) DO
               BEGIN
                  read(theFile, c);
                  d := ord(c) - 40;
                  CASE d OF
                     space :
                        ;
                     skip :
                        move(0, 64);
                     nextcol :
                        BEGIN
                           h := h + 1;
                           moveto(h, vmin)
                        END;{nextcol}
                     quit :
                        BEGIN
                           moveto(300, 10);
```

```
                        ni := 1;
                        readln(theFile);
                        FOR ii := 1 TO 4 DO
                           BEGIN
                              WHILE ((NOT eoln(theFile)) AND (ni < 35)) DO
                                 BEGIN
                                    read(theFile, t);
                                    DrawChar(t);
                                    ni := ni + 1
                                 END;{while}
                              ni := 1;
                              readln(theFile);
                              moveto(300, 10 + 12 * ii);
                           END;{for}
                        thru := true;
                     END;
                  OTHERWISE
                     BEGIN   {all valid moves}
                        move(0, d);
                        line(0, 0);
                     END;{otherwise}
               END;{case d}
            END;{while not eoln}
         readln(thefile);
      END;{until}
   UNTIL (thru);
   SetRect(plotRect, hmin - 3, vmin - 3, h + 3, vmin + 293);
   FrameRect(plotRect);
END;{plot}

PROCEDURE miniloop;
   VAR
      WindowPointedTo : WindowPtr;
      MouseLoc : Point;
      WindoLoc : integer;
BEGIN
   HiliteMenu(0);
   REPEAT
      IF GetNextEvent(everyEvent, theEvt) THEN
         CASE theEvt.what OF
            mousedown :
               BEGIN
                  MouseLoc := theEvt.Where;
                  WindoLoc := FindWindow(MouseLoc, WindowPointedTo);
                  CASE WindoLoc OF
                     inMenuBar :
                        ProcessMenu_in(MenuSelect(MouseLoc));
                     inSysWindow :
                        SystemClick(theEvt, WindowPointedTo);
                     OTHERWISE
                        ;
                  END;{case theEvt.what}
               END;{mousedown}
            mouseup :
               ;
            updateEvt, activateEvt :
               ;
            OTHERWISE
               ;
```

```
            END;{case what event}
{and of then}
      UNTIL done OR doneWithFile OR next;
   END;{miniloop}


   PROCEDURE findnext;
      VAR
         one : char;
   BEGIN
      REPEAT
         IF eof(theFile) THEN
            DoneWithFile := true
         ELSE
            readln(theFile, one);
      UNTIL (one = '~') OR DoneWithFile
   END;{findnext}

   PROCEDURE openplot;
      VAR
         reply : SFreply;
         typeList : SFTypeList;
         where : point;
   BEGIN
      EnableItem(nextMenu, 0);
      DisableItem(fileMenu, 3);
      EnableItem(fileMenu, 4);
      enableItem(NextMenu, 1);
      DrawMenuBar;
      initCursor;
      where.h := 80;
      where.v := 100;
      typeList[0] := 'TEXT';
      SFGetFile(where, '', NIL, 1, TypeList, NIL, reply);
      IF reply.good THEN
         BEGIN
            doneWithFile := false;
            open(theFile, reply.fName);
            findNext;
            WHILE NOT doneWithFile DO
               BEGIN
                  plot;
                  next := false;
                  miniloop;
                  eraseRect(blankrect);
                  findNext;
               END;{while}
            close(theFile);
         END;{then and if}
      DisableItem(nextMenu, 0);
      DisableItem(fileMenu, 4);
      EnableItem(fileMenu, 3);
      DrawMenuBar;
      DrawMenuBar;
   END;{openplot}


   PROCEDURE Setup;
   BEGIN
```

```
      Initgraf(@thePort);
      Initfonts;
      FlushEvents(everyEvent, 0);
      SetEventMask(everyEvent);
      InitCursor;
      InitWindows;
      SetRect(WindowRect, 1, 20, 510, 350);
      setRect(blankRect, 1, 1, 509, 310);
      myWindow := NewWindow(@wRecord, WindowRect, 'Contour Plot', false, 4,
pointer(-1), false, 0);
      ShowWindow(myWindow);
      SetPort(myWindow);
      myWindow^.txSize := 10;
      SelectWindow(myWindow);

   END;{Setup}
   PROCEDURE SetupMenus;
      VAR
         MenuTopic : MenuHandle;
         apple : char;
   BEGIN
      apple := '@';
      apple := chr(AppleMark);
      appleMenu := NewMenu(AppleID, apple);  {get the apple desk accessories
menu}
      AddResMenu(appleMenu, 'DRVR');      {adds all names into item list}
      InsertMenu(appleMenu, 0);           {put in list held by menu manager}

      fileMenu := NewMenu(FileID, 'File');   {always need this for Quiting}
      AppendMenu(FileMenu, 'Quit/Q');
      AppendMenu(fileMenu, '(-');
      AppendMenu(fileMenu, 'Open.../O;(Close ');
      InsertMenu(fileMenu, 0);

      editMenu := NewMenu(EditID, 'Edit');   {always need for editing Desk
Accessories}
      AppendMenu(EditMenu, 'Undo/z');
      Appendmenu(editMenu, '(-');
      appendMenu(editMenu, 'Cut/x;Copy/C;Paste/V;Clear');
      InsertMenu(editMenu, 0);

      nextMenu := newMenu(NextID, 'Next');
      AppendMenu(nextMenu, '(Next Plot');
      InsertMenu(nextMenu, 0);

      DrawMenuBar;           {all done so show the menu bar}
   END;

   PROCEDURE ProcessMenu_in; {(CodeWord : longint)}
      VAR
         Menu_No : integer;        {menu number that was selected}
         Item_No : integer;        {item in menu that was selected}
         NameHolder : Str255;         {name holder for desk accessory or font}
         DNA : integer;        {OpenDA will never return 0, so don't care}
   BEGIN
      IF CodeWord <> 0 THEN
         BEGIN {go ahead and process the command}
            Menu_No := HiWord(CodeWord);   {get the Hi word of...}
            Item_no := LoWord(CodeWord);   {get the Lo word of...}
```

```
            CASE Menu_No OF
               AppleID :
                  BEGIN
                     GetItem(GetMHandle(AppleID), Item_No, NameHolder);
                     DNA := OpenDeskAcc(NameHolder);
                  END;
               FileID :
                  BEGIN
                     CASE Item_No OF
                        1 :
                           BEGIN
                              Done := True;          {quit}
                              doneWithFile := true;
                           END;{quit}
                     {  2: -------}
                        3 :
                           openplot;
                        4 :
                           doneWithFile := true;   {close}
                     END;
                  END;
               EditID :
                  BEGIN
                     IF NOT SystemEdit(Item_no - 1) THEN {if not for a desk
accessory}
                        CASE Item_No OF
                           1 :
                              BEGIN
                              END;                {undo}
         { 2:                                line divider}
                           3 :
                              BEGIN
                              END;                {cut}
                           4 :
                              BEGIN
                              END;                {copy}
                           5 :
                              BEGIN
                              END;                {paste}
                           6 :
                              BEGIN
                              END;                {clear}
                        END;
                  END;
               NextID :
                  next := true;
            END;
            HiliteMenu(0);                {unhilite after processing menu}
         END;
   END; {of ProcessMenu_in procedure}

   PROCEDURE initThings;
   BEGIN{initThings}
      setupmenus;
   END;{initThings}

   PROCEDURE loop;
      VAR
```

```
        WindowPointedTo : WindowPtr;
        MouseLoc : Point;
        WindoLoc : integer;
   BEGIN

      REPEAT
         IF GetNextEvent(everyEvent, theEvt) THEN
            CASE theEvt.what OF
               mousedown :
                  BEGIN
                     MouseLoc := theEvt.Where;
                     WindoLoc := FindWindow(MouseLoc, WindowPointedTo);
                     CASE WindoLoc OF
                        inContent :
                           IF WindowPointedTo <> FrontWindow THEN
                              SelectWindow(WindowPointedTo);
                        inDrag :
                           ;
                        inGrow :
                           ;
                        inGoAway :
                           done := TrackGoAway(WindowPointedTo, MouseLoc);
                        inMenuBar :
                           ProcessMenu_in(MenuSelect(MouseLoc));
                        inSysWindow :
                           SystemClick(theEvt, WindowPointedTo);
                        OTHERWISE
                           ;
                     END;{case theEvt.what}
                  END;{mousedown}
               mouseup :
                  ;
               updateEvt, activateEvt :
                  ;
               OTHERWISE
                  ;
            END;{case what event}
{and of then}
      UNTIL done;
   END;{loop}

BEGIN{main program}
   Setup;
   initThings;
   done := false;
   loop;


END.{Main program}

END.
```

D.3 Three-D

```
PROGRAM ThreeD;
{This program produces a projection of a three dimensional graph from
data encoded and sent by subroutine Threed (in FORTRAN) from the Mainframe
It supports the full Macintosh Interface and shares much of its
Macintosh skeleton with contour.pas}

   CONST {menu stuff}
      AppleID = 1000;
      FileID = 1001;
      EditID = 1002;
      nextID = 1003;

   VAR
      done, doneWithFile, next : Boolean;
      theEvt : eventRecord;
      wRecord : windowRecord;
      myWindow, TheWindow : windowPtr;
      WindowRect, blankRect : Rect;
      appleMenu, fileMenu, editMenu, nextMenu : MenuHandle;
      pen : penState;
      thefile : text;
   PROCEDURE ProcessMenu_in (CodeWord : longint);
   forward;

   PROCEDURE plot;
      CONST
         numlines = 16;
         dhdi = 16;
         dhdj = 0;
         dhdk = -16;
         dvdi = 2;
         dvdj = -2;
         dvdk = 12;
         h0 = 256;
         v0 = 128;
      VAR
         d : char;
         nx, nz, i, j, k : integer;
         x, z, h, v : integer;
         dxdi, dzdk : integer;
         f : ARRAY[0..63, 0..63] OF integer;
   BEGIN
      readln(theFile, nx, nz);
      FOR x := 0 TO nx - 1 DO
         BEGIN
            FOR z := 0 TO nz - 1 DO
               BEGIN
                  read(theFile, d);
                  f[x, z] := ord(d) - 40;
               END;{for z}
            readln(thefile);
         END;{for x}
      dxdi := nx DIV numlines;
      dzdk := nz DIV numlines;
      FOR i := 0 TO numlines - 1 DO
         BEGIN
            x := i * dxdi;
```

```
            h := h0 + dhdi * i + dhdj * f[x, 0];
            v := v0 + dvdi * i + dvdj * f[x, 0];
            moveto(h, v);
            FOR z := 1 TO nz - 1 DO
               BEGIN
                  h := h0 + dhdi * i + (z * dhdk) DIV dzdk + dhdj * f[x, z];
                  v := v0 + dvdi * i + (z * dvdk) DIV dzdk + dvdj * f[x, z];
                  lineto(h, v);
               END;{for z}
         END;{for i}
      FOR k := 0 TO numlines - 1 DO
         BEGIN
            z := k * dzdk;
            h := h0 + dhdk * k + dhdj * f[0, z];
            v := v0 + dvdk * k + dvdj * f[0, z];
            moveto(h, v);
            FOR x := 1 TO nx - 1 DO
               BEGIN
                  h := h0 + dhdk * k + (x * dhdi) DIV dxdi + dhdj * f[x, z];
                  v := v0 + dvdk * k + (x * dvdi) DIV dxdi + dvdj * f[x, z];
                  lineto(h, v);
               END;{for x}
         END;{for k}
      h := h0;
      v := v0;
      moveto(h, v);
      getPenState(pen);
      pen.pnSize.v := 2;
      pen.pnsize.h := 2;
      setPenState(pen);
      line(64 * dhdj, 64 * dvdj);{y axis}
      moveto(h, v);
      line(17 * dhdi, 17 * dvdi);{x axis}
      moveto(h, v);
      line(17 * dhdk, 17 * dvdk);{z axis}
      pen.pnSize.v := 1;
      pen.pnsize.h := 1;
      setPenState(pen);

END;{plot}

PROCEDURE miniloop;
   VAR
      WindowPointedTo : WindowPtr;
      MouseLoc : Point;
      WindoLoc : integer;
BEGIN
   HiliteMenu(0);
   REPEAT
      IF GetNextEvent(everyEvent, theEvt) THEN
         CASE theEvt.what OF
            mousedown :
               BEGIN
                  MouseLoc := theEvt.Where;
                  WindoLoc := FindWindow(MouseLoc, WindowPointedTo);
                  CASE WindoLoc OF
                     inMenuBar :
                        ProcessMenu_in(MenuSelect(MouseLoc));
                     inSysWindow :
                        SystemClick(theEvt, WindowPointedTo);
```

```
                        OTHERWISE
                           ;
                     END;{case theEvt.what}
                  END;{mousedown}
               mouseup :
                  ;
               updateEvt, activateEvt :
                  ;
               OTHERWISE
                  ;
            END;{case what event}
{and of then}
      UNTIL done OR doneWithFile OR next;
   END;{miniloop}


   PROCEDURE findnext (VAR t : char);
      VAR
         one : char;
         ni : integer;
         Qstring : str255;
   BEGIN
      Qstring := 'Type s to skip, <ret> to plot';
      REPEAT
         IF eof(theFile) THEN
            DoneWithFile := true
         ELSE
            readln(theFile, one);
      UNTIL (one = '~') OR DoneWithFile;
      moveto(100, 10);
      ni := 1;
      IF (NOT donewithfile) THEN
         BEGIN
            WHILE ((NOT eoln(theFile)) AND (ni < 72)) DO
               BEGIN
                  read(theFile, t);
                  DrawChar(t);
                  ni := ni + 1
               END;{while}
            ni := 1;
            readln(theFile);
            moveto(100, 30);
            drawstring(qstring);
            read(t);
         END;{if}
      eraseRect(blankrect);
   END;{findnext}

   PROCEDURE openplot;
      VAR
         reply : SFreply;
         typeList : SFTypeList;
         where : point;
         t : char;
   BEGIN
      EnableItem(nextMenu, 0);
      DisableItem(fileMenu, 3);
      EnableItem(fileMenu, 4);
      enableItem(NextMenu, 1);
      DrawMenuBar;
```

```
      initCursor;
      where.h := 80;
      where.v := 100;
      typeList[0] := 'TEXT';
      SFGetFile(where, '', NIL, 1, TypeList, NIL, reply);
      IF reply.good THEN
         BEGIN
            doneWithFile := false;
            open(theFile, reply.fName);
            findNext(t);
            WHILE NOT doneWithFile DO
               BEGIN
                  CASE t OF
                     's' :
                        ;
                     OTHERWISE
                        BEGIN
                           plot;
                           next := false;
                           miniloop;
                           eraseRect(blankrect);
                        END;{otherwise}
                  END;{case}
                  IF NOT doneWithFile THEN
                     findNext(t);
               END;{while}
            close(theFile);
         END;{then and if}
      DisableItem(nextMenu, 0);
      DisableItem(fileMenu, 4);
      EnableItem(fileMenu, 3);
      DrawMenuBar;
      DrawMenuBar;
   END;{openplot}


   PROCEDURE Setup;
   BEGIN
      Initgraf(@thePort);
      Initfonts;
      FlushEvents(everyEvent, 0);
      SetEventMask(everyEvent);
      InitCursor;
      InitWindows;
      SetRect(WindowRect, 1, 20, 512, 350);
      setRect(blankRect, 0, 0, 511, 340);
      myWindow := NewWindow(@wRecord, WindowRect, 'Contour Plot', false, 4,
pointer(-1), false, 0);
      ShowWindow(myWindow);
      SetPort(myWindow);
      myWindow^.txSize := 10;
      SelectWindow(myWindow);

   END;{Setup}
   PROCEDURE SetupMenus;
      VAR
         MenuTopic : MenuHandle;
         apple : char;
   BEGIN
      apple := '@';
```

```
      apple := chr(AppleMark);
      appleMenu := NewMenu(AppleID, apple);  {get the apple desk accessories
menu}
      AddResMenu(appleMenu, 'DRVR');      {adds all names into item list}
      InsertMenu(appleMenu, 0);           {put in list held by menu manager}

      fileMenu := NewMenu(FileID, 'File');   {always need this for Quiting}
      AppendMenu(FileMenu, 'Quit/Q');
      AppendMenu(fileMenu, '(-');
      AppendMenu(fileMenu, 'Open.../O;(Close ');
      InsertMenu(fileMenu, 0);

      editMenu := NewMenu(EditID, 'Edit');   {always need for editing Desk
Accessories}
      AppendMenu(EditMenu, 'Undo/z');
      Appendmenu(editMenu, '(-');
      appendMenu(editMenu, 'Cut/x;Copy/C;Paste/V;Clear');
      InsertMenu(editMenu, 0);

      nextMenu := newMenu(NextID, 'Next');
      AppendMenu(nextMenu, '(Next Plot');
      InsertMenu(nextMenu, 0);

      DrawMenuBar;            {all done so show the menu bar}
   END;

   PROCEDURE ProcessMenu_in; {(CodeWord : longint)}
      VAR
         Menu_No : integer;         {menu number that was selected}
         Item_No : integer;         {item in menu that was selected}
         NameHolder : Str255;          {name holder for desk accessory or font}
         DNA : integer;         {OpenDA will never return 0, so don't care}
   BEGIN
      IF CodeWord <> 0 THEN
         BEGIN {go ahead and process the command}
            Menu_No := HiWord(CodeWord);    {get the Hi word of...}
            Item_no := LoWord(CodeWord);    {get the Lo word of...}

            CASE Menu_No OF
               AppleID :
                  BEGIN
                     GetItem(GetMHandle(AppleID), Item_No, NameHolder);
                     DNA := OpenDeskAcc(NameHolder);
                  END;
               FileID :
                  BEGIN
                     CASE Item_No OF
                        1 :
                           BEGIN
                              Done := True;           {quit}
                              doneWithFile := true;
                           END;{quit}
                     {  2: -------}
                        3 :
                           openplot;
                        4 :
                           doneWithFile := true;    {close}
                     END;
                  END;
               EditID :
```

```
                BEGIN
                   IF NOT SystemEdit(Item_no - 1) THEN {if not for a desk
accessory}
                      CASE Item_No OF
                         1 :
                            BEGIN
                            END;                {undo}
         { 2:                                line divider}
                         3 :
                            BEGIN
                            END;                {cut}
                         4 :
                            BEGIN
                            END;                {copy}
                         5 :
                            BEGIN
                            END;                {paste}
                         6 :
                            BEGIN
                            END;                {clear}
                      END;
                END;
             NextID :
                next := true;
          END;
          HiliteMenu(0);                {unhilite after processing menu}
       END;
   END; {of ProcessMenu_in procedure}

   PROCEDURE initThings;
   BEGIN{initThings}
      setupmenus;
   END;{initThings}

   PROCEDURE loop;
      VAR
         WindowPointedTo : WindowPtr;
         MouseLoc : Point;
         WindoLoc : integer;
   BEGIN

      REPEAT
         IF GetNextEvent(everyEvent, theEvt) THEN
            CASE theEvt.what OF
               mousedown :
                  BEGIN
                     MouseLoc := theEvt.Where;
                     WindoLoc := FindWindow(MouseLoc, WindowPointedTo);
                     CASE WindoLoc OF
                        inContent :
                           IF WindowPointedTo <> FrontWindow THEN
                              SelectWindow(WindowPointedTo);
                        inDrag :
                           ;
                        inGrow :
                           ;
                        inGoAway :
                           done := TrackGoAway(WindowPointedTo, MouseLoc);
                        inMenuBar :
                           ProcessMenu_in(MenuSelect(MouseLoc));
```

```
                        inSysWindow :
                           SystemClick(theEvt, WindowPointedTo);
                        OTHERWISE
                           ;
                     END;{case theEvt.what}
                  END;{mousedown}
               mouseup :
                  ;
               updateEvt, activateEvt :
                  ;
               OTHERWISE
                  ;
            END;{case what event}
{and of then}
      UNTIL done;
   END;{loop}

BEGIN{main program}
   Setup;
   initThings;
   done := false;
   loop;


END.{Main program}

END.
```

D.4 FFT

```
* THESE ROUTINES ARE THE FAST FOURIER TRANSFORM ROUTINES
* BASED ON PRESS ET AL
      SUBROUTINE FOUR1(DATA,NN,ISIGN)
* THIS SUBROUTINE IS FROM PRESS ET AL
      DOUBLE PRECISION WR,WI,WPR,WPI,WTEMP,THETA
      REAL DATA(2*NN)
      N=2*NN
      J=1
      DO 11 I=1,N,2
         IF(J.GT.I)THEN
            TEMPR=DATA(J)
            TEMPI=DATA(J+1)
            DATA(J)=DATA(I)
            DATA(J+1)=DATA(I+1)
            DATA(I)=TEMPR
            DATA(I+1)=TEMPI
            ENDIF
         M=N/2
    1    IF((M.GE.2).AND.(J.GT.M)) THEN
            J=J-M
            M=M/2
            GOTO 1
            ENDIF
         J=J+M
   11 CONTINUE
      MMAX=2
    2 IF(N.GT.MMAX) THEN
       ISTEP=2*MMAX
       THETA=6.28318530717959D0/(ISIGN*MMAX)
       WPR=-2.D0*DSIN(0.5D0*THETA)**2
       WPI=DSIN(THETA)
       WR=1.D0
       WI=0.D0
       DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
             J=I+MMAX
             TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
             TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
             DATA(J)=DATA(I)-TEMPR
             DATA(J+1)=DATA(I+1)-TEMPI
             DATA(I)=DATA(I)+TEMPR
             DATA(I+1)=DATA(I+1)+TEMPI
   12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI= WI*WPR+WTEMP*WPI+WI
   13     CONTINUE
       MMAX=ISTEP
       GOTO 2
       ENDIF
      RETURN
      END

      SUBROUTINE TWOFFT(DATA1,DATA2,FFT1,FFT2,N)
* THIS ROUTINE WAS ADAPTED FROM PRESS ET AL
* MODIFIED T.ESPINOLA
      REAL DATA1(N),DATA2(N)
      COMPLEX FFT1(N+1),FFT2(N+1),H1,H2,C1,C2
```

```
      C1=(0.5,0.0)
      C2=(0.0,-0.5)
      DO 11 J=1,N
         FFT1(J)=CMPLX(DATA1(J),DATA2(J))
   11    CONTINUE
      CALL FOUR1(FFT1,N,1)
      FFT1(N+1)=FFT1(1)
      N2=N+2
      DO 12 J=1,N/2+1
         H1=C1*(FFT1(J)+CONJG(FFT1(N2-J)))
         H2=C2*(FFT1(J)-CONJG(FFT1(N2-J)))
         FFT1(J)=H1
         FFT1(N2-J)=CONJG(H1)
         FFT2(J)=H2
         FFT2(N2-J)=CONJG(H2)
   12    CONTINUE
      RETURN
      END

* THE REMAINING ROUTINES ARE BY T. ESPINOLA
      SUBROUTINE FFTR(DATA1,DATA2,FFT1,N)
      REAL DATA1(N),DATA2(N)
      COMPLEX FFT1(N+1),H1,H2,C1,C2
      C1=(0.5,0.0)
      C2=(0.0,-0.5)
      DO 11 J=1,N
         FFT1(J)=CMPLX(DATA1(J),DATA2(J))
   11    CONTINUE
      CALL FOUR1(FFT1,N,1)
      FFT1(N+1)=FFT1(1)
      N2=N+2
      DO 12 J=1,N/2+1
         H1=C1*(FFT1(J)+CONJG(FFT1(N2-J)))
         H2=C2*(FFT1(J)-CONJG(FFT1(N2-J)))
         FFT1(J)=H1
         FFT1(N2-J)=H2
   12    CONTINUE
      RETURN
      END

      SUBROUTINE FFTI2(DATA1,DATA2,WORK,N)
      COMPLEX DATA1(N/2),DATA2(N/2),WORK(N+1),I
      I=(0.0,1.0)
      NN=N/2
      N2=N+2
      DO 10 J=2,NN
         WORK(J)=DATA1(J)+DATA2(J)*I
         WORK(N2-J)=CONJG(DATA1(J))+I*CONJG(DATA2(J))
   10    CONTINUE
      WORK(1)=CMPLX(REAL(DATA1(1)),REAL(DATA2(1)))
      WORK(NN+1)=CMPLX(AIMAG(DATA1(1)),AIMAG(DATA2(1)))
      CALL FOUR1(WORK,N,-1)
      RETURN
      END

      SUBROUTINE FFTI(DATA1,DATA2,WORK,N)
      REAL DATA1(N),DATA2(N)
      COMPLEX WORK(N+1)
      CALL FFTI2(DATA1,DATA2,WORK,N)
      DO 10 J=1,N
```

```
         DATA1(J)=REAL(WORK(J))/N
         DATA2(J)=AIMAG(WORK(J))/N
   10    CONTINUE
      RETURN
      END

      SUBROUTINE FFT2(DATA1,DATA2,WORK,N,ISIGN)
* THIS IS THE ROUTINE DIRECTLY CALLED BY GMHD
* DATA1 AND DATA2 ARE THE TWO ARRAYS TO BE TRANSFORMED
* THE TRANSFORMATION IS STORED IN PLACE
* WORK IS WORK SPACE AND MUST BE DIMENSIONED AT LEAST 4*(N+1)
* N IS THE SIZE OF THE ARRAY AND MUST BE A POWER OF 2
* ISIGN DETERMINES WHETHER IT IS A THE TRANSFORM OR AN INVERSE TRANSFORM
* ISIGN =1 FOR TRANSFORM    ISGN=-1 FOR INVERSE TRANSFORM
      REAL DATA1(N),DATA2(N),WORK(2*N+2,2)
      IF (ISIGN.EQ.1)THEN
         CALL TWOFFT(DATA1,DATA2,WORK(1,1),WORK(1,2),N)
         DO 10 I=1,N
            DATA1(I)=WORK(I,1)
            DATA2(I)=WORK(I,2)
   10       CONTINUE
         DATA1(2)=WORK(N+1,1)
         DATA2(2)=WORK(N+1,2)
      ELSE
         CALL FFTI(DATA1,DATA2,WORK,N)
      ENDIF
      RETURN
      END
```

D.5 Makcon

```
      SUBROUTINE MakCon(F,Z,NX,NZ,NUMLIN,TITLE)
* THESE ROUTINES MAKE A CONTOUR PLOT OF THE ARRAY F
* THE CELLS ARE EVENLY SPACED IN THE X DIRECTION WHILE THE ARRAY
* Z CONTAINS THE Z POSITION OF THE GRID POINTS
* THE PLOTTING INFORMATION IS ENCODED AS CHARACTERS TO BE DOWNLOADED
* TO THE MAC
      IMPLICIT NONE
* PASSED VARIABLES
      INTEGER NX,NZ,NUMLIN
      REAL F(NX,NZ),Z(NX,NZ)
      CHARACTER*20 TITLE
* LOCAL VARIABLES
      INTEGER NY,NH,NAX,MAXA,NHMAX,I,J,NYY,MAXAA
      PARAMETER(NY=292,MAXA=1024,NHMAX=292)
      REAL A(NY,0:33),FMIN,FMAX,ZMIN,ZMAX,DF,DZ
      INTEGER FIRST(NY),LAST(NY)
      COMMON /CONZZL/ NH,NYY,MAXAA,DZ,FMIN,ZMIN,DF
      IF(NX.LT.8)THEN
      PRINT *,'NX MUST >= 8'
      RETURN
      ENDIF
      NYY=NY
      MAXAA=MAXA
      CALL FINDRA(F,NX,NZ,FMIN,FMAX)
      CALL FINDRA(Z,NX,NZ,ZMIN,ZMAX)
      IF(FMIN.EQ.FMAX.OR.ZMIN.EQ.ZMAX)THEN
         PRINT*,'Error in array passed to countour.'
         PRINT*,'Function or z has no variation.'
      ELSE
      PRINT *,CHAR(126),' Start of plot. ',TITLE
         DF=REAL(MAXA)/(FMAX-FMIN)
         DZ=(ZMAX-ZMIN)/NY
         J=1
         CALL GETCOL(FIRST,F,Z,NX,NZ,J)
         NAX=NHMAX/NX
         NH=1+(NX-1)*NAX

         DO 10 J=2,NX
            CALL GETCOL(LAST,F,Z,NX,NZ,J)
            CALL FILCOL(A,FIRST,LAST,NAX,NUMLIN)
            CALL OUTA(A,NAX)
            DO 20 I=1,NY
               FIRST(I)=LAST(I)
   20          CONTINUE
   10       CONTINUE
      ENDIF
      PRINT *,CHAR(107),'End of plot.'
      PRINT 1000,FMIN,FMAX,(fmax-fmin)/numlin
 1000 FORMaT(1X,'Min = ',f9.4,/,1x,'max= ',f9.4,/,1x,'contours= ',f9.5)
      RETURN
      END

      SUBROUTINE GETCOL(COL,F,Z,NX,NZ,J)
      IMPLICIT NONE
* GLOBAL VARIABLES
      INTEGER NH,NY,MAXA
      REAL DZ,FMIN,ZMIN,DF
```

```
      COMMON /CONZZL/NH,NY,MAXA,DZ,FMIN,ZMIN,DF
* PASSED VARIABLES
      INTEGER NX,NZ,J
      REAL F(NX,NZ),Z(NX,NZ)
      INTEGER COL(NY)
* LOCAL VARIABLES
      INTEGER K,L,K0,KK,K1
      REAL DELTA,F0
* BEGIN GETCOL

      K=1
      L=1
* WHILE
    1 IF(Z(J,L).GT.(ZMIN+K*DZ))THEN
      COL(K)=-MAXA
      K=K+1
         GOTO1
         ENDIF
      K1=K
      L=L+1
    3 IF (L.LT.NZ)THEN
         K0=K
         DELTA=DZ*(F(J,L+1)-F(J,L))/(Z(J,L+1)-Z(J,L))
         F0=F(J,L)-FMIN
    2    IF(Z(J,L).GT.(ZMIN+K*DZ))THEN
            COL(K)=(F0+(K-K0)*DELTA)*DF
            K=K+1
            GOTO 2
            ENDIF
      L=L+1
     GOTO 3
      ENDIF
      COL(K)=(F(J,NZ)-FMIN)*DF
      DO 10 KK=K+1,NY
         COL(KK)=-MAXA
   10 CONTINUE
      RETURN
      END

      SUBROUTINE FILCOL(A,FIRST,LAST,NAX,NUMLIN)
      IMPLICIT NONE
* GLOBAL VARIABLES
      INTEGER NH,NY,MAXA
      REAL DZ,FMIN,ZMIN,DF
      COMMON /CONZZL/NH,NY,MAXA,DZ,FMIN,ZMIN,DF
* PASSED VARIABLES
      INTEGER A(NY,0:33),FIRST(NY),LAST(NY),NAX,NUMLIN
* LOCAL VARIABLES
      INTEGER I,K
      REAL DELTA,DC
* BEGIN FILCOL
      DO 10 K=1,NY
         A(K,0)=FIRST(K)
         A(K,NAX)=LAST(K)
         DELTA=REAL(LAST(K)-FIRST(K))/NAX
         DO 20 I= 1,NAX-1
            A(K,I)=FIRST(K)+DELTA*I
   20       CONTINUE
   10    CONTINUE
      DC=REAL(NUMLIN)/REAL(MAXA)*.5
```

```
      DO 30 I=0,NAX
      DO 30 K=1,NY
         A(K,I)=(A(K,I)+ABS(A(K,I)))*DC
   30    CONTINUE
      RETURN
      END

      SUBROUTINE OUTA(A,NAX)
      IMPLICIT NONE
* GLOBAL VARIABLES
      INTEGER NH,NY,MAXA
      REAL DZ,FMIN,ZMIN,DF
      COMMON /CONZZL/NH,NY,MAXA,DZ,FMIN,ZMIN,DF
* PASSED VARIABLES
      INTEGER A(NY,0:33),NAX
* LOCAL VARIABLES
      INTEGER NUMOUT,COUNT,K,I
      CHARACTER*1 OUTIT(5000)

* BEGIN OUTA
      NUMOUT=1
      DO 10 I=0,NAX-1
         OUTIT(NUMOUT)=CHAR(106)
         NUMOUT=NUMOUT+1
         K=1
    1    IF(K.LT.NY)THEN
            COUNT=1
    2       IF((COUNT.LE.64).AND.(K.LT.NY))THEN
               IF( ( (A(K,I).EQ.A(K,I+1)) .OR. (A(K,I)*A(K,I+1).EQ.0))
     $.AND. ((A(K,I).EQ.A(K+1,I)).OR.(A(K,I)*A(K+1,I).EQ.0)))THEN
                  COUNT=COUNT+1
               ELSE
                  OUTIT(NUMOUT)=CHAR(40+COUNT)
                  NUMOUT=NUMOUT+1
                  COUNT=1
               ENDIF
               K=K+1
               GOTO 2
               ENDIF
            IF(COUNT.GE.65)THEN
               OUTIT(NUMOUT)=CHAR(40+COUNT)
               NUMOUT=NUMOUT+1
               ENDIF
            GOTO 1
            ENDIF
   10    CONTINUE
      WRITE(*,1000)(OUTIT(I),I=1,NUMOUT-1)
 1000 FORMAT(1X,80A1)
      RETURN
      END

      SUBROUTINE FINDRA(F,NX,NZ,FMIN,FMAX)
* PASSED VARIABLES
      INTEGER NX,NZ
      REAL F(NX,NZ),FMIN,FMAX
* LOCAL VARIABLES
      INTEGER J,K
* BEGIN FINDRA
      FMIN=F(1,1)
      FMAX=F(1,1)
```

```
      DO 10 J=1,NX
      DO 10 K=1,NZ
         FMIN=AMIN1(FMIN,F(J,K))
         FMAX=AMAX1(FMAX,F(J,K))
10       CONTINUE
      RETURN
      END
```

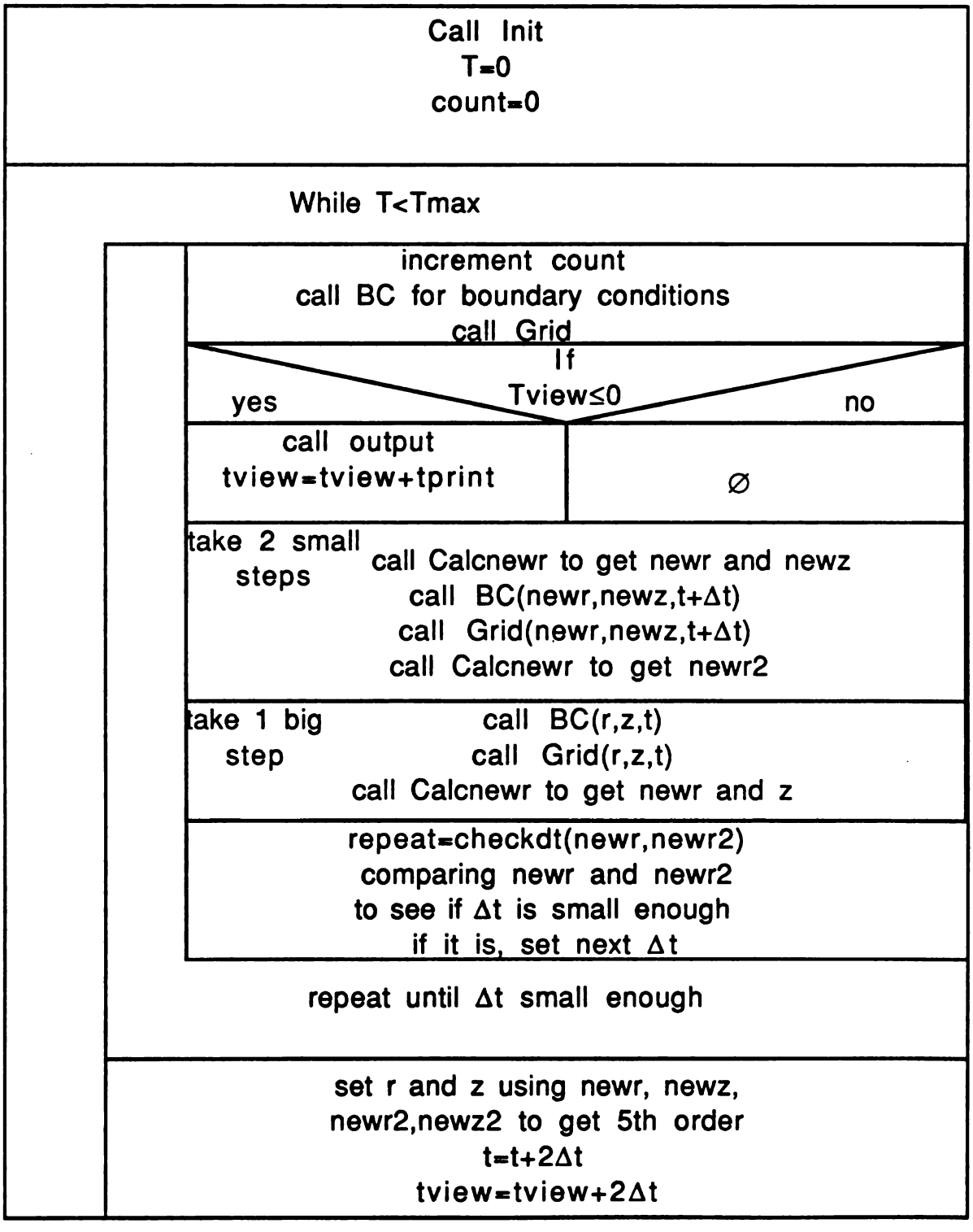
D.6 Make3D

```
      SUBROUTINE THREED(F,NX,NZ,TITLE)
* THESE ROUTINES PRODUCE A 3D PROJECTION GRAPH OF THE DATA IN ARRAY F
* IT DOES NOT ACCOUNT FOR THE Z POSITION OF THE GRID POINTS BUT PLOTS
* THEM EQUALLY SPACED. THE PLOTTING INFORMATION IS ENCODED AS CHARACTERS
* FOR DOWNLOADING TO THE MAC. THIS IS A BETA
      IMPLICIT NONE
* PASSED VARIABLES
      INTEGER NX,NZ
      REAL F(-1:NX,0:NZ-1)
      CHARACTER*40 TITLE
* LOCAL VARIABLES
      REAL MAX,MIN,DF,DX,DZ,D,X,Z
      INTEGER J,K,I,NH,NV,NL,KK
      CHARACTER*1 A(0:63,0:63)

      MAX=F(0,0)
      MIN=F(0,0)
      DO 5 J=0,NX-1
      DO 5 K=0,NZ-1
         MIN=AMIN1(MIN,F(J,K))
         MAX=AMAX1(MAX,F(J,K))
    5    CONTINUE
      IF(MAX.EQ.MIN)THEN
         PRINT *,'NO VARIATION IN PLOT ',TITLE
         RETURN
         ENDIF
      DF=64/(MAX-MIN)
      PRINT *,'~'
      WRITE( *,2000)TITLE(1:30),MAX,MIN
 2000 FORMAT(1X,A30,' MAX=',F10.4,' MIN=',F10.4)
      NH=64
      NV=64

      DX=FLOAT(NX-1)/(NH-1)
      DZ=FLOAT(NZ-1)/(NV-1)
      DO 10 K=0,NV-1
         Z=K*DZ
         KK=Z
         DO 20 J=0,NH-1
            X=J*DX
            I=X
            D=F(I,KK)+(F(I+1,KK)-F(I,KK))*(X-I)+
     $        (F(I,KK+1)-F(I,KK))*(Z-KK)
            A(J,K)=CHAR(IFIX((D-MIN)*DF+40))
   20       CONTINUE
   10 CONTINUE

      PRINT *,NH,NV
      DO 100 K=0,NV-1
         WRITE(*,1000)(A(J,K),J=0,NH-1)
 1000    FORMAT(1X,80A1)
  100 CONTINUE
      END
```

Appendix E
The Program

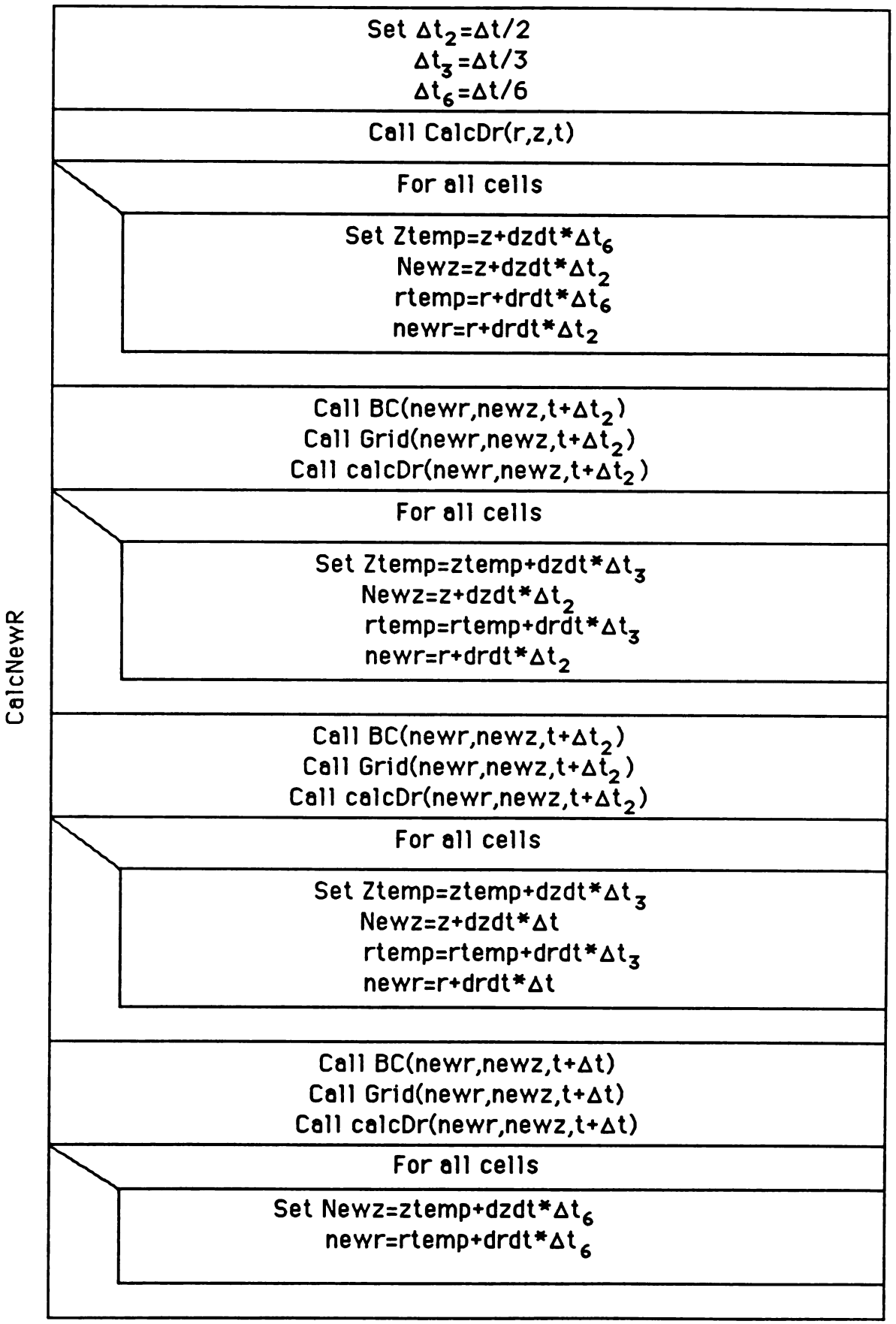
The main program and its included subroutines and functions are given below. I have included the Nassi-Shneiderman diagrams, where appropriate to illustrate their structure.

Main Program



```
      PROGRAM GMHD
* THIS IS THE MAIN PROGRAM FOR DOING GRAVITATIONAL MAGNETOHYDRODYMIC
* SIMULATIONS.  IT IS THE 2D VERSION USED PRIMARILY ON THE VAX
* IT DOES NOT CONTAIN ALL THE VECTORIZING OPTIMIZATION
* IT IS ONLY SPARSELY COMMENTED BECAUSE IT IS DOWNLOADED AT 1200 BAUD
      IMPLICIT NONE
* GLOBAL VARIABLES
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      INTEGER IFAX(13),INC,JUMP,NPTS,NVEC
      REAL WORK(0:31,1:50,8),TRIGS(451)
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL MASS,ENERGY,MASS0,ENERGY0,DENERGY,DMASS,OLDM,OLDE
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
      INTEGER SYMFLAG
      REAL C(0:31,1:50,8)
     0COMMON /FFT/WORK,SYMFLAG
     1       /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     2       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     3       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     4       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     5       /CHECK/MASS,ENERGY,MASS0,ENERGY0,DENERGY,DMASS,OLDM,OLDE
     6       /GRID/DZDX,DZDL,DZDT,DX
     7       /SOMMER/C
* LOCAL VARIABLES
     0REAL R(0:31,1:50,8),NEWR(0:31,1:50,8),QR(0:31,5,8),
     1     Z(0:31,1:50),NEWZ(0:31,1:50),
     2     NEWR2(0:31,1:50,8),NEWZ2(0:31,1:50),NEWT,T,
     3     DT,DTX2,CURRENT,DT2
      INTEGER J,L,M,COUNT
      LOGICAL REPEAT,CHECKDT

* BEGIN MAIN PROGRAM
      CALL INIT(R,Z,DT)
      T=0.0
      COUNT=0
* THE MAIN LOOP-- WHILE T<TMAX
    2 IF(T.LE.(TMAX+.01))THEN
* A DO-UNTIL LOOP WHICH REPEATS UNTIL THE ERROR IS SMALL ENOUGH
    3 CONTINUE
      COUNT=COUNT+1
         CALL BC(R,Z,T)
         CALL GRID(R,Z,T,DT)
         IF(TVIEW.LE.1E-6) THEN
      PRINT 1000,T,COUNT,T/COUNT
 1000FORMAT(' T=',F9.5,'  STEPS=',I5,'  AVERAGE DT=',F9.7)
            CALL OUTPUT(R,T,Z)
            TVIEW=TVIEW+TPRINT
            ENDIF
         CALL CALCNEWR(R,NEWR,Z,NEWZ,T,DT)
         NEWT=T+DT
         CALL BC(NEWR,NEWZ,NEWT)
         CALL GRID(NEWR,NEWZ,NEWT,DT)
         CALL CALCNEWR(NEWR,NEWR2,NEWZ,NEWZ2,NEWT,DT)
         DTX2=2.*DT
         CALL BC(R,Z,T)
         CALL GRID(R,Z,T,DT)
         CALL CALCNEWR(R,NEWR,Z,NEWZ,T,DTX2)
         REPEAT=CHECKDT(R,NEWR,NEWR2,T,DT)
```

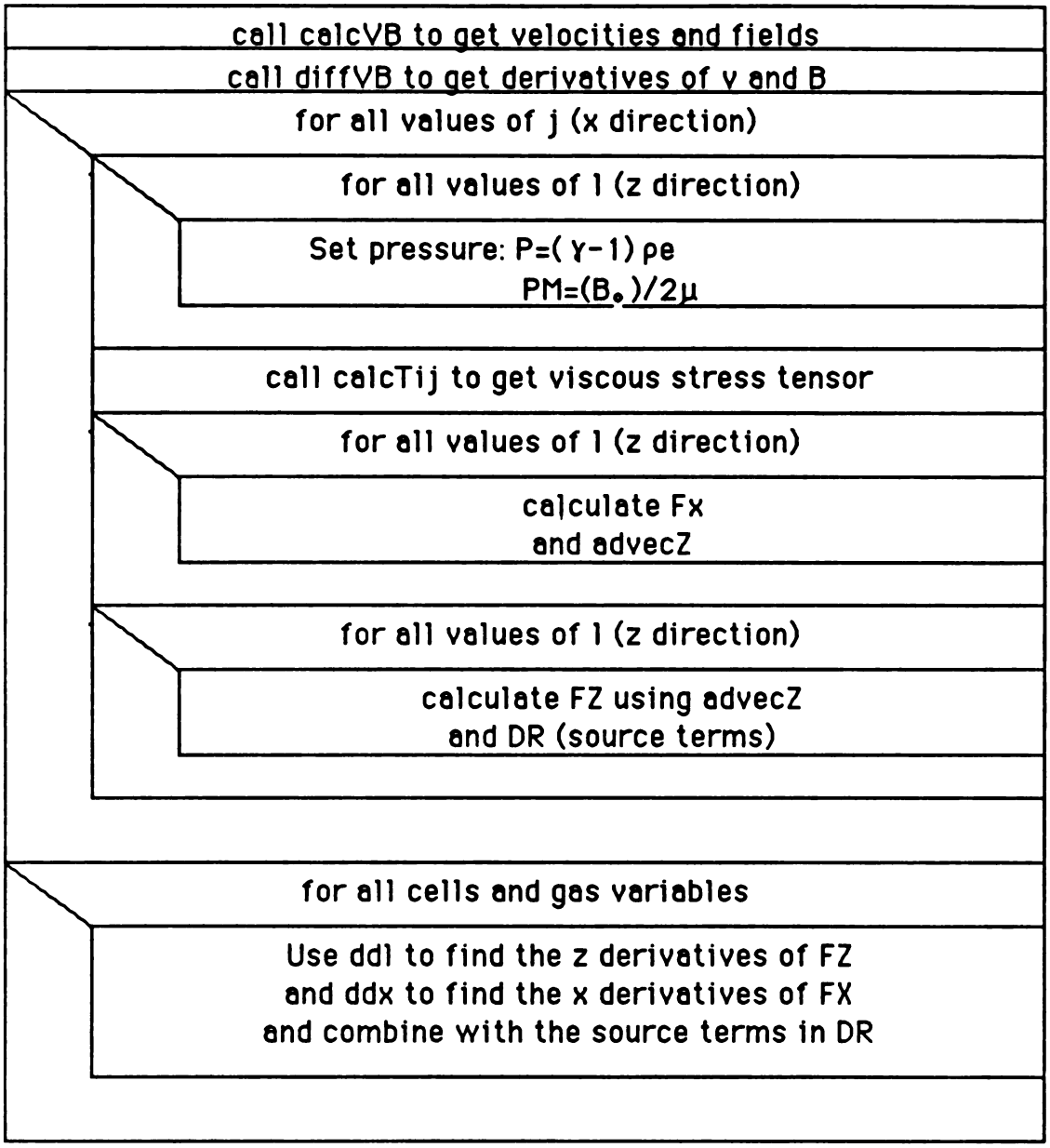
```
         IF(REPEAT)GOTO 3
* END OF DO UNTIL ERROR IS SMALL ENOUGH
* USE THE EXTRA INFO TO FIND NEWR TO 5TH ORDER
      DO 10 M=1,8
      DO 10 J=XLO,XHI
         DO 10 L=ZLO,ZHI
C       R(J,L,M)=NEWR2(J,L,M)
C            R(J,L,M)=F16D15*NEWR2(J,L,M)-F1D15*NEWR(J,L,M)
      R(J,L,M)=NEWR2(J,L,M)+F1D15*(NEWR2(J,L,M)-NEWR(J,L,M))
   10       CONTINUE
      DO 20 J=XLO,XHI
      DO 20 L=ZLO,ZHI
C       Z(J,L)=NEWZ2(J,L)
C           Z(J,L)=F16D15*NEWZ2(J,L)-F1D15*NEWZ(J,L)
      Z(J,L)=NEWZ2(J,L)+F1D15*(NEWZ2(J,L)-NEWZ(J,L))
   20    CONTINUE
      T=T+DTX2
      TVIEW=TVIEW-DTX2
      GOTO 2
* END OF WHILE
      ENDIF
      PRINT *,'REACHED MAX TIME OF ',TMAX
      END
```


CalcNewR
Set $\Delta t_2=\Delta t/2$
$\Delta t_3=\Delta t/3$
$\Delta t_6=\Delta t/6$
Call CalcDr(r,z,t)
For all cells
Set Ztemp=z+dzdt*Δt_6
Newz=z+dzdt*Δt_2
rtemp=r+drdt*Δt_6
newr=r+drdt*Δt_2
Call BC(newr,newz,t+Δt_2)
Call Grid(newr,newz,t+Δt_2)
Call calcDr(newr,newz,t+Δt_2)
For all cells
Set Ztemp=ztemp+dzdt*Δt_3
Newz=z+dzdt*Δt_2
rtemp=rtemp+drdt*Δt_3
newr=r+drdt*Δt_2
Call BC(newr,newz,t+Δt_2)
Call Grid(newr,newz,t+Δt_2)
Call calcDr(newr,newz,t+Δt_2)
For all cells
Set Ztemp=ztemp+dzdt*Δt_3
Newz=z+dzdt*Δt
rtemp=rtemp+drdt*Δt_3
newr=r+drdt*Δt
Call BC(newr,newz,t+Δt)
Call Grid(newr,newz,t+Δt)
Call calcDr(newr,newz,t+Δt)
For all cells
Set Newz=ztemp+dzdt*Δt_6
newr=rtemp+drdt*Δt_6

```
*********************************************************************
* THIS SUBROUTINE USES A FOURTH ORDER SCHEME TO FIND THE NEW
* VARIABLES.
*********************************************************************

      SUBROUTINE CALCNEWR(R,NEWR,Z,NEWZ,T,DT)
      IMPLICIT NONE
*  PASSED VARIABLES
     0REAL  R(0:31,1:50,8),NEWR(0:31,1:50,8),
     1      Z(0:31,1:50),NEWZ(0:31,1:50),T,DT

* GLOBAL VARIABLES
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)

     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     2       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     3       /GRID/DZDX,DZDL,DZDT,DX

* LOCAL VARIABLES
     0REAL DT2,DT3,DT6,DR(0:31,1:50,8),RTEMP(0:31,1:50,8),
     1     ZTEMP(0:31,1:50),NEWT
      INTEGER J,L,M

* BEGIN CALCNEWR
* FIND THE DT'S NEEDED
      DT2=.5*DT
      DT3=DT/3.0
      DT6=.5*DT3

      CALL CALCDR(R,Z,DR,T,DT2)
* DR AT T
      DO 10 J=XLO,XHI
         DO 20 L=ZLO,ZHI
            ZTEMP(J,L)=Z(J,L)+DT6*DZDT(J,L)
            NEWZ(J,L)=Z(J,L)+DT2*DZDT(J,L)
 20         CONTINUE
         DO 30 M=1,8
         DO 30 L=ZLO,ZHI
            NEWR(J,L,M)=R(J,L,M)+DT2*DR(J,L,M)
            RTEMP(J,L,M)=R(J,L,M)+DT6*DR(J,L,M)
 30         CONTINUE
 10    CONTINUE
      NEWT=T+DT2
      CALL BC(NEWR,NEWZ,NEWT)
      CALL GRID(NEWR,NEWZ,NEWT,DT)
      CALL CALCDR(NEWR,NEWZ,DR,NEWT,DT2)
* DR AT T+DT/2 FIRST TIME
      DO 40 J=XLO,XHI
         DO 50 L=ZLO,ZHI
            ZTEMP(J,L)=ZTEMP(J,L)+DT3*DZDT(J,L)
 50         NEWZ(J,L)=Z(J,L)+DT2*DZDT(J,L)
         DO 60 M=1,8
         DO 60 L=ZLO,ZHI
            NEWR(J,L,M)=R(J,L,M)+DT2*DR(J,L,M)
            RTEMP(J,L,M)=RTEMP(J,L,M)+DT3*DR(J,L,M)
 60    CONTINUE
```

```
   40 CONTINUE

      CALL BC(NEWR,NEWZ,NEWT)
      CALL GRID(NEWR,NEWZ,NEWT,DT)
      CALL CALCDR(NEWR,NEWZ,DR,NEWT,DT)
* DR AT T+DT/2 SECOND TIME
      DO 70 J=XLO,XHI
         DO 80 L=ZLO,ZHI
            ZTEMP(J,L)=ZTEMP(J,L)+DT3*DZDT(J,L)
   80       NEWZ(J,L)=Z(J,L)+DT*DZDT(J,L)
         DO 90 M=1,8
         DO 90 L=ZLO,ZHI
            NEWR(J,L,M)=R(J,L,M)+DT*DR(J,L,M)
            RTEMP(J,L,M)=RTEMP(J,L,M)+DT3*DR(J,L,M)
   90    CONTINUE
   70 CONTINUE

      NEWT=T+DT
      CALL BC(NEWR,NEWZ,NEWT)
      CALL GRID(NEWR,NEWZ,NEWT,DT)
      CALL CALCDR(NEWR,NEWZ,DR,NEWT,DT)
* DR AT T+DT

      DO 110 J=XLO,XHI
         DO 120 L=ZLO,ZHI
  120       NEWZ(J,L)=ZTEMP(J,L)+DT6*DZDT(J,L)
         DO 130 M=1,8
         DO 130 L=ZLO,ZHI
            NEWR(J,L,M)=RTEMP(J,L,M)+DT6*DR(J,L,M)
  130       CONTINUE
  110 CONTINUE
C      CALL SOMMER(R,NEWR,DT,Z,NEWZ)
      CALL BC(NEWR,NEWZ,NEWT)
      RETURN
      END
```

calcDR




```
************************************************************************
* THIS SUBROUTINE CALCULATES THE DERIVATIVES TO BE USED IN CALCNEWR
************************************************************************
      SUBROUTINE CALCDR(R,Z,DR,T,DT)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),Z(0:31,1:50),DR(0:31,1:50,8),T,DT
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      REAL MASS,ENERGY,MASS0,ENERGY0,DENERGY,DMASS,OLDM,OLDE
      COMPLEX DX(0:31)
      INTEGER FXSYM(8)
```

```
     0COMMON
     1    /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     2    /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     3    /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     4    /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     5    /CHECK/MASS,ENERGY,MASS0,ENERGY0,DENERGY,DMASS,OLDM,OLDE
     6    /GRID/DZDX,DZDL,DZDT,DX
     7    /SYMS/FXSYM

* LOCAL VARIABLES
      INTEGER J,L,M
     0REAL  ETA(1:50),P(1:50),PM(1:50),
     1      U(0:31,1:50),V(0:31,1:50),W(0:31,1:50),TXX(1:50),
     2      TXY(1:50),TXZ(1:50),TYY(1:50),TYZ(1:50),TZZ(1:50),
     3      DUDX(0:31,1:50),DWDX(0:31,1:50),ADVECZ(1:50),
     4      DUDZ(0:31,1:50),DVDZ(0:31,1:50),DWDZ(0:31,1:50),
     5      DVDX(0:31,1:50),BX(0:31,1:50),BY(0:31,1:50),
     6      BZ(0:31,1:50),DBXDX(0:31,1:50),DBYDX(0:31,1:50),
     7      DBZDX(0:31,1:50),DBXDZ(0:31,1:50),DBYDZ(0:31,1:50),
     8      DBZDZ(0:31,1:50),CURLB2(1:50),FX(0:31,1:50,8),
     9      FZ(0:31,1:50,8)

* BEGIN CALCDR
* GET THE VELOCITIES, B FIELDS, AND THEIR DERIVATIVES
      CALL CALCVB(R,U,V,W,BX,BY,BZ)
     0CALL DIFFVB(U,DUDX,DUDZ,V,DVDX,DVDZ,W,DWDX,DWDZ,
     1             BX,DBXDX,DBXDZ,BY,DBYDX,DBYDZ,BZ,DBZDX,DBZDZ)

* THE MAIN LOOP OVER J FOR CALCULATING FX,FZ,DR
      DO 5 J=XLO,XHI
* CALCULATE THE PRESSURES,ETA, AND THE CURL OF B SQUARED
      DO 10 L=ZLO,ZHI
         P(L)=GM1*R(J,L,5)/DZDL(J,L)
         PM(L)=(BX(J,L)*BX(J,L)+BY(J,L)*BY(J,L)+BZ(J,L)*BZ(J,L))*OVER8PI
     0   CURLB2(L)=DBYDZ(J,L)*DBYDZ(J,L)+DBYDX(J,L)*DBYDX(J,L)
     1             +(DBXDZ(J,L)-DBZDX(J,L))*(DBXDZ(J,L)-DBZDX(J,L))
         ETA(L)=ETA0
   10 CONTINUE

* CALCULATE THE VISCOUS STRESS TENSOR TIJ
     0CALL CALCTIJ(R,J,TXX,TXY,TXZ,TYY,TYZ,TZZ,
     1             U,DUDX,DUDZ,V,DVDX,DVDZ,W,DWDX,DWDZ)
*=================================================================
* BLOCK:CALCFX
* CALCULATE FX AND ADVECZ
*=================================================================
      DO 20 L=ZLO,ZHI
         FX(J,L,1)=R(J,L,2)
         FX(J,L,2)=R(J,L,2)*U(J,L)+(P(L)+PM(L)-TXX(L))*DZDL(J,L)
         FX(J,L,3)=R(J,L,3)*U(J,L)-TXY(L)*DZDL(J,L)
         FX(J,L,4)=R(J,L,4)*U(J,L)-TXZ(L)*DZDL(J,L)
         FX(J,L,5)=R(J,L,5)*U(J,L)
         FX(J,L,6)=0.
     0   FX(J,L,7)=(BY(J,L)*U(J,L)-BX(J,L)*V(J,L)-ETA(L)*DBYDX(J,L))
     1             *DZDL(J,L)
     0   FX(J,L,8)=(BZ(J,L)*U(J,L)-BX(J,L)*W(J,L)+
     1             ETA(L)*(DBXDZ(J,L)-DBZDX(J,L)))*DZDL(J,L)
* CALCULATE THE ADVECTION TERM
         ADVECZ(L)=(W(J,L)-DZDT(J,L))/DZDL(J,L)
   20 CONTINUE

*=================================================================
```

```
* BLOCK: CALCFZ
* CALCULATE FZ, DR
*=================================================================
      DO 30 L=ZLO,ZHI
         FZ(J,L,1)=ADVECZ(L)*R(J,L,1)-FX(J,L,1)*DZDX(J,L)
         FZ(J,L,2)=ADVECZ(L)*R(J,L,2)-TXZ(L)-FX(J,L,2)*DZDX(J,L)
         FZ(J,L,3)=ADVECZ(L)*R(J,L,3)-TYZ(L)-FX(J,L,3)*DZDX(J,L)
         FZ(J,L,4)=ADVECZ(L)*R(J,L,4)+P(L)+PM(L)-TZZ(L)
     $      -FX(J,L,4)*DZDX(J,L)
         FZ(J,L,5)=ADVECZ(L)*R(J,L,5)-FX(J,L,5)*DZDX(J,L)
         FZ(J,L,6)=BX(J,L)*W(J,L)-BZ(J,L)*U(J,L)+ETA(L)*
     $    (DBZDX(J,L)-DBXDZ(J,L))-BX(J,L)*DZDT(J,L)-FX(J,L,6)*DZDX(J,L)
         FZ(J,L,7)=BY(J,L)*W(J,L)-BZ(J,L)*V(J,L)-
     $    ETA(L)*DBYDZ(J,L)-BY(J,L)*DZDT(J,L)
         FZ(J,L,8)=-BZ(J,L)*DZDT(J,L)-FX(J,L,8)*DZDX(J,L)
* NOW DR
         DR(J,L,1)=QH*R(J,L,4)
         DR(J,L,2)=QH*R(J,L,4)*U(J,L)
     $       +2.*(R(J,L,6)*DBXDX(J,L)+R(J,L,8)*DBXDZ(J,L))*OVER8PI
         DR(J,L,3)=QH*R(J,L,4)*V(J,L)
     $       +2.*(R(J,L,6)*DBYDX(J,L)+R(J,L,8)*DBYDZ(J,L))*OVER8PI
         DR(J,L,4)=QH*R(J,L,4)*W(J,L)
     $       +2.*(R(J,L,6)*DBZDX(J,L)+R(J,L,8)*DBZDZ(J,L))*OVER8PI
         DR(J,L,5)=(-P(L)*(DUDX(J,L)+DWDZ(J,L))+
     $    TXX(L)*DUDX(J,L)+TZZ(L)*DWDZ(J,L)+
     $    TXY(L)*(DVDX(J,L))+TYZ(L)*(DVDZ(J,L))
     $    +TXZ(L)*(DUDZ(J,L)))*DZDL(J,L)+QH*R(J,L,4)*R(J,L,5)/R(J,L,1)
     $       +2.*ETA(L)*OVER8PI*CURLB2(L)*DZDL(J,L)
         DR(J,L,6)=0.
         DR(J,L,7)=0.
         DR(J,L,8)=0.
   30 CONTINUE
* END OF LOOP
    5 CONTINUE
*=================================================================
*BLOCK: COMBINEFZ
* NOW COMBINE DR WITH THE DERIVATIVE OF FZ AND STORE IN DR
* USE V AND DVDZ AS A TEMP STORAGE FOR FZ AND DFZDZ SINCE
* FZ DOES NOT HAVE THE RIGHT SHAPE
*=================================================================

      DO 60 M=1,8
         DO 40 J=XLO,XHI
         DO 40 L=ZLO,ZHI
            V(J,L)=FZ(J,L,M)
   40       CONTINUE
         CALL DDL(V,DVDZ)
         DO 50 J=XLO,XHI
         DO 50 L=ZLO,ZHI
            DR(J,L,M)=DR(J,L,M)-DVDZ(J,L)
   50       CONTINUE
   60   CONTINUE
*=================================================================
* BLOCK:COMBINEFX
* NOW COMBINE DR WITH X DERIVATIVE OF FX AND STORE IN DR
* USE V AS TEMP STORAGE FOR FX AND DFXDX
*=================================================================

      DO 90 M=1,8
         DO 70 J=XLO,XHI
         DO 70 L=ZLO,ZHI
            V(J,L)=FX(J,L,M)
```

```
   70     CONTINUE
        CALL DDX(FXSYM(M),V,DVDZ)
          DO 80 J=XLO,XHI
          DO 80 L=ZLO,ZHI
             DR(J,L,M)=DR(J,L,M)-DVDZ(J,L)
   80     CONTINUE
   90    CONTINUE
*=================================================================
      RETURN
      END



*******************************************************************
      SUBROUTINE CALCVB(R,U,V,W,BX,BY,BZ)
      IMPLICIT NONE
* PASSED VARIABLES
     0REAL  R(0:31,1:50,8),U(0:31,1:50),V(0:31,1:50),
     1      W(0:31,1:50),BX(0:31,1:50),BY(0:31,1:50),BZ(0:31,1:50)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /GRID/DZDX,DZDL,DZDT,DX
* LOCAL VARIABLES
      INTEGER J,L

* BEGIN CALCVB
      DO 10 J=XLO,XHI
      DO 10 L=ZLO,ZHI
         U(J,L)=R(J,L,2)/R(J,L,1)
         V(J,L)=R(J,L,3)/R(J,L,1)
         W(J,L)=R(J,L,4)/R(J,L,1)
         BX(J,L)=R(J,L,6)/DZDL(J,L)
         BY(J,L)=R(J,L,7)/DZDL(J,L)
         BZ(J,L)=R(J,L,8)/DZDL(J,L)
   10 CONTINUE
      RETURN
      END



*******************************************************************
     0SUBROUTINE DIFFVB(U,DUDX,DUDZ,V,DVDX,DVDZ,W,DWDX,DWDZ,
     1                   BX,DBXDX,DBXDZ,BY,DBYDX,DBYDZ,BZ,DBZDX,DBZDZ)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL  U(0:31,1:50),DUDX(0:31,1:50),DUDZ(0:31,1:50),
     1      V(0:31,1:50),DVDX(0:31,1:50),DVDZ(0:31,1:50),
     2      W(0:31,1:50),DWDX(0:31,1:50),DWDZ(0:31,1:50),
     3      BX(0:31,1:50),DBXDX(0:31,1:50),DBXDZ(0:31,1:50),
     4      BY(0:31,1:50),DBYDX(0:31,1:50),DBYDZ(0:31,1:50),
     5      BZ(0:31,1:50),DBZDX(0:31,1:50),DBZDZ(0:31,1:50)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /GRID/DZDX,DZDL,DZDT,DX
* LOCAL VARIABLES
      INTEGER J,L
```

```
* BEGIN DIFFVB
* CALCULATE X DERIVATIVES USING SPECTRAL METHODS
      CALL DDX(-1,U,DUDX)
      CALL DDX(1,V,DVDX)
      CALL DDX(1,W,DWDX)
      CALL DDX(1,BX,DBXDX)
      CALL DDX(-1,BY,DBYDX)
      CALL DDX(-1,BZ,DBZDX)

* CALCULATE DUDZ, DVDZ, DWDZ USING FINITE DIFFERENCE
      CALL DDL(U,DUDZ)
      CALL DDL(V,DVDZ)
      CALL DDL(W,DWDZ)
      CALL DDL(BX,DBXDZ)
      CALL DDL(BY,DBYDZ)
      CALL DDL(BZ,DBZDZ)
* CORRECT FOR MOVING GRID
      DO 10 J=XLO,XHI
      DO 10 L=ZLO,ZHI
         DUDX(J,L)=DUDX(J,L)-DZDX(J,L)*DUDZ(J,L)
         DVDX(J,L)=DVDX(J,L)-DZDX(J,L)*DVDZ(J,L)
         DWDX(J,L)=DWDX(J,L)-DZDX(J,L)*DWDZ(J,L)
         DBXDX(J,L)=DBXDX(J,L)-DZDX(J,L)*DBXDZ(J,L)
         DBYDX(J,L)=DBYDX(J,L)-DZDX(J,L)*DBYDZ(J,L)
         DBZDX(J,L)=DBZDX(J,L)-DZDX(J,L)*DBZDZ(J,L)

         DUDZ(J,L)=DUDZ(J,L)/DZDL(J,L)
         DVDZ(J,L)=DVDZ(J,L)/DZDL(J,L)
         DWDZ(J,L)=DWDZ(J,L)/DZDL(J,L)
         DBXDZ(J,L)=DBXDZ(J,L)/DZDL(J,L)
         DBYDZ(J,L)=DBYDZ(J,L)/DZDL(J,L)
         DBZDZ(J,L)=DBZDZ(J,L)/DZDL(J,L)
   10 CONTINUE
      RETURN
      END



**********************************************************************
     0SUBROUTINE CALCTIJ(R,J,TXX,TXY,TXZ,TYY,TYZ,TZZ,
     1                    U,DUDX,DUDZ,V,DVDX,DVDZ,W,DWDX,DWDZ)
      IMPLICIT NONE
* PASSED VARIABLES
      INTEGER J
     0REAL  R(0:31,1:50,8),TXX(1:50),TXY(1:50),TXZ(1:50),
     1      TYY(1:50),TYZ(1:50),TZZ(1:50),
     2      U(0:31,1:50),DUDX(0:31,1:50),DUDZ(0:31,1:50),
     3      V(0:31,1:50),DVDX(0:31,1:50),DVDZ(0:31,1:50),
     4      W(0:31,1:50),DWDX(0:31,1:50),DWDZ(0:31,1:50)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /GRID/DZDX,DZDL,DZDT,DX
     2       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
* LOCAL VARIABLES
      REAL DVKDXK(1:50)
* TO SAVE DIVISIONS, DVKDXK(L)=1/3 OF ACTUAL VALUE THIS CHANGES ZETA2
* to 1/3 OF THE VALUE IN THE PROGRAM
      INTEGER L
```

```
* BEGIN CALCTIJ
      DO 10 L=ZLO,ZHI
      DVKDXK(L)=(DUDX(J,L)+DWDZ(J,L))/3.
   10CONTINUE
* CALCULATE TXX, TXY, TXZ, TYY, TYZ, TZZ
      DO 20 L=ZLO,ZHI
         TXX(L)=ZETA1*(DUDX(J,L)-2.*DVKDXK(L))+ZETA2*DVKDXK(L)
         TXY(L)=ZETA1*DVDX(J,L)
         TXZ(L)=ZETA1*(DUDZ(J,L)+DWDX(J,L))
         TYY(L)=DVKDXK(L)*(ZETA2-2.*ZETA1)
         TYZ(L)=ZETA1*DVDZ(J,L)
         TZZ(L)=ZETA1*(DWDZ(J,L)-2.*DVKDXK(L))+ZETA2*DVKDXK(L)
   20CONTINUE
      RETURN
      END



**********************************************************************
* THIS SUBROUTINE FINDS THE Z DERIVATIVE OF THE ELEMENTS OF THE
* PASSED ARRAY.  THE ARRAY MUST BE SHAPED (0:NR,1:NL)
**********************************************************************

      SUBROUTINE DDL(F,DFDL)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL  F(0:31,1:50),DFDL(0:31,1:50)
      INTEGER SYM
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
* LOCAL VARIABLES
      INTEGER J,L,M
* BEGIN DDL     USING FOURTH ORDER EVERYWHERE
      DO 20 J=XLO,XHI
         DO 10 L=3,LM2
C          PRINT *,J,L,DFDL(J,L)
             DFDL(J,L)=F(J,L-2)-8.*F(J,L-1)+8.*F(J,L+1)-F(J,L+2)
   10        CONTINUE
     0   DFDL(J,1)=-25.*F(J,1)+48.*F(J,2)-36.*F(J,3)+
     116.*F(J,4)-3.*F(J,5)
         DFDL(J,2)=-3.*F(J,1)-10.*F(J,2)+18.*F(J,3)-6.*F(J,4)+F(J,5)
     0   DFDL(J,LM)=-F(J,LM4)+6.*F(J,LM3)-18.*F(J,LM2)+10.*F(J,LM)
     1              +3.*F(J,NL)
     0   DFDL(J,NL)=3.*F(J,LM4)-16.*F(J,LM3)+36.*F(J,LM2)-
     148.*F(J,LM)+25.*F(J,NL)
   20    CONTINUE
  120    CONTINUE
      END



**********************************************************************
* THIS ROUTINE USES FINTIE DIFFERENCE TO CALCULATE THE X DERIVATIVES
* IT IS ONLY USED WHEN PERIODIC BOUNDARY CONDITIONS ARE INAPPROPRIATE
**********************************************************************

      SUBROUTINE DDJ(F,DFDL)
      IMPLICIT NONE
* PASSED VARIABLES
```

```
      REAL  F(0:31,1:50),DFDL(0:31,1:50)
      INTEGER SYM
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     $       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
* LOCAL VARIABLES
      INTEGER J,L,M,JM,JM2,JM3,JM4
      REAL DXDJ
* BEGIN DDL     USING FOURTH ORDER EVERYWHERE
      JM=NR-1
      JM2=NR-2
      JM3=NR-3
      JM4=NR-4
      DXDJ=1.0/(12.0*LX)
      DO 20 L=ZLO,ZHI
         DO 10 J=2,JM2
            DFDL(J,L)=F(J-2,L)-8.*F(J-1,L)+8.*F(J+1,L)-F(J+2,L)
   10       CONTINUE
     0   DFDL(0,L)=-25.*F(0,L)+48.*F(1,L)-36.*F(2,L)+
     116.*F(3,L)-3.*F(4,L)
         DFDL(1,L)=-3.*F(0,L)-10.*F(1,L)+18.*F(2,L)-6.*F(3,L)+F(4,L)
     0   DFDL(JM,L)=-F(JM4,L)+6.*F(JM3,L)-18.*F(JM2,L)+10.*F(JM,L)
     1              +3.*F(NR,L)
     0   DFDL(NR,L)=3.*F(JM4,L)-16.*F(JM3,L)+36.*F(JM2,L)-
     148.*F(JM,L)+25.*F(NR,L)
   20    CONTINUE
  120    CONTINUE
      DO 200 J=0,NR
      DO 200 L=ZLO,ZHI
         DFDL(J,L)=DFDL(J,L)*DXDJ
  200    CONTINUE
      END



**********************************************************************
* THIS ROUTINE CALCULATES X DERIVATIVES BY SPECTRAL METHODS
**********************************************************************
      SUBROUTINE DDX(SYM,A,DADX)
      IMPLICIT NONE
* GLOBAL VARIABLES
      COMPLEX DX(0:31)
      REAL W(0:31,1:50,8),TRIGS(451)
      INTEGER IFAX(13),INC,JUMP,NPTS,NVEC
      INTEGER SYMFLAG
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /FFT/W,SYMFLAG
     2       /GRID/DZDX,DZDL,DZDT,DX
* PASSED VARIABLES
      COMPLEX DADX(0:NC,NL),A(0:NC,NL)
      INTEGER SYM
* LOCAL VARIABLES
      INTEGER  J,L,M,L1,NX2,NR2
      COMPLEX B(0:255),C(0:255),D(0:31,1:50)

      GOTO (1,2,3,4,1)SYMFLAG
* ONE D
    3    DO 10 J=0,NC
```

```
          DO 10 L=1,NL
   10        DADX(J,L)=0.0
          GOTO 999
* NO SYMMETRY
     1    DO 110 L=ZLO,ZHI,2
             L1=L+1
             DO 120 J=0,NC
                B(J)=A(J,L)
                C(J)=A(J,L1)
  120           CONTINUE
             CALL FFT2(B,C,W,NX,1)
             DO 130 J=0,NC
                B(J)=B(J)*DX(J)
                C(J)=C(J)*DX(J)
  130           CONTINUE
             CALL FFT2(B,C,W,NX,-1)
             DO 140 J=0,NC
                DADX(J,L)=B(J)
                DADX(J,L1)=C(J)
  140           CONTINUE
  110        CONTINUE
          IF(SYMFLAG.EQ.5)THEN
             CALL DDJ(A,D)
             DO 150 L=ZLO,ZHI
             DO 160 J=0,1
                DADX(J,L)=D(J,L)
                DADX(NR-J,L)=D(NR-J,L)
  160           CONTINUE
  150           CONTINUE
             ENDIF
          GOTO 999
*EVEN SYMMETRY IN X
     2    NX2=2*NX
          NR2=NX2-1
          DO 210 L=ZLO,ZHI,2
             L1=L+1
             DO 220 J=0,NC
                B(J)=A(J,L)
                C(J)=A(J,L1)
  220           CONTINUE
             CALL EXTEND(B,NR,NR2,SYM)
             CALL EXTEND(C,NR,NR2,SYM)
             CALL FFT2(B,C,W,NX2,1)
             DO 230 J=XLO,XHI
                B(J)=B(J)*DX(J)
                C(J)=C(J)*DX(J)
  230           CONTINUE
             CALL FFT2(B,C,W,NX2,-1)
             DO 240 J=0,NC
                DADX(J,L)=B(J)
                DADX(J,L1)=C(J)
  240           CONTINUE
  210        CONTINUE
        GOTO 999
* USE FINITE DIFFERENCE IN X DIRECTION
     4 CALL DDJ(A,DADX)

   999 RETURN

       END
```

```
      SUBROUTINE EXTEND(B,N1,N2,SYM)
      IMPLICIT NONE
      INTEGER N1,N2,SYM,J
      REAL B(0:N2)
      DO 10 J=0,N1
         B(N2-J)=B(J)*SYM
10    CONTINUE
      RETURN
      END
```

```
*************************************************************************
* THIS SUBROUTINE OUTPUTS THE RESULTS
* PFLAG=1 CAUSES 1D IN Z TO BE OUTPUT
* 2 CAUSES  2D TABLE
*3 CAUSES  2D TABLE AND CONTOUR PLOT
*4 CAUSES  CONTOUR PLOT ONLY
*5 CAUSE 1D IN X
*************************************************************************

      SUBROUTINE OUTPUT(R,T,Z)

      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),T,Z(0:31,1:50)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
      REAL TMAX,TPRINT,PFLAG,TVIEW
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     2       /GRID/DZDX,DZDL,DZDT,DX
     3       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
* LOCAL VARIABLES
     0 REAL D(0:31,1:50),VX(0:31,1:50),VY(0:31,1:50),VZ(0:31,1:50),
     1      E(0:31,1:50),P(0:31,1:50),BX(0:31,1:50),BY(0:31,1:50),
     2      BZ(0:31,1:50),V(0:31,1:50),A(0:31,0:31),AZ(0:31,0:31)
      INTEGER J,L,K,M,NX2,FLAG3D,Q
      CHARACTER*40 TITLE

      FLAG3D=2
      IF(PFLAG.GE.2.)THEN
      DO 10 J=0,NR
      DO 10 L=1,NL
         D(J,L)=R(J,L,1)/DZDL(J,L)
         VX(J,L)=R(J,L,2)/R(J,L,1)
         VY(J,L)=R(J,L,3)/R(J,L,1)
         VZ(J,L)=R(J,L,4)/R(J,L,1)
         V(J,L)=SQRT(VX(J,L)*VX(J,L)+VZ(J,L)*VZ(J,L))
         E(J,L)=R(J,L,5)/(0.9D0*R(J,L,1))
C         P(J,L)=R(J,L,5)/DZDL(J,L)
         BX(J,L)=R(J,L,6)/DZDL(J,L)
         BY(J,L)=R(J,L,7)/DZDL(J,L)
         BZ(J,L)=R(J,L,8)/DZDL(J,L)
   10    CONTINUE
      ENDIF

      IF(PFLAG.GE.2.AND.PFLAG.LE.3)THEN
      DO 20 L=1,NL
         WRITE(6,2000)L
 2000    FORMAT('  ================>   L=',I4)
         WRITE(6,3000)'J','Z','DENSITY','VX','VY','VZ','TEMP',
     $       'BY','BZ'
 3000    FORMAT(1X,A2,1X,9(A8,1X))
         DO 30 J=XLO,XHI
            WRITE(6,4000)J,Z(J,L),D(J,L),VX(J,L),VY(J,L),
     $       VZ(J,L),E(J,L),BY(J,L),BZ(J,L)
 4000          FORMAT(1X,I3,1X,F8.4,1X,9(F8.4,1X))
   30       CONTINUE
   20    CONTINUE
      ENDIF
```

```
      IF ((PFLAG.GE.3..AND.PFLAG.LT.5.).OR.(PFLAG.EQ.9))THEN
         GOTO (1,2)FLAG3D
           TITLE='DENSITY'
           CALL MAKCON(D,Z,NX,NL,10,TITLE)
         PRINT *,'DENSITY AT T= ',T
         PRINT*,' '
C           TITLE='VX'
C           CALL MAKCON(VX,Z,NX,NL,10,TITLE)
C          PRINT *,'PLOT OF VX AT T= ',T
C           PRINT*,' '
C            TITLE='VZ'
C           CALL MAKCON(VZ,Z,NX,NL,10,TITLE)
C           PRINT *,'VZ AT T= ',T
C           PRINT*,' '
           PRINT *,'3D PLOTS AT T= ',T
           TITLE='PLOT OF ENERGY'
         CALL MAKCON(E,Z,NX,NL,10,TITLE)
         PRINT *,'ENERGY DENSITY AT T= ',T
         PRINT*,' '
           TITLE='SPEED'
         CALL MAKCON(V,Z,NX,NL,10,TITLE)
         PRINT *,'SPEED AT T= ',T
         PRINT*,' '
         GOTO 999
    1    CONTINUE
             TITLE='DENSITY'
             CALL THREED(D,NX-2,NL,TITLE)
             TITLE='TEMPERATURE'
             CALL THREED(E,NX-2,NL,TITLE)
             TITLE='SPEED'
             CALL THREED(V,NX-2,NL,TITLE)
         GOTO 999
    2   CONTINUE
         Q=0.4*NL-NC
         DO 40 J= 0,NR
         DO 40 K= 0,NR
   40    AZ(J,K)=Z(J,K+Q)
         DO 41 J= 0,NR
         DO 41 K= 0,NR
   41    A(J,K)=D(J,K+Q)
         TITLE='DENSITY'
       CALL MAKCON(A,AZ,NX,NX,10,TITLE)
       PRINT *,'PLOT OF DENSITY AT T= ',T
       PRINT*,' '
         DO 42 J= 0,NR
         DO 42 K= 0,NR
   42    A(J,K)=E(J,K+Q)
         TITLE='PLOT OF ENERGY'
       CALL MAKCON(A,AZ,NX,NX,10,TITLE)
       PRINT *,'TEMPERATURE AT T= ',T
       PRINT*,' '
         DO 43 J= 0,NR
         DO 43 K= 0,NR
   43    A(J,K)=V(J,K+Q)
         TITLE='SPEED'
       CALL MAKCON(A,AZ,NX,NX,10,TITLE)
       PRINT *,'SPEED AT T= ',T
       PRINT*,' '
  999 CONTINUE
       ENDIF

       IF((PFLAG.EQ.1).OR.(PFLAG.EQ.3.5))CALL OUTZ1D(R,Z,T)
```

```
      IF((PFLAG.EQ.5).OR.(PFLAG.EQ.9))CALL OUTX1D(R,Z,T)
      IF(PFLAG.EQ.25.)THEN
      DO 720 L=10,30
         WRITE(6,2000)L
         WRITE(6,3000)'J','Z','DENSITY','VX','VY','VZ','TEMP',
     $      'BY','BZ'
         DO 730 J=XLO,XHI
            WRITE(6,4000)J,Z(J,L),D(J,L),VX(J,L),VY(J,L),
     $      VZ(J,L),E(J,L),BY(J,L),BZ(J,L)
730         CONTINUE
720      CONTINUE
      ENDIF
      RETURN
      END
```

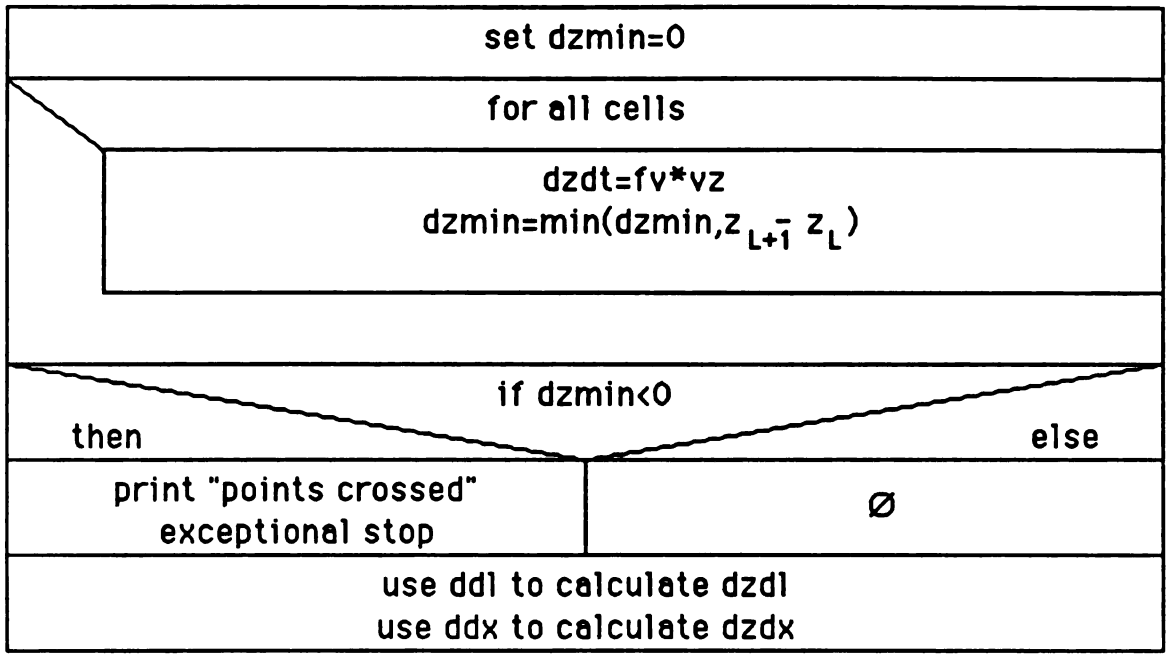
```
*****************************************************************
* THIS IS THE BOUNDARY CONDITION ROUTINE
*****************************************************************
      SUBROUTINE BC(R,Z,T)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),Z(0:31,1:50),T
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      REAL ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
      REAL WORK(0:31,1:50,8)
      INTEGER SYMFLAG
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /GRID/DZDX,DZDL,DZDT,DX
     2       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     3      /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
     4      /MAGNET/ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
     5      /FFT/WORK,SYMFLAG

* LOCAL VARIABLES
      INTEGER J,L,M
      REAL VPSTART,W,R1,E1,D,SIGN,A,U,MM,W0,U0
      REAL XFACE,ZFACE,SN,CSN
      CHARACTER*8 BCFLAG
*****************************************************************
* BCFLAG DETERMINES WHICH BC ARE USED
*****************************************************************
      BCFLAG='PISTONZ'
      IF(BCFLAG.EQ.'XPISLAB')THEN
         DO 5 J=XLO,XHI
            SN=DZDL(J,1)/DZDL(J,2)
            CSN=DZDL(J,NL)/DZDL(J,LM)
            DO 6 M=1,8
               R(J,1,M)=R(J,2,M)*SN
               R(J,NL,M)=R(J,LM,M)*CSN
    6          CONTINUE
    5       CONTINUE
         GOTO 999
         ENDIF
* BORING BC IN Z
       DO 20 J=XLO,XHI
          R(J,1,1)=R(J,2,1)*DZDL(J,1)/DZDL(J,2)
          R(J,NL,1)=R(J,LM,1)*DZDL(J,NL)/DZDL(J,LM)
          R(J,1,2)=R(J,2,2)*R(J,1,1)/R(J,2,1)
          R(J,NL,2)=R(J,LM,2)*R(J,NL,1)/R(J,LM,1)
          R(J,1,4)=-R(J,2,4)*R(J,1,1)/R(J,2,1)
          R(J,NL,4)=-R(J,LM,4)*R(J,NL,1)/R(J,LM,1)
          R(J,1,5)=R(J,2,5)*R(J,1,1)/R(J,2,1)
          R(J,NL,5)=R(J,LM,5)*R(J,NL,1)/R(J,LM,1)
   20 CONTINUE

C      IF(T.LT.6.2831853/OM)THEN
* FOR MAGNETOACOUSTIC WAVES
C      DO 40 L=1,NL
C         R(1,L,1)=R0*(1.+W/A)*DZDL(1,L)
C         R(1,L,4)=-ALFZ*ALFX/(A*A-ALFZ*ALFZ)*W*R(1,L,1)
C         R(1,L,3)=0.
```

```
C          R(1,L,2)=W*R(1,L,1)
C          R(1,L,5)=E0*(1.0+GM1*W/A)*R(1,L,1)
C          R(1,L,8)=BZ0*(1.0+A*W/(A*A-ALFZ*ALFZ))*DZDL(1,L)
C          R(1,L,7)=0.
C          R(1,L,6)=BX0*DZDL(1,L)
C   40     CONTINUE
C       ELSE
C       XLO=0
C        XHI=NR
C       ENDIF
       IF(BCFLAG.EQ.'PISTONZ')THEN
* BC FOR Z PISTON
          IF(SYMFLAG.GE.4)THEN
             DO 14 L=1,NL
             DO 12 M=1,8
                R(0,L,M)=R(1,L,M)*DZDL(0,L)/DZDL(1,L)
C 2*R(1,L,M)-R(2,L,M)
                R(NR,L,M)=R(NR-1,L,M)*DZDL(NR,L)/DZDL(NR-1,L)
C 2*R(NR-1,L,M)-R(NR-2,L,M)
   12           CONTINUE
               R(0,L,2)=0.5*(R(0,L,2)-R(NR,L,2))
               R(NR,L,2)=-R(0,L,2)
   14          CONTINUE
             ENDIF
          DO 10 J=XLO,XHI
             R(J,1,1)=R(J,2,1)*DZDL(J,1)/DZDL(J,2)
             R(J,1,2)=R(J,2,2)*DZDL(J,1)/DZDL(J,2)
C             R(J,1,2)=-0.05*R(J,1,1)
             R(J,1,4)=VP*R(J,1,1)
C             R(J,1,4)=2*R(J,1,1)*VP-R(J,2,4)*DZDL(J,1)/DZDL(J,2)
           R(J,1,5)=R(J,2,5)*DZDL(J,1)/DZDL(J,2)
             R(J,1,6)=R(J,2,6)*DZDL(J,1)/DZDL(J,2)
             R(J,1,8)=R(J,2,8)*DZDL(J,1)/DZDL(J,2)
             R(J,NL,4)=0.0
   10        CONTINUE
          GOTO 999
          ENDIF

       IF(BCFLAG.EQ.'ZPISLAB')THEN
* BC FOR Z PISTON IN SLAB
          DO 100 J=XLO,XHI
             R(J,1,1)=R(J,2,1)*DZDL(J,1)/DZDL(J,2)
             R(J,1,2)=R(J,2,2)*DZDL(J,1)/DZDL(J,2)
             R(J,1,4)=0.0
C             R(J,1,4)=2*R(J,1,1)*VP-R(J,2,4)*DZDL(J,1)/DZDL(J,2)
           R(J,1,5)=R(J,2,5)*DZDL(J,1)/DZDL(J,2)
             R(J,1,6)=R(J,2,6)*DZDL(J,1)/DZDL(J,2)
             R(J,1,8)=R(J,2,8)*DZDL(J,1)/DZDL(J,2)
             R(J,NL,4)=0.0
  100     CONTINUE
          DO 110 J=XLO,XHI
C SPLIT+1,NX-SPLIT-1
             R(J,1,4)=VP*R(J,1,1)
  110        CONTINUE
          GOTO 999
          ENDIF
* FOR ALFVEN WAVE
C       DO 20 J=XLO,XHI
C       R(J,1,1)=R(J,2,1)*DZDL(J,1)/DZDL(J,2)
C       R(J,1,2)=R(J,1,1)*W
C       R(J,1,4)=0.
C      R(J,1,5)=R(J,2,5)*DZDL(J,1)/DZDL(J,2)
```

```
C       R(J,1,6)=(BX0-BZ0*W/ALF)*DZDL(J,1)
C       R(J,NL,4)=0.
C   20    CONTINUE* SURFACE WAVES IN THE X DIRECTION
        IF(BCFLAG.EQ.'SURFACEX')THEN
           IF(2.0*T*OM.GT.6.2831853)THEN
              SN=0.0
              CSN=0.0
              XLO=1
              XHI=NC
           ELSE
              SN=SIN(OM*T)
              CSN=COS(OM*T)
              XLO=0
              XHI=NR
           ENDIF
           J=1
C           ZFACE=0.5*(Z(J,SPLIT)+Z(J,SPLIT+1))
           ZFACE=Z(J,SPLIT)
* THE FIELD-FREE SIDE OF THE INTERFACE
           W0=ME*CSURF*VP/((1.0-CSURF**2)*OM)*SN
           U0=VP*CSN
           DO 200 L=1,SPLIT-1
              W=W0*EXP(ME*(Z(J,L)-ZFACE))
              U=U0*EXP(ME*(Z(J,L)-ZFACE))
              R(J,L,1)=RM0*(1.0+CSURF*W)*DZDL(J,L)
              R(J,L,4)=R(J,L,1)*U
              R(J,L,2)=R(J,L,1)*W
              R(J,L,5)=(RM0*E0+RM0*CSURF*W/GM1)*DZDL(J,L)
              R(J,L,6)=0.0
C (-MM*IW/OM)*BX0*DZDL(J,L)
              R(J,L,8)=0.0
C (BZ0+BX0*W/CSURF)*DZDL(J,L)
  200      CONTINUE
* ON THE INTERFACE
           R(J,SPLIT,2)=0.0
           R(J,SPLIT,4)=U0*R(J,SPLIT,1)
           R(J,SPLIT,8)=(0.5*BX0*U0/CSURF)*DZDL(J,SPLIT)
* MAGNETIC SIDE OF INTERFACE
C           W0=-CSURF*MO*VP/((1-CSURF**2)*OM)*SN
           W0=-MO*CSURF*VP/((1.0-CSURF**2)*OM)*SN
           DO 210 L=SPLIT+1,NL
              W=W0*EXP(-MO*(Z(J,L)-ZFACE))
              U=U0*EXP(-MO*(Z(J,L)-ZFACE))
              R(J,L,1)=R0*(1.0+CSURF*W)*DZDL(J,L)
              R(J,L,4)=R(J,L,1)*U
              R(J,L,2)=R(J,L,1)*W
              R(J,L,5)=(R0*E0+R0*CSURF*W/GM1)*DZDL(J,L)
              R(J,L,6)=BX0*(1.0-(1-CSURF**2)*W/CSURF)*DZDL(J,L)
              R(J,L,8)=(-BX0*U/CSURF)*DZDL(J,L)
  210      CONTINUE
           GOTO 999
        ENDIF
* SURFACE WAVES IN THE +Z DIRECTION
        IF(BCFLAG.EQ.'SURFACEZ')THEN
           IF(T*OM.GT.6.2831853)THEN
              SN=0.0
              CSN=1.0
           ELSE
              SN=SIN(OM*T)
              CSN=COS(OM*T)
           ENDIF
           L=1
```

```
          XFACE=LX*SPLIT+0.5*LX
*  THE FIELD-FREE SIDE OF THE INTERFACE
          W0=ME*CSURF*VP/((1.0-CSURF**2)*OM)*SN
          U0=VP*CSN
          DO 300 J=0,SPLIT
             W=W0*EXP(ME*(LX*J-XFACE))
             U=U0*EXP(ME*(LX*J-XFACE))
             R(J,L,1)=RM0*(1.0-CSURF*W)*DZDL(J,L)
             R(J,L,2)=R(J,L,1)*U
             R(J,L,4)=R(J,L,1)*W
             R(J,L,5)=(RM0*E0-RM0*CSURF*W/GM1)*DZDL(J,L)
             R(J,L,6)=0.0
C (-MM*IW/OM)*BX0*DZDL(J,L)
             R(J,L,8)=0.0
C (BZ0+BX0*W/CSURF)*DZDL(J,L)
  300     CONTINUE
*  MAGNETIC SIDE OF INTERFACE
          W0=-MO*CSURF*VP/((1.0-CSURF**2)*OM)*SN
          DO 310 J=SPLIT+1,NR
             W=W0*EXP(-MO*(LX*J-XFACE))
             U=U0*EXP(-MO*(LX*J-XFACE))
             R(J,L,1)=R0*(1.0-CSURF*W)*DZDL(J,L)
             R(J,L,2)=R(J,L,1)*U
             R(J,L,4)=R(J,L,1)*W
             R(J,L,5)=(R0*E0-R0*CSURF*W/GM1)*DZDL(J,L)
             R(J,L,8)=BZ0*(1.0-(1-CSURF**2)*W/CSURF)*DZDL(J,L)
             R(J,L,6)=(-BZ0*U/CSURF)*DZDL(J,L)
  310     CONTINUE
          GOTO 999
        ENDIF
        IF(BCFLAG.EQ.'JPISTON')THEN
*  BC FOR Z PISTON
          DO 410 L=1,NL
             R(0,L,1)=R(1,L,1)*DZDL(0,L)/DZDL(1,L)
             R(0,L,4)=R(1,L,4)*DZDL(0,L)/DZDL(1,L)
C             R(1,L,2)=-0.05*R(1,L,1)
             R(0,L,2)=VP*R(0,L,1)
            R(0,L,5)=R(1,L,5)*DZDL(0,L)/DZDL(1,L)
             R(0,L,6)=R(1,L,6)*DZDL(0,L)/DZDL(1,L)
             R(0,L,8)=R(1,L,8)*DZDL(0,L)/DZDL(1,L)
             R(NR,L,2)=0.0
  410        CONTINUE
          GOTO 999
          ENDIF


*  PERIODIC (IN Z) BC
C       DO 100 M=1,8
C       DO 100 J=XLO,XHI
C          R(J,1,M)=R(J,LM3,M)*DZDL(J,1)/DZDL(J,LM3)
C          R(J,2,M)=R(J,LM2,M)*DZDL(J,2)/DZDL(J,LM2)
C          R(J,LM,M)=R(J,3,M)*DZDL(J,LM)/DZDL(J,3)
C          R(J,NL,M)=R(J,4,M)*DZDL(J,NL)/DZDL(J,4)
C  100     CONTINUE
  999 CONTINUE
*  THE END OF THE CASE STRUCTURE
        RETURN
        END


*************************************************************************
*  THE ROUTINES TO SMOOTH THE X 1 CELL VARIATIONS
```

```
*************************************************************************
      SUBROUTINE SMOOTH(R)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /GRID/DZDX,DZDL,DZDT,DX
* LOCAL VARIABLES
      REAL A(0:31,1:50)
      INTEGER M,J,L
* BEGIN SMOOTH
      DO 10 M=1,8
         DO 20 J=XLO,XHI
         DO 20 L=1,NL
            A(J,L)=R(J,L,M)/DZDL(J,L)
   20       CONTINUE
         CALL SMOOTH1(A)
         DO 30 J=XLO,XHI
         DO 30 L=1,NL
            R(J,L,M)=A(J,L)*DZDL(J,L)
   30    CONTINUE
   10    CONTINUE
      RETURN
      END


*************************************************************************
      SUBROUTINE SMOOTH1(V)
      IMPLICIT NONE
*GLOBAL VARIABLES
      REAL WORK(0:31,1:50,8),TRIGS(451)
      INTEGER IFAX(13),NPTS,INC,JUMP,NVEC
      INTEGER SYMFLAG
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      COMMON /FFT/WORK,SYMFLAG
     $       /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
* PASSED VARIABLES
       COMPLEX V(0:NC,NL)
* LOCAL VARIABLES
      INTEGER L,M,ISIGN,NVEC3
C          CALL FFT99(V,WORK,TRIGS,IFAX,INC,JUMP,NPTS,NVEC3,ISIGN)
         DO 50 L=1,NL
   50       V(NC,L)=0.0
C          CALL FFT99(V,WORK,TRIGS,IFAX,INC,JUMP,NPTS,NVEC3,ISIGN)
      RETURN
      END
```




```
************************************************************************
* THE GRID ROUTINE CONTROLS THE MOTION OF THE GRID.
* DZDL,DZDX, & DZDT ARE CALCULATED HERE
************************************************************************
      SUBROUTINE GRID(R,Z,T,DT)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),Z(0:31,1:50),T,DT
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     2       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     3       /GRID/DZDX,DZDL,DZDT,DX
* LOCAL VARIABLES
      INTEGER J,L,M
      REAL DZ1,DZMIN,A(0:31,1:50)

* BEGIN GRID
      DZMIN=LZ

      DO 10 J=XLO,XHI
         DZDT(J,1)=R(J,1,4)/R(J,1,1)
         DZDT(J,2)=FV*R(J,2,4)/R(J,2,1)
         DZDT(J,LM)=FV*R(J,LM,4)/R(J,LM,1)
         DZDT(J,NL)=1.*R(J,NL,4)/R(J,NL,1)

         DO 20 L=3,LM2
            DZDT(J,L)=FV*R(J,L,4)/R(J,L,1)
            DZ1=Z(J,L+1)-Z(J,L)
            DZMIN=AMIN1(DZMIN,DZ1)
   20       CONTINUE
   10    CONTINUE
      IF(DZMIN.LE.0.)THEN
```

```
         PRINT *,'   POINTS CROSSED AT T=',T
         CALL OUTPUT(R,T,Z)
         STOP
         ENDIF
      CALL DDL(Z,A)
      DO 30 J=XLO,XHI
      DO 30 L=ZLO,ZHI
      DZDL(J,L)=A(J,L)
   30CONTINUE

      CALL DDX(1,Z,A)
      DO 40 J=XLO,XHI
      DO 40 L=ZLO,ZHI
      DZDX(J,L)=A(J,L)/DZDL(J,L)
   40CONTINUE

      IF(DT.LT.1E-7)THEN
         PRINT *,'DT TOO SMALL AT DT=',DT
      CALL OUTPUT(R,T,Z)
         STOP
         ENDIF
      RETURN
      END
*************************************************************************
      SUBROUTINE OUTX1D(R,Z,TIME)
      COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     $       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     $       /GRID/DZDX,DZDL,DZDT,DX
     $       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     $       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      INTEGER I,J,V,VMAX,VAXIS,L,M,NX,NL,NR,NC,LM,LM2,LM3,LM4,
     $        VD,VV,VT,XLO,XHI,ZLO,ZHI,SPLIT
      REAL R(0:31,1:50,8),MIND,MAXD,DY,OVER8PI,DZDL(0:31,1:50),
     $     DZDX(0:31,1:50),D(0:31),VX(0:31),T(0:31),MAXVZ,MINVZ,
     $     MAXT,MINT,Z(0:31,1:50),DD,DVZ,DTT,DZDT(0:31,1:50),
     $     TMAX,TPRINT,PFLAG,TVIEW,VY,VZ,BX,BY,BZ
      COMPLEX DX(0:31)

      PRINT *,'      '
      PRINT *,'@FTF'
      PRINT *,NR,'  NUMBER OF POINTS FOR PLOTTER'
      WRITE(6,1000)TIME
 1000 FORMAT('  TIME= ',F11.4,F10.5)

      L=NL/2
      WRITE(*,2000)'J','X','DENSITY','VX','VZ','TEMP','BX',
     $               'BZ','VY','BY'
      WRITE(*,3100) ' 0 ',0.,1.,0.,0.,1.,0.,0.,0.,0.
 3100 FORMAT(1X,A4,1X,9(F7.4,1X))
 2000 FORMAT(1X,A4,1X,9(A7,1X))
      DO 20 J=XLO,XHI
         D(J)=R(J,L,1)/DZDL(J,L)
         VX(J)=R(J,L,2)/R(J,L,1)
         VY=R(J,L,3)/R(J,L,1)
         VZ=R(J,L,4)/R(J,L,1)
         T(J)=R(J,L,5)/R(J,L,1)/0.9D0
         BX=R(J,L,6)/DZDL(J,L)
         BY=R(J,L,7)/DZDL(J,L)
         BZ=R(J,L,8)/DZDL(J,L)
         WRITE(*,3000)J,J*LX,D(J),VX(J),VZ,T(J),BX,BZ,VY,BY
 3000      FORMAT(1X,I4,1X,F8.4,1X,9(F7.4,1X))
```

```
 20   CONTINUE
      RETURN
      END


      SUBROUTINE OUTZ1D(R,Z,TIME)
      COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     $       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     $       /GRID/DZDX,DZDL,DZDT,DX
     $       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     $       /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      CHARACTER*1,A(-1:132)
      INTEGER I,J,V,VMAX,VAXIS,L,M,NX,NL,NR,NC,LM,LM2,LM3,LM4,
     $        VD,VV,VT
      REAL R(0:31,1:50,8),MIND,MAXD,DY,OVER8PI,DZDL(0:31,1:50),
     $     DZDX(0:31,1:50),D(1:50),VZ(1:50),T(1:50),MAXVZ,MINVZ,
     $     MAXT,MINT,Z(0:31,1:50),DD,DVZ,DTT,DZDT(0:31,1:50),VY,BY
     $     TMAX,TPRINT,PFLAG,TVIEW
      COMPLEX DX(0:31)

      PRINT *,'       '
      PRINT *,'@FTF'
      PRINT *,NL,'  NUMBER OF POINTS FOR PLOTTER'
      WRITE(6,1000)TIME
1000  FORMAT('  TIME= ',F11.4,F10.5)

      J=NC
      WRITE(*,2000)'L','Z','DENSITY','VX','VZ','TEMP',
     $   'BX','BZ','VY','BY'
      WRITE(*,3100) '-0',0.,1.,0.,0.,1.,BX0,0.0
3100  FORMAT(1X,A4,2X,9(F7.4,1X),1A25)
2000  FORMAT(1X,A4,2X,9(A7,1X))
      DO 20 L=1,NL
         D(L)=R(J,L,1)/DZDL(J,L)
         VX=R(J,L,2)/R(J,L,1)
         VY=R(J,L,3)/R(J,L,1)
         VZ(L)=R(J,L,4)/R(J,L,1)
         T(L)=R(J,L,5)/R(J,L,1)/0.9D0
         BX=R(J,L,6)/DZDL(J,L)
         BY=R(J,L,7)/DZDL(J,L)
       BZ=R(J,L,8)/DZDL(J,L)
         WRITE(*,3000)L,Z(J,L),D(L),VX,VZ(L),T(L),BX,BZ,VY,BY
3000       FORMAT(1X,I4,2X,F8.4,1X,9(F7.4,1X))
 20    CONTINUE
       RETURN

      END

      LOGICAL FUNCTION CHECKDT(R,R1,R2,T,DT)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),R1(0:31,1:50,8),R2(0:31,1:50,8),T,DT
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      REAL TMAX,TPRINT,PFLAG,TVIEW
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     2 /PRINTS/TMAX,TPRINT,PFLAG,TVIEW

* LOCAL VARIABLES
```

```
      REAL EPD,EPE,EPV,DLIM,ELIM,VLIM,CRIT,DTMAX,DTVIEW,CURRENT
      INTEGER J,L,M,MSFLAG
      DATA DLIM,ELIM,VLIM,DTMAX/.00001,.00001,.00001,2.0/
      EPD=0.
      EPE=0.
      EPV=0.
      MSFLAG=0
      DTVIEW=-DT+TVIEW*0.5
      IF (DTVIEW.LE.0.0)THEN
      DTVIEW=DTMAX
      ENDIF
      DO 10 J=1,NR-1
      DO 10 L=2,LM2
         EPD=AMAX1(EPD,ABS((R1(J,L,1)-R2(J,L,1))/R2(J,L,1)))
         EPE=AMAX1(EPE,ABS((R1(J,L,5)-R2(J,L,5))/R2(J,L,5)))
         EPV=AMAX1(EPV,ABS((R1(J,L,4)-R2(J,L,4))/R2(J,L,1)))
   10 CONTINUE
      CRIT=AMAX1(EPD/DLIM,EPE/ELIM,EPV/VLIM)
      IF(CRIT.GE.1.)THEN
         CHECKDT=.TRUE.
         DT=.9*DT*(CRIT**(-0.25))
         PRINT *,'REPEATING STEP AT T=',T,'    DT=',DT
      ELSEIF(CRIT.GT.0.)THEN
         CHECKDT=.FALSE.
      CURRENT=T+2.*DT
         DT=AMIN1(.8*DT*(CRIT**(-0.2)),DTMAX,DTVIEW)
      IF(MSFLAG.NE.0)PRINT *,'SETTING DT=',DT,'  AT T=',CURRENT
      ELSE
        CHECKDT=.FALSE.
      ENDIF
      RETURN
      END
*************************************************************************
* THIS SUBROUITNE CALCULATES THE WAVES SPEEDS
*************************************************************************

     SUBROUTINE SOMMER(R,NEWR,DT,Z,NEWZ)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),NEWR(0:31,1:50,8),NEWZ(0:31,1:50),DT
      REAL Z(0:31,1:50)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL C(0:31,1:50,8)
     0COMMON   /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1/GRID/DZDX,DZDL,DZDT,DX
     2  /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     3/SOMMER/C
* LOCAL VARIABLES
      INTEGER J,L,M
      REAL A(0:31,1:50),DADL(0:31,1:50),Q
* BEGIN SOMMER
      DO 10 J=XLO,XHI
      NEWR(J,NL,1)=R(J,LM,1)*DZDL(J,NL)/DZDL(J,LM)
   10CONTINUE
      DO 20 M=6,8
      DO 20 J=XLO,XHI
     0 C(J,LM,M)=AMIN1(LZ/DT,AMAX1(1.0,-(NEWR(J,LM,M)/DZDL(J,LM)-
     1R(J,LM,M)/DZDL(J,LM))*(Z(J,NL)-Z(J,LM2))/
```

```
     2(R(J,NL,M)/DZDL(J,NL)-R(J,LM2,M)/DZDL(J,LM2)+1E-20)))
   20CONTINUE
      DO 30 M=2,5
      DO 30 J=XLO,XHI
      C(J,LM,M)=AMIN1(LZ/DT,AMAX1(1.0,-(NEWR(J,LM,M)/NEWR(J,LM,1)-
     1R(J,LM,M)/R(J,LM,1))/DT*(Z(J,NL)-Z(J,LM2))/
     2(R(J,NL,M)/R(J,NL,1)-R(J,LM2,M)/R(J,LM2,1)+1E-20)))
   30 CONTINUE
      DO 40 M=6,8
      DO 40 J=XLO,XHI
      Q=C(J,LM,M)*DT/(Z(J,NL)-Z(J,LM))
      NEWR(J,NL,M)=R(J,LM,M)*DZDL(J,NL)/DZDL(J,LM)*Q+(1.-Q)*R(J,NL,M)
   40CONTINUE
      DO 50 M=2,5
     DO 60 J=XLO,XHI
      Q=C(J,LM,M)*DT/(Z(J,NL)-Z(J,LM))
      NEWR(J,NL,M)=R(J,LM,M)*R(J,NL,1)/R(J,LM,1)*Q+(1.-Q)*R(J,NL,M)
   60CONTINUE
   50 CONTINUE
      RETURN
      END
*****************************************************************
*ALL OF THE INITIALIZATION ROUTINES FOLLOW
*****************************************************************
* THIS SUBROUTINE INITIALIZES THE PARAMETERS AND VARIABLES
*****************************************************************
      SUBROUTINE INIT(R,Z,DT)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8),Z(0:31,1:50),DT
* GLOBAL VARIABLES
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      REAL ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     0COMMON   /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1         /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
     2       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     3         /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     4         /MAGNET/ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
* LOCAL VARIABLES
      REAL T,RATIO,W,E1,R1,PM0,PM,S,Q0,RADIUS
      INTEGER J,L,MIDJ,MIDL,SPLIT2
      CHARACTER*8 INITFLAG
      INITFLAG='Y TUBE'
      WRITE(*,1001)INITFLAG
 1001 FORMAT('1 GMHDI  INITIAL CONDITIONS= ',A8)
* ==============================================================
* SET PARAMETERS
      CALL INITNUM
      CALL INITZER
      CALL INITPAR
      CALL SETCON
      IF((BX0+BY0+BZ0.NE.0.0))CALL INITMAG
* ---------------------------------------------------
* SET UP INITIAL ATMOSPHERE
* ---------------------------------------------------

      DO 10 J=0,NR
      DO 10 L=1,NL
```

```
      R(J,L,1)=R0
        R(J,L,2)=VX0
        R(J,L,3)=VY0
        R(J,L,4)=VZ0
      R(J,L,5)=E0
        R(J,L,6)=BX0
        R(J,L,7)=BY0
        R(J,L,8)=BZ0
   10   CONTINUE
* SPECIAL INIT CONDITIONS
      IF (INITFLAG.EQ.'Z TUBE')THEN
* Shock tube initial conditions
         RATIO=0.02
         PRINT *,'SHOCK TUBE WITH INITIAL PRESSURE RATIO 0F ',RATIO
         SPLIT=NL*.5
         J=NC
         DO 30 J=XLO,XHI
         DO 20 L=1,SPLIT
            R(J,L,1)=R0*RATIO
            R(J,L,5)=E0*RATIO
   20       CONTINUE
         R(J,SPLIT,1)=R0*.5*(RATIO+1)
         R(J,SPLIT,5)=E0*.5*(RATIO+1)
   30    CONTINUE
      GOTO 999
      ENDIF
      IF (INITFLAG.EQ.'X FACE')THEN
* x magnetic interface
         split=nl/2
         PM0=(BX0*BX0+BY0*BY0+BZ0*BZ0)*OVER8PI
         RM0=R0*(1+PM0/(GM1*E0))
         DO 150 J=XLO,XHI
         DO 140 L=1,SPLIT
            R(J,L,6)=0.0
            R(J,L,7)=0.0
  140    CONTINUE
           R(J,SPLIT,6)=0.5*(R(J,SPLIT-1,6)+R(J,SPLIT+1,6))
           R(J,SPLIT,7)=0.5*(R(J,SPLIT-1,7)+R(J,SPLIT+1,7))
           R(J,SPLIT,8)=0.5*(R(J,SPLIT-1,8)+R(J,SPLIT+1,8))
  150    CONTINUE
        DO 160 J=XLO,XHI
        DO 160 L=1,SPLIT
           PM=PM0-(R(J,L,6)**2+R(J,L,8)**2)*OVER8PI
           R(J,L,1)=R0+PM/(GM1*E0)
           R(J,L,5)=R(J,L,1)*E0
  160      CONTINUE
      GOTO 999
      ENDIF
      IF (INITFLAG.EQ.'Z FACE')THEN
* x magnetic interface
         split=NC
         PM=(BX0*BX0+BY0*BY0+BZ0*BZ0)*OVER8PI
         RM0=R0*(1+PM/(GM1*E0))
         DO 250 J=0,SPLIT
         DO 240 L=1,NL
            R(J,L,6)=0.0
            R(J,L,7)=0.0
            R(J,L,8)=0.0
C            R(J,L,5)=E0+R0*PM/GM1
            R(J,L,1)=RM0
            R(J,L,5)=E0*R(J,L,1)
  240    CONTINUE
```

```
  250    CONTINUE
      GOTO 999
      ENDIF
      IF (INITFLAG.EQ.'X SLAB')THEN
* x magnetic interface
         split=NL
         PM0=(BX0*BX0+BY0*BY0+BZ0*BZ0)*OVER8PI
         RM0=R0*(1+PM0/(GM1*E0))
         DO 350 J=XLO,XHI
         DO 340 L=1,SPLIT
            R(J,L,6)=0.0
            R(J,L,7)=0.0
  340    CONTINUE
         SPLIT2=NL-SPLIT+1
         DO 345 L=SPLIT2,NL
            R(J,L,6)=0.0
            R(J,L,7)=0.0
  345    CONTINUE
           R(J,SPLIT,6)=0.5*(R(J,SPLIT-1,6)+R(J,SPLIT+1,6))
           R(J,SPLIT2,6)=0.5*(R(J,SPLIT2-1,6)+R(J,SPLIT2+1,6))
  350    CONTINUE
        DO 360 J=XLO,XHI
        DO 360 L=1,NL
           PM=PM0-(R(J,L,6)**2+R(J,L,8)**2)*OVER8PI
           R(J,L,1)=R0+PM/(GM1*E0)
           R(J,L,5)=R(J,L,1)*E0
  360      CONTINUE
* TO PRODUCE SHOCK IN X DIRECTION
        DO 370 L=1,NL
        DO 370 J=0,NC
           R(J,L,2)=-VP*R(J,L,1)
           R(NR-J,L,2)=VP*R(NR-J,L,1)
  370   CONTINUE
      GOTO 999
      ENDIF

      IF (INITFLAG.EQ.'Z SLAB')THEN
* x magnetic interface
         split=NC-4
         PM0=(BX0*BX0+BY0*BY0+BZ0*BZ0)*OVER8PI
         RM0=R0*(1+PM0/(GM1*E0))
         DO 450 L=1,NL
         DO 440 J=0,SPLIT
            R(J,L,8)=0.0
  440    CONTINUE
         SPLIT2=NX-SPLIT
         DO 445 J=SPLIT2,NR
            R(J,L,8)=0.0
  445    CONTINUE
           R(SPLIT2,L,8)=0.5*(R(SPLIT2-1,L,8)+R(SPLIT2+1,L,8))
           R(SPLIT,L,8)=0.5*(R(SPLIT-1,L,8)+R(SPLIT+1,L,8))
  450    CONTINUE
        DO 460 J=XLO,XHI
        DO 460 L=1,NL
           PM=PM0-(R(J,L,6)**2+R(J,L,8)**2)*OVER8PI
           R(J,L,1)=R0+PM/(GM1*E0)
           R(J,L,5)=R(J,L,1)*E0
  460      CONTINUE
      GOTO 999
      ENDIF
      IF(INITFLAG.EQ.'TEST')THEN
* TO PRODUCE Z SHOCK
```

```
      DO 495 J=XLO,XHI
      DO 490 L=1,15
  490 R(J,L,4)=VP*R(J,L,1)
      DO 495 L=15,NL
  495 R(J,L,4)=-VP*R(J,L,1)
      GOTO 999
      ENDIF
      IF(INITFLAG.EQ.'Y TUBE')THEN
* A CYLINDRICAL FLUX TUBE EXTENDING IN THE Y DIRECTION
        S=0.50
        RADIUS=4.0
        MIDJ=NC
        MIDL=NL*.4
        PM0=(BX0*BX0+BY0*BY0+BZ0*BZ0)*OVER8PI
        WRITE(*,1000)MIDJ,MIDL,RADIUS,S
 1000   FORMAT('  CENTER= ',I3,',',I3,' RADIUS= ',F6.3,
     $     ' THICKNESS=', F7.4)
        DO 570 L=1,NL
        DO 560 J=XLO,XHI
*HYPERBOLIC TANGENT IN r
          Q0=SQRT(((J-MIDJ)*LX)**2+((L-MIDL)*LZ)**2)
          R(J,L,7)=0.5*BY0*(1.0+TANH((RADIUS-Q0)/S))
C          PRINT *,J,L,Q0,R(J,L,7)
  560   CONTINUE
  570   CONTINUE
        DO 580 J=XLO,XHI
        DO 580 L=1,NL
          PM=PM0-(R(J,L,6)**2+R(J,L,7)**2+R(J,L,8)**2)*OVER8PI
          R(J,L,1)=R0+PM/(GM1*E0)
          R(J,L,5)=R(J,L,1)*E0
  580     CONTINUE
      GOTO 999
      ENDIF
  999 CONTINUE
      CALL SETGRID(R,Z,DT)
      CALL INITFFT(R)
      RETURN
      END

************************************************************
      SUBROUTINE INITNUM
      IMPLICIT NONE
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      INTEGER FXSYM(8)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1         /SYMS/FXSYM

      NX=32
      NC=(NX-2)/2
      NR=NX-1
      NL=50
      PRINT *,'NX= ',NX,'   NL= ',NL
      LM=NL-1
      LM2=NL-2
      LM3=NL-3
      LM4=NL-4
      XLO=0
      XHI=NR
      ZLO=1
      ZHI=NL
      FXSYM(1)=-1
```

```
      FXSYM(2)=+1
      FXSYM(3)=-1
      FXSYM(4)=-1
      FXSYM(5)=-1
      FXSYM(6)=-1
      FXSYM(7)=+1
      FXSYM(8)=+1
      RETURN
      END

*************************************************************************
      SUBROUTINE INITZER
      IMPLICIT NONE
* GLOBAL VARIABLES
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      COMMON /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0

      R0=1.0
      VX0=0.0
      VY0=0.0
      VZ0=0.0
      E0=0.9
      BX0=0.0
      BY0=3.0
      BZ0=0.0
      RETURN
      END

*************************************************************************
      SUBROUTINE INITPAR
      IMPLICIT NONE
* GLOBAL VARIABLES
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      REAL TMAX,TPRINT,PFLAG,TVIEW
     0COMMON /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     1       /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
     2       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW

      GM1=2.0/3.0
      G=0.0
      QH=G*R0/(GM1*E0)
  IF(G.EQ.0.)THEN
      PRINT *,' No Gravity'
      ELSE
      PRINT *,'G= ',G,'SCALE HEIGHT= ',1./QH
      ENDIF
C      READ (*,*)VP
      VP=0.1
      OM=0.0
      FV=1.0
      WRITE(*,900)VP,OM,FV
  900 FORMAT(1X,'VP= ',F10.5,'   OM= ',F10.5,'   FV= ',F10.5)
      ZETA2=9*(.05+VP)
      ZETA1=ZETA2/3
      IF(ZETA1+ZETA2.EQ.0.)THEN
      PRINT *, 'No Viscosity'
      ELSE
        WRITE(*,1000)ZETA1,ZETA2/3.
 1000   FORMAT(1X,'Zeta1= ',F6.4,'   Zeta2= ',F6.4)
      ENDIF
      ETA0=0.0
```

```
      IF(ETA0.EQ.0.)THEN
      PRINT *,'Infinite Conductivity'
      ELSE
      PRINT *,'Eta=',ETA0
      ENDIF
      ERF=0.
      LX=1.5
      LY=1.0
      LZ=1.0
      PRINT *,'LX= ',LX,'   LZ= ',LZ

      TMAX=50.0
      tprint=10.0
      PFLAG=25.0
* REMEMBER TO CHANGE FLAG3D
      PRINT *,'TMAX= ',TMAX,'   TPRINT= ',TPRINT
C      TVIEW=TPRINT
C      TVIEW=0.
      TVIEW=30.
* SETTING TVIEW=0.0 CAUSES THE INITIAL CONDITONS TO BE PRINTED
* WHILE   TVIEW=TPRINT CAUSES IT TO SKIP THE FIRST PRINT

      RETURN
      END

*************************************************************************
      SUBROUTINE SETCON
      IMPLICIT NONE
* GLOBAL VARIABLES
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      COMMON /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
      PI=4.*ATAN(1.0)
      FOURPI=4.*PI
      OVER8PI=.125/PI
      F16D15=16.D0/15.D0
      F1D15=1.D0/15.D0
      RETURN
      END

*************************************************************************
      SUBROUTINE INITMAG
      IMPLICIT NONE
* GLOBAL VARIABLES
      REAL ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      REAL BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     0COMMON /MAGNET/ALF,ALFX,ALFY,ALFZ,BETA,CFM,CSM,CT,CSURF,ME,MO
     1       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     2       /ZEROES/BX0,BY0,BZ0,E0,R0,VX0,VY0,VZ0
     3       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      BETA=1E20
      IF(BZ0.NE.0..OR.BX0.NE.0..OR.BY0.NE.0.)
     $   BETA=.6/((BZ0*BZ0+BY0*BY0+BX0*BX0)*OVER8PI)
      ALFZ=BZ0/SQRT(FOURPI)
      ALFY=BY0/SQRT(FOURPI)
      ALFX=BX0/SQRT(FOURPI)
      ALF=SQRT(ALFX*ALFX+ALFY*ALFY+ALFZ*ALFZ)
      CSM=SQRT(.5*(1.+ALF*ALF-SQRT((1+ALF*ALF)**2-4.*ALFZ*ALFZ)))
      CFM=SQRT(.5*(1.+ALF*ALF+SQRT((1+ALF*ALF)**2-4.*ALFZ*ALFZ)))
      PRINT *,' BETA= ',BETA
      PRINT *,' ALFVEN VELOCITY (A,AX,AY,AZ)= ',ALF,ALFX,ALFY,ALFZ
```

```
      PRINT *,' SLO MODE SPEED= ',CSM
      PRINT *,' FAST MODE SPEED= ',CFM

      CSURF=SQRT((SQRT((ALF**2+2)**2+(GM1+1)**2*ALF**4+4*GM1*ALF**2)
     $      -(ALF**2+2))/(0.5*(GM1+1)**2*ALF**2+2*GM1))
      ME=SQRT(OM**2*(1/CSURF**2-1.0))
      CT=SQRT(ALF**2/(1+ALF**2))
      MO=OM/CSURF*SQRT((1-CSURF**2)*(ALF**2-CSURF**2)
     $    /(1+ALF**2)/(CT**2-CSURF**2))
      WRITE(*,1000)CSURF,MO,ME
 1000 FORMAT(1X,'Csurf= ',f10.5,'  Mo= ',F10.5,'  Me= ',F10.5)
      RETURN
      END

*************************************************************************
* THIS ROUTINE INITIALIZES THE GRID AND CONVERTS
* R TO MOVING GRID
*************************************************************************
      SUBROUTINE SETGRID(R,Z,DT)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL T,R(0:31,1:50,8),Z(0:31,1:50),DT
* GLOBAL VARIABLES
      REAL G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL TMAX,TPRINT,PFLAG,TVIEW
      REAL F1D15,F16D15,OVER8PI,PI,FOURPI
      REAL DZDX(0:31,1:50),DZDL(0:31,1:50),DZDT(0:31,1:50)
      COMPLEX DX(0:31)
     0COMMON /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
     1       /PARAM/G,QH,GM1,VP,OM,LZ,RM0,FV,ZETA2,ZETA1,ETA0,LX,LY,ERF
     2       /CONST/F1D15,F16D15,OVER8PI,PI,FOURPI
     3       /PRINTS/TMAX,TPRINT,PFLAG,TVIEW
     4       /GRID/DZDX,DZDL,DZDT,DX
* LOCAL VARIABLES
      COMPLEX II
      INTEGER J,L,M
      REAL A(0:31,1:50)
* CREATE DX FOR DERIVATIVES
      II=(0.,1.)
      IF(NX.GT.0)THEN
         DO 10 J=0,NC
            DX(J)=-2.*PI*II*J/(LX*NX)
   10       CONTINUE
         ENDIF
      DT=0.01
      DO 20 J=0,NR
         DO 30 L=1,NL
   30       Z(J,L)=LZ*(L-1)
   20    CONTINUE

      DO 50 J=0,NR
      DO 50 L=1,NL
         DZDT(J,L)=0.
   50 CONTINUE

      CALL DDL(Z,A)
   DO 60 J=0,NR
      DO 60 L=1,NL
      DZDL(J,L)=A(J,L)
   60 CONTINUE
```

```
* ------------------------------------------------------------------------
* MULTIPLY BY DZDL TO CHANGE GAS VARIABLES AS DESIRED
* FOR DYNAMICAL ZONING
* ------------------------------------------------------------------------
      DO 80 M=1,8
      DO 80 J=0,NR
      DO 80 L=1,NL
   80    R(J,L,M)=R(J,L,M)*DZDL(J,L)
      T=0.0
C      CALL BC(R,Z,T)
      RETURN
      END
*****************************************************************
* THIS SUBROUTINE SETS UP THE FFT PARAMETERS
*****************************************************************
      SUBROUTINE INITFFT(R)
      IMPLICIT NONE
* PASSED VARIABLES
      REAL R(0:31,1:50,8)
* GLOBAL VARIABLES
      INTEGER LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      REAL WORK(0:31,1:50,8),TRIGS(451)
      INTEGER IFAX(13),NPTS,INC,JUMP,NVEC
      INTEGER SYMFLAG

      COMMON /FFT/WORK,SYMFLAG
     $       /NUMBER/LM,LM2,LM3,LM4,NC,NL,NR,NX,XLO,XHI,ZLO,ZHI,SPLIT
      SYMFLAG=1
      PRINT *, 'SYMFLAG= ',SYMFLAG
      IF ((NX.LT.4).AND.(SYMFLAG.NE.3))SYMFLAG=0
      IF ((SYMFLAG.NE.0).AND.(SYMFLAG.NE.3))THEN
         IF(NL.NE.(2*(NL/2)))THEN
            PRINT *,'NL must be even.'
            STOP
            ENDIF
         ELSE
            XHI=0
         ENDIF
      RETURN
      END
```

Acknowledgements

I would like to express my gratitude to Dr. Robert Stein for his patient guidance. I would also like to thank Dr. Sheridan Simon and Dr. Rex Adelberger for their support and assistance— both professional and personal. I would also like to especially thank Charlie White and the rest of the Guilford College computer services staff for their patience and assistance.

This research was supported in part by NSF grant AST 83-16231.

The encouragement of Guilford College and their support, both economic and moral, has made the completion of this work possible.

Bibliography

Abramowitz, M., and I. A. Stegun. 1964. *Handbook of Mathematical Functions.* Volume 55 of *Applied Mathematics Series.* New York: Dover Publications.

Alexander, L. F., and J. D. Walecka. 1980. *Theoretical Mechanics of Particles and Continua.* New York: McGraw-Hill.

Allen, C. W. 1973. *Astrophysical Quantities.* London: The Athlone Press.

Braun, D,C., T.L. Duvall, and B.J. Labonte. 1987. Acoustic Absorption by Sunspots. *Astrophysical Journal* **319**:L27

Braun, D,C., T.L. Duvall, and B.J. Labonte. 1988. The Absorption of High-Degree P_mode Oscillations In and Around Sunspots. Accepted for publication in the *Astrophysical Journal.* May,1988.

Castor, J. I., C. G. Davis, and D. K. Davisson. 1977. *Dynamical Zoning Within a Lagrangian Mesh by Use of DYN, a Stellar Pulsation Code.* Los Alamos: Los Alamos Scientific Laboratory.

Conte, S. D., and C. de Boor. 1972. *Elementary Numerical Analysis: An Algorithmic Approach.* New York: McGraw-Hill.

Gottlieb, D., and S. A. Orszag. *Numerical Analysis of Spectral Methods: Theory and Applications.* CBMS-NSF Regional Conference series, monograph 26. Philadelphia: Society of Industrial and Applied Mathematics.

Huang, Kerson. 1963. *Statistical Mechanics.* New York: John Wiley & Sons.

Hughes, W. F., and F. J. Young. 1966. *The Electrodynamics of Fluids.* New York: John Wiley & Sons.

Jackson, J. D. 1975. *Classical Electrodynamics.* New York: John Wiley & Sons.

Krall, N. A., and A. W. Trivelpiece. 1973. *Principles of Plasma Physics.* New York: McGraw-Hill.

Landau, L. D., and E. M. Lifshitz. 1959. *Fluid Mechanics.* Volume 6 of *Course of Theoretical Physics.* Translated from the Russian by J. B. Sykes and W. H. Reid. Oxford: Pergamon Press.

---------- . 1960. *Electrodynamics of Continuous Media.* Volume 8 of *Course of Theoretical Physics.* Translated from the Russian by J. B. Sykes and J. S. Bell. Oxford: Pergamon Press.

Lighthill, James. 1978. *Waves in Fluids.* Cambridge: Cambridge University Press.

Orlanski, I. 1976. A simple boundary condition for unbounded hyperbolic flows. *Journal of Computational Physics* **21**:251-269.

Orszag, S. A. 1980. Spectral Methods for problems in complex geometries. *Journal of Computational Physics* **27**:70-92.

Pain, H. J. 1983. *The Physics of Vibrations and Waves.* Chichester: John Wiley & Sons.

Press, W. H., B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. 1986. *Numerical Recipes: The Art of Scientific Computing.* Cambridge: Cambridge University Press.

Rae, I. C., and B. Roberts. 1983. MHD wave motion in magnetically structured atmospheres. *Astronomy and Astrophysics* **119**:28-34.

Roberts, B. 1981. Wave propagation in a magneticlly structured atmosphere I: Surface Waves at a magnetic interface. *Solar Physics* **69**:27-38.

———— 1981. Wave propagation in a magneticlly structured atmosphere II: Waves in a magnetic slab. *Solar Physics* **69**:38-56.

———— 1982. Wave propagation in a magneticlly structured atmosphere III: The slab in a magnetic environment. *Solar Physics* **76**:239-259.

Steinolfson, R.S. 1984. Resonant absorption of phased-mixed Alfvén surface waves in ideal and resistive magnetohydrodynamics: Initial-value problem. *Phys. Fluids* **27**:781-783.

Tataronis, J.A. 1975. Energy absorption in the continuous spectrum of ideal MHD. *Journal of Plasma Physics* **13**:87-105.

Wilson, P. R. 1978. Wave modes in a magnetic flux sheath. *Astrophysical Journal* **221**:672-676.