5-4-2007

Numerical Hydrodynamics of Relativistic Extragalactic Jets

Eunwoo Choi

Numerical Hydrodynamics of Relativistic Extragalactic Jets

by

Eunwoo Choi

Under the Direction of Paul J. Wiita

Abstract

This dissertation describes a multidimensional relativistic hydrodynamic code which solves the special relativistic hydrodynamic equations as a hyperbolic system of conservation laws based on the total variation diminishing (TVD) scheme. Several standard tests and test simulations are presented to demonstrate the accuracy, robustness and flexibility of the code. Using this code we have studied three-dimensional hydrodynamic interactions of relativistic extragalactic jets with two-phase ambient media. The deflection angle of the jet is influenced more by the density contrast of the cloud than by the beam Mach number of the jet, and a relativistic jet with low relativistic beam Mach number can eventually be slightly bent after it crosses the dense cloud. Relativistic jet impacts on dense clouds do not necessarily destroy the clouds completely, and much of the cloud body can survive as a coherent blob due to the combination of the geometric influence of off-axis collisions and the lower rate of cloud fragmentation through the Kelvin-Helmholtz instability for relativistic flows. We find that relativistic jets interacting with clouds can produce synchrotron emission knots similar to structures observed in many VLBI-scale radio sources and the synchrotron emission peaks right before the jet passes through the cloud.

Index Words: Extragalactic jets, Hydrodynamics, Numerical methods, Special relativity

Numerical Hydrodynamics of Relativistic Extragalactic Jets

by

Eunwoo Choi

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

in the College of Arts and Sciences

Georgia State University

2007

Numerical Hydrodynamics of Relativistic Extragalactic Jets

by

Eunwoo Choi

Major Professor:	Paul J. Wiita
Committee:	D. Michael Crenshaw
	Douglas R. Gies
	Xiaochun He
	H. Richard Miller

Electronic Version Approved:

Office of Graduate Studies
College of Arts and Sciences
Georgia State University
May 2007

Table of Contents

List of Tables v

List of Figures vi

1 Introduction 1

2 Numerical Relativistic Hydrodynamics 10
2.1 Basic Equations . . . 10
2.2 Characteristic Decomposition . . . 12
2.3 One-Dimensional Functioning Code Based on the TVD Scheme . . . 15
2.4 Multidimensional Extension . . . 18
2.5 Lorentz Transformation . . . 19

3 Numerical Tests 21
3.1 Relativistic Shock Tube . . . 21
3.2 Relativistic Wall Shock . . . 27
3.3 Relativistic Blast Wave . . . 31
3.4 Relativistic Hawley-Zabusky Shock . . . 33
3.5 Relativistic Extragalactic Jets . . . 36

4 Problem Description 38

5 Numerical Simulations 42
5.1 Basic Equations . . . 42
5.2 Numerical Method and Setup . . . 44

6 Results 49
6.1 Morphology and Dynamics . . . 49
6.2 Cloud Evolutions . . . 61
6.3 Synchrotron Emission . . . 67

7 Summary and Discussion 74

References 78

Appendix 81

List of Tables

3.1 Norm errors for the relativistic shock tube tests 24

3.2 Mean errors for the relativistic wall shock tests 29

5.1 Simulation parameters 48

List of Figures

1.1 Image of the powerful FR I radio galaxy 3C 31 8

1.2 Image of the prototype FR II radio galaxy Cygnus A 9

3.1 (a) 1D, 2D, and 3D mildly relativistic shock tube tests 25

3.1 (b) 1D, 2D, and 3D highly relativistic shock tube tests 26

3.2 One-dimensional relativistic wall shock test 30

3.3 Two-dimensional relativistic blast wave test 32

3.4 Two-dimensional relativistic Hawley-Zabusky shock 35

3.5 Two-dimensional relativistic extragalactic jet 37

6.1 (a) Gray-scale images of density, pressure, and Lorentz factor for model M1 . 55

6.1 (b) Gray-scale images of density, pressure, and Lorentz factor for model M2 . 56

6.1 (c) Gray-scale images of density, pressure, and Lorentz factor for model M3 . 57

6.1 (d) Gray-scale images of density, pressure, and Lorentz factor for model M4 . 58

6.2 Distributions of density, pressure, and Lorentz factor 59

6.3 Projection images of Lorentz factors 60

6.4 Volume-rendering images of cloud density 65

6.5 Time evolutions of integrated quantities 66

6.6 Contour maps of synchrotron intensity 72

6.7 Total synchrotron intensity curves 73

Chapter 1

Introduction

Many high-energy astrophysical problems involve relativistic flows, and thus understanding relativistic flows is important for correctly interpreting astrophysical phenomena. For instance, intrinsic beam velocities larger than $0.9c$ are typically required to explain the apparent superluminal motions observed in relativistic jets in microquasars in the Galaxy (Mirabel & Rodríguez, 1999) as well as in extragalactic radio sources associated with active galactic nuclei (Zensus, 1997). In some powerful extragalactic radio sources, ejections from galactic nuclei produce true beam velocities of more than $0.98c$. Relativistic descriptions are also inevitable in other situations of rapid expansion such as the early stages of supernova explosions (Burrows, 2000) and the production of energetic gamma-ray bursts (Mészáros, 2002). In the general fireball model of gamma-ray bursts, the internal energy of gas is converted into the bulk kinetic energy during expansion and this expansion leads to relativistic outflows with high bulk Lorentz factors $\gtrsim 100$. Since such relativistic flows are highly nonlinear and intrinsically complex, in addition to possessing large Lorentz factors, often studying them *numerically* is the only possible approach.

For numerical study of non-relativistic hydrodynamics, explicit finite difference upwind schemes (those dominated by backward differences toward the direction from which the fluid is flowing), have been developed and implemented successfully. The schemes which have been used for astrophysical research include the Roe scheme (Roe, 1981), the total variation diminishing (TVD) scheme (Harten, 1983), the piecewise parabolic method (PPM) scheme

(Colella & Woodward, 1984), and the essentially non-oscillatory (ENO) scheme (Harten et al., 1987). These schemes are based on exact or approximate Riemann solvers using the characteristic decomposition of the hyperbolic system of hydrodynamic conservation equations. They all are able to capture sharp discontinuities robustly in the complex flows, and to describe the physical solution accurately.

Although the upwind schemes were originally developed for non-relativistic hydrodynamics, some have been extended to special relativistic hydrodynamics. For instance, Dolezal & Wong (1995) adapted the ENO scheme to one-dimensional relativistic hydrodynamics. They fulfilled the ENO scheme using the local characteristic approach which depends on the local linearizion of the system of conservation equations. Martí & Müller (1996) adapted the PPM scheme to one-dimensional relativistic hydrodynamics using an exact relativistic Riemann solver to calculate numerical fluxes at cell interfaces. Donat et al. (1998) and Aloy et al. (1999a) constructed multidimensional relativistic hydrodynamic codes based on the ENO scheme and the PPM scheme, respectively. Reviews of various numerical approaches and test problems can be found in Martí & Müller (2003) and Wilson & Mathews (2003). These works showed that the advantages of the upwind schemes, such as high accuracy and robustness, in ordinary hydrodynamics are carried over to relativistic hydrodynamics.

In the first half of this dissertation we describe a multidimensional code for special relativistic hydrodynamics based on the total variation diminishing (TVD) scheme (Harten, 1983). The TVD scheme is an explicit Eulerian finite difference upwind scheme and an extension of the Roe scheme to second-order accuracy in space and time. The advantage of the TVD scheme is that a code based on it is simple and fast, and yet performs well.

A non-relativistic hydrodynamic code based the TVD scheme was built and applied to astrophysical problems such as large scale structure formation in the universe by Ryu et al. (1993). The special relativistic hydrodynamic code in this dissertation was built by extending this non-relativistic code. All the components of the non-relativistic code were kept, so the relativistic code has a structure parallel to that of the non-relativistic counterpart. This approach makes the relativistic code comprehensible and easily usable. Through tests, we demonstrate that the newly developed code for special relativistic hydrodynamics can handle interesting astrophysical problems involving large Lorentz factors or ultrarelativistic regimes where energy densities greatly exceed rest mass densities. Most of the material in the first half was shown in Choi & Ryu (2005).

Relativistic jets emerging from extragalactic sources associated with active galactic nuclei (AGNs) are the most important means of transporting energy and mass from AGNs to an external medium over large distances. To understand how these relativistic jets interact with an inhomogeneous external medium containing small, dense gas clouds or clumps has been recognized as important for a long time. These interactions may substantially change the direction of relativistic jet flows, trigger extensive star formation in the shocked clouds, and possibly explain the basic mechanism behind the morphology of many extragalactic radio jets.

The morphology and power of jets at extragalactic scales are responsible for the key dichotomy of radio galaxies, the so-called FR I and FR II classes (Fanaroff & Riley, 1974). The FR I sources are dominated by emission from their inner parts, most of which comes directly from the jets themselves. The FR II sources have morphologies where most of the emission comes from the outer portions which contain hotspots. The nominal luminosity

boundary between FR I and FR II sources is $L(178\ \mathrm{MHz}) \sim 10^{25}$ W Hz^{-1} sr^{-1}, with the FR II sources being more powerful. Both FR I and FR II radio galaxies often extend over distances exceeding 100 kpc, but the largest sources tend to be FR II sources. The most widely accepted interpretation of the difference is that FR I jets become less powerful toward the outer regions and emit more in the inner regions as the result of a smooth deceleration from relativistic to nonrelativistic jet speed over the kpc scales (e.g., Bicknell, 1995). On the other hand, FR II sources retain powerful jet thrusts that feed the hotspots in their outer lobes, indicating that in these sources relativistic motion extends up to scales of hundreds of kpc.

On parsec scales, VLBI radio maps of the jets show a highly collimated bright core at one end of the jet and a series of knots which separate from the core, moving sometimes at apparently superluminal speeds. The knots are usually associated with shocks moving down the jets and outbursts in the radio (and other bands) are frequently coincident with the times at which these knots appear to emerge from the core (Hughes et al., 1985; Jorstad et al., 2001). In the now common and standard model, the superluminal motions are understood as a consequence of relativistic bulk motion of jets propagating at small angles to the line of sight with high Lorentz factors up to 20 or more. This relativistic bulk motion of the jet is believed to be a model for blazars. BL Lac objects are commonly understood to be the highly beamed fraction of the FR I radio galaxies, and flat spectrum radio quasars are highly beamed FR II radio galaxies, while "normal" radio loud quasars are FR II radio galaxies viewed at intermediate angles to the jet line of sight (Urry & Padovani, 1995).

Figure 1.1 shows the image of the radio galaxy 3C 31 (NGC 383). This system is a powerful FR I radio source with conical inner jets originating from the radio core of the

galaxy. The jets eventually develop into bent and distorted lobes which stretch to a distance of 300 kpc from the center of the radio galaxy. Figure 1.2 shows Cygnus A, the prototype FR II radio galaxy, which is both very close to us ($z = 0.056$) and very powerful. This is one of the few FR II radio sources where both jet and counterjet can be seen. In most FR II radio galaxies the apparent jet is Doppler boosted while the receding jet is Doppler dimmed and thus too faint to see.

Recent observations have revealed strong evidence of features associated with changes in jet directions resulting from interactions with small gas clouds in the narrow-line regions of Seyfert galaxies (e.g., Mundell et al., 2003). Fast outflows of gas observed in the central regions of powerful radio galaxies can also be caused by such interactions (e.g., Emonts et al., 2005; Morganti et al., 2005). The most likely interpretation of fast outflows is that all gas clouds are not destroyed by the jet; some clouds can severely disrupt the jet while some clouds are accelerated to the observed high outflow velocities by the thrust of the jet. It is argued that despite of the high energies involved in the interactions, only a few percent of the outflowing gas appears to be ionized, while the rest of the gas cools and becomes neutral due to highly efficient cooling near the jet bow shock.

In the context of nonrelativistic hydrodynamic simulations, previous numerical works were performed to investigate jet interactions with clouds (de Gouveia Dal Pino, 1999; Higgins et al., 1999; Wang et al., 2000; Saxton et al., 2005) or jets crossing a medium interface (e.g., Wiita et al., 1990; Wiita & Norman, 1992), focusing on the effects of the interactions on the morphology and kinematics of jets. Others studied shock interactions, focusing on the structure and evolution of the clouds produced by the interactions in adiabatic cases (Klein et al., 1994; Xu & Stone, 1995; Poludnenko et al., 2002) and in radiative cases (Mellema et

al., 2002; Fragile et al., 2004). According to de Gouveia Dal Pino (1999), simulations with conditions appropriate to protostellar jets making off-axis collisions with clouds produced a deflected beam. The deflection angle tended to decrease with time as the beam slowly penetrated the cloud and when the jet penetrated most of the cloud the deflected beam faded and the jet resumed its original propagation direction. Wang et al. (2000) found the following: powerful extragalactic jets eventually destroyed the clouds they considered, and these collisions induced nonaxisymmetric instabilities in the jets; weak jets can be effectively halted or destroyed by massive clouds; and slow, dense jets that were bent remained stable for extended times. Synthetic radio images produced by hydrodynamic simulations for comparison with observations also supported the hypothesis that these interactions are responsible for the distorted structures of some radio jets (e.g., Higgins et al., 1999). All those numerical works considered nonrelativistic jet speeds less than $0.5c$, but the observed apparent superluminal motions of extragalactic radio sources indicate intrinsic jet speeds up to at least $0.98c$ (Zensus, 1997). Thus, it is essential to perform relativistic hydrodynamic simulations of this problem in order to cover the range of true jet speeds.

Since time-dependent numerical simulations of relativistic jets were first reported (van Putten, 1993; Duncan & Hughes, 1994; Martí et al., 1994), multidimensional relativistic hydrodynamic simulations have been used as an important method in understanding relativistic jets (Martí et al., 1997; Komissarov & Falle, 1998; Aloy et al., 1999b; Rosen et al., 1999; Hughes et al., 2002; Mizuta et al., 2004). The morphological and dynamical properties of relativistic jets propagating through a homogeneous medium were studied by Martí et al. (1997) in two dimensions and by Aloy et al. (1999b) in three dimensions. Komissarov & Falle (1998) investigated the large-scale flows produced by classical and relativistic jets

in a uniform external medium using analytical and numerical studies. Hughes et al. (2002) performed in three dimensions a study of the deflection of relativistic jets by an oblique density gradient and of the precession of relativistic jets. They found that fast relativistic jets can be significantly influenced by an oblique density gradient, showing a rotation of the Mach disk with the flow bent via a strong oblique internal shock.

In the second half of this dissertation we present results from three-dimensional hydrodynamic simulations of the interactions of relativistic jets with dense clouds. We focus on the off-axis collision of the relativistic jet with a steady spherical cloud. The main concerns of this study are how the relativistic jets are influenced by these interactions and how the interaction affects the evolution of the cloud. Most of the material in the second half has appeared in Choi et al. (2007).

This dissertation is organized as follows. In Chapter 2 we describe the step by step procedures for building the code including the basic equations, characteristic decomposition, TVD scheme, multidimensional extension, and Lorentz transformation. Numerical tests are presented in Chapter 3. In Chapter 4 we briefly outline the dynamical problem, while the basic equations, numerical method and setup we employ are described in Chapter 5. In Chapter 6 we describe the results, and we present a summary and discussion in Chapter 7. Finally the parallelized version of the source code follows in the Appendix.

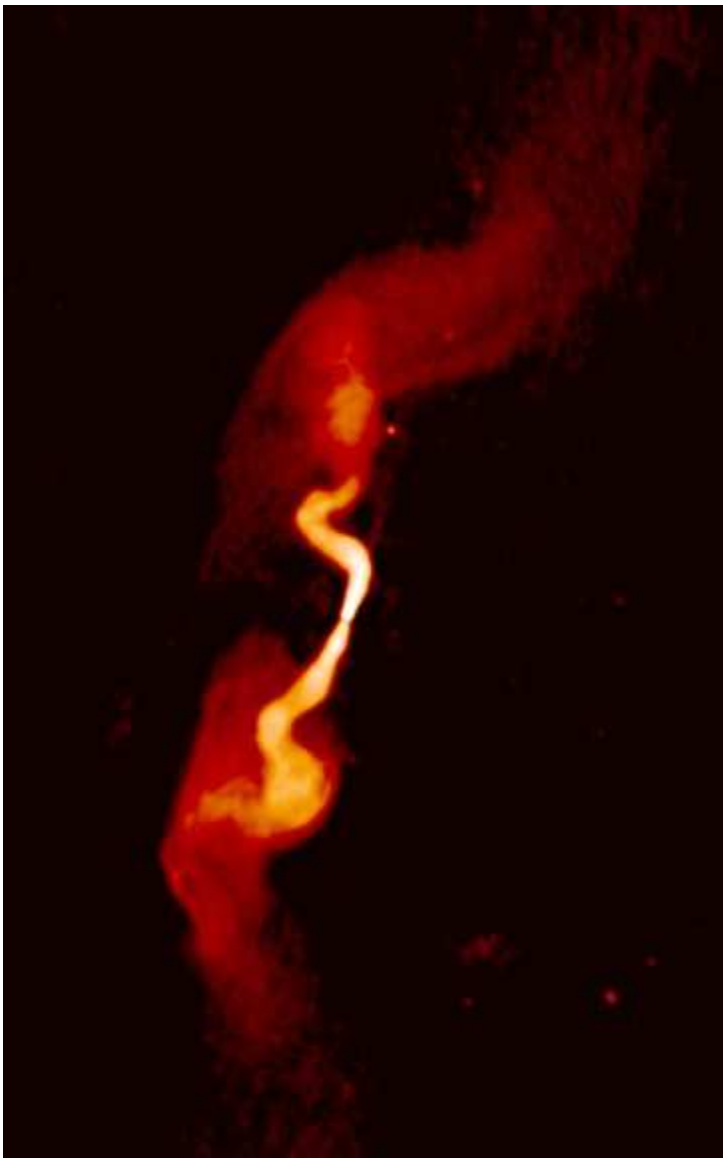

Figure 1.1: Very Large Array (VLA) 1.4 GHz image of the powerful FR I radio galaxy 3C 31. The conical inner jets originate from the center of the galaxy and eventually develop into bent and distorted lobes at a distance of 300 kpc from the center. Image courtesy of NRAO/AUI.

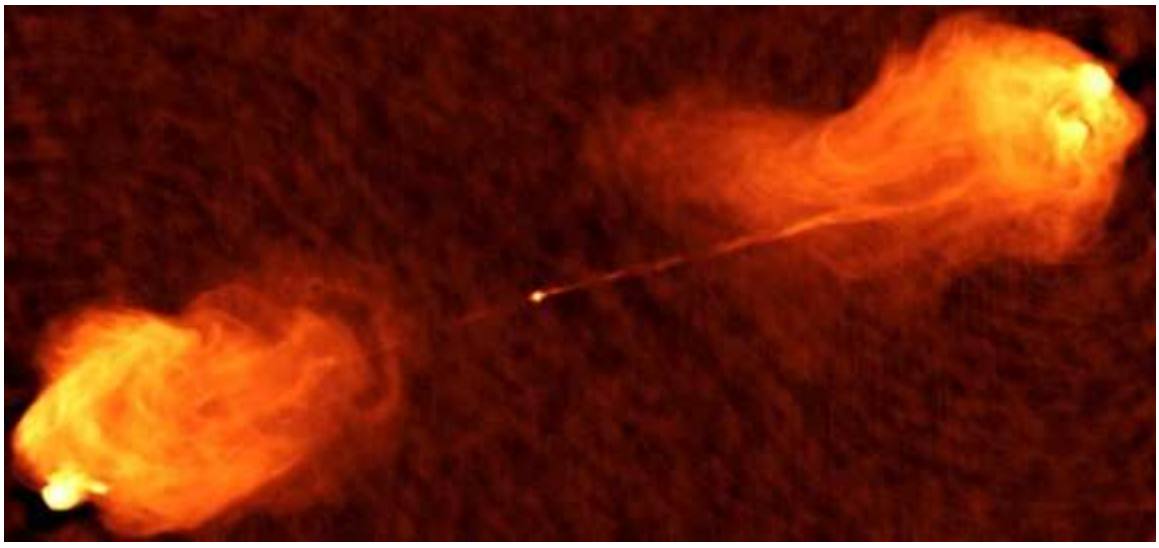

Figure 1.2: VLA 5 GHz image of the prototype FR II radio galaxy Cygnus A. This is both very close and very powerful and one of the few FR II radio sources where both jet and counterjet can be seen. Image courtesy of NRAO/AUI.

Chapter 2

Numerical Relativistic Hydrodynamics

2.1 Basic Equations

The ideal relativistic hydrodynamic equations can be written as a hyperbolic system of conservation equations

$$\frac{\partial D}{\partial t} + \frac{\partial}{\partial x_j}\left(D v_j\right) = 0, \tag{2.1}$$

$$\frac{\partial M_i}{\partial t} + \frac{\partial}{\partial x_j}\left(M_i v_j + p\delta_{ij}\right) = 0, \tag{2.2}$$

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}\left[\left(E + p\right) v_j\right] = 0, \tag{2.3}$$

where the equation of state is given by

$$p = \left(\gamma - 1\right)\left(e - \rho\right). \tag{2.4}$$

Here, D, M_i, and E are the mass density, momentum density, and total energy density in the reference frame, respectively and ρ, v_j, and e are the mass density, velocity, and internal plus mass energy density in the local rest frame, respectively. In general, the adiabatic index γ is taken as $5/3$ for mildly relativistic cases and as $4/3$ for ultrarelativistic cases where $e \gg \rho$. In equations (2.1)–(2.3), the indices i and j run over x, y, and z and the conventional Einstein summation is used. The speed of light is set to unity ($c \equiv 1$) throughout this dissertation.

The quantities in the reference frame are related to those in the local rest frame via Lorentz transformations

$$D = \Gamma\rho, \tag{2.5}$$

$$M_i = \Gamma^2 (e + p) v_i, \tag{2.6}$$

$$E = \Gamma^2 (e + p) - p, \tag{2.7}$$

where the Lorentz factor is given by

$$\Gamma = \frac{1}{\sqrt{1 - v^2}} \tag{2.8}$$

with $v^2 = v_x^2 + v_y^2 + v_z^2$.

In the non-relativistic limit, the quantities D, M_i, and E approach their non-relativistic counterparts ρ^N, $\rho^N v_i^N$, and $E^N + \rho^N c^2$, and equations (2.1)–(2.3) reduce to the non-relativistic hydrodynamic equations

$$\frac{\partial \rho^N}{\partial t} + \frac{\partial}{\partial x_j} \left(\rho^N v_j^N \right) = 0, \tag{2.9}$$

$$\frac{\partial \rho^N v_i^N}{\partial t} + \frac{\partial}{\partial x_j} \left(\rho^N v_i^N v_j^N + p^N \delta_{ij} \right) = 0, \tag{2.10}$$

$$\frac{\partial E^N}{\partial t} + \frac{\partial}{\partial x_j} \left[\left(E^N + p^N \right) v_j^N \right] = 0, \tag{2.11}$$

where the pressure is given by

$$p^N = (\gamma - 1) \left(E^N - \frac{1}{2} \rho^N {v^N}^2 \right). \tag{2.12}$$

2.2 Characteristic Decomposition

Equations (2.1)–(2.3) can be combined as

$$\frac{\partial \vec{q}}{\partial t} + \frac{\partial \vec{F}_j}{\partial x_j} = 0 \tag{2.13}$$

with the state and flux vectors

$$\vec{q} = \begin{bmatrix} D \\ M_i \\ E \end{bmatrix}, \qquad \vec{F}_j = \begin{bmatrix} Dv_j \\ M_i v_j + p\delta_{ij} \\ (E+p)\, v_j \end{bmatrix}, \tag{2.14}$$

or as

$$\frac{\partial \vec{q}}{\partial t} + A_j \frac{\partial \vec{q}}{\partial x_j} = 0, \qquad A_j = \frac{\partial \vec{F}_j}{\partial \vec{q}}. \tag{2.15}$$

Here, A_j is the 5×5 Jacobian matrix composed with the state and flux vectors. The construction of the matrix A_j can be simplified by introducing a parameter vector, $\vec{u}$, as

$$A_j = \frac{\partial \vec{F}_j}{\partial \vec{u}} \frac{\partial \vec{u}}{\partial \vec{q}}. \tag{2.16}$$

We choose the parameter vector which consists of the physical quantities in the local rest frame,

$$\vec{u} = \begin{bmatrix} \rho \\ v_i \\ e \end{bmatrix}. \tag{2.17}$$

In building an upwind code to solve a hyperbolic system of conservation equations, the eigen-structure (eigenvalues and eigenvectors) of the Jacobian matrix is required. Eigen-structures for relativistic hydrodynamics in multidimensions were previously described, for instance, in Donat et al. (1998). However, the state vector in this dissertation is different from that of Donat et al. (1998), so the eigen-structure is different. In the following, our eigen-structure of equation (2.16) is presented. We first define the specific enthalpy, h, and the the sound speed, c_s, respectively as

$$h = \frac{e+p}{\rho}, \qquad c_s^2 = \frac{\gamma p}{\rho h}. \tag{2.18}$$

Then the eigenvalues of A_x for $j = x$ are

$$a_1 = \frac{(1-c_s^2)\, v_x - \sqrt{(1-v^2)\, c_s^2 \left[1 - v^2 c_s^2 - (1-c_s^2)\, v_x^2\right]}}{1 - v^2 c_s^2}, \tag{2.19}$$

$$a_2 = v_x, \tag{2.20}$$

$$a_3 = v_x, \tag{2.21}$$

$$a_4 = v_x, \tag{2.22}$$

$$a_5 = \frac{(1-c_s^2)\, v_x + \sqrt{(1-v^2)\, c_s^2 \left[1 - v^2 c_s^2 - (1-c_s^2)\, v_x^2\right]}}{1 - v^2 c_s^2}. \tag{2.23}$$

The eigenvalues a_{1-5} represent the five characteristic speeds associated with two sound wave modes ($a_{1,5}$) and three entropy modes (a_{2-4}).

The complete set of the corresponding right eigenvectors ($A_x\vec{R} = a\vec{R}$) is

$$\vec{R}_1 = \left[\frac{1 - v_x a_1}{\Gamma h\left(1 - v_x^2\right)}, a_1, \frac{\left(1 - v_x a_1\right) v_y}{1 - v_x^2}, \frac{\left(1 - v_x a_1\right) v_z}{1 - v_x^2}, 1\right]^{\mathrm{T}}, \tag{2.24}$$

$$\vec{R}_2 = \left[\frac{-\Gamma\left(2h - 1\right) v_y}{h}, 0, 1, 0, 0\right]^{\mathrm{T}}, \tag{2.25}$$

$$\vec{R}_3 = \left[\frac{\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) + h}{\Gamma h}, v_x, 0, 0, 1\right]^{\mathrm{T}}, \tag{2.26}$$

$$\vec{R}_4 = \left[\frac{-\Gamma\left(2h - 1\right) v_z}{h}, 0, 0, 1, 0\right]^{\mathrm{T}}, \tag{2.27}$$

$$\vec{R}_5 = \left[\frac{1 - v_x a_5}{\Gamma h\left(1 - v_x^2\right)}, a_5, \frac{\left(1 - v_x a_5\right) v_y}{1 - v_x^2}, \frac{\left(1 - v_x a_5\right) v_z}{1 - v_x^2}, 1\right]^{\mathrm{T}}. \tag{2.28}$$

The complete set of the left eigenvectors ($\vec{L}A_x = a\vec{L}$), which are orthonormal to the right eigenvectors, is

$$\vec{L}_1 = \left[\frac{-\Gamma h\left(v_x - a_5\right)}{\left(h - 1\right)\left(a_1 - a_5\right)}, \Delta_{12}, \frac{-\Gamma^2\left(2h - 1\right)\left(v_x - a_5\right) v_y}{\left(h - 1\right)\left(a_1 - a_5\right)}, \frac{-\Gamma^2\left(2h - 1\right)\left(v_x - a_5\right) v_z}{\left(h - 1\right)\left(a_1 - a_5\right)}, \Delta_{15}\right], \tag{2.29}$$

$$\vec{L}_2 = \left[\frac{\Gamma h v_y}{h - 1}, \frac{\left[\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) + h\right] v_x v_y}{\left(h - 1\right)\left(1 - v_x^2\right)}, \frac{\Gamma^2\left(2h - 1\right) v_y^2}{h - 1} + 1, \frac{\Gamma^2\left(2h - 1\right) v_y v_z}{h - 1},\right.$$
$$\left.\frac{-\left[\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) + h\right] v_y}{\left(h - 1\right)\left(1 - v_x^2\right)}\right], \tag{2.30}$$

$$\vec{L}_3 = \left[\frac{\Gamma h}{h - 1}, \frac{\left[\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) + 1\right] v_x}{\left(h - 1\right)\left(1 - v_x^2\right)}, \frac{\Gamma^2\left(2h - 1\right) v_y}{h - 1}, \frac{\Gamma^2\left(2h - 1\right) v_z}{h - 1},\right.$$
$$\left.\frac{-\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) - 1}{\left(h - 1\right)\left(1 - v_x^2\right)}\right], \tag{2.31}$$

$$\vec{L}_4 = \left[\frac{\Gamma h v_z}{h - 1}, \frac{\left[\Gamma^2\left(2h - 1\right)\left(v^2 - v_x^2\right) + h\right] v_x v_z}{\left(h - 1\right)\left(1 - v_x^2\right)}, \frac{\Gamma^2\left(2h - 1\right) v_y v_z}{h - 1}, \frac{\Gamma^2\left(2h - 1\right) v_z^2}{h - 1} + 1,\right.$$

$$\left. \frac{-\left[\Gamma^2\left(2h-1\right)\left(v^2-v_x^2\right)+h\right]v_z}{\left(h-1\right)\left(1-v_x^2\right)}\right], \tag{2.32}$$

$$\vec{L}_5 = \left[\frac{-\Gamma h\left(v_x-a_1\right)}{\left(h-1\right)\left(a_5-a_1\right)}, \Delta_{52}, \frac{-\Gamma^2\left(2h-1\right)\left(v_x-a_1\right)v_y}{\left(h-1\right)\left(a_5-a_1\right)}, \frac{-\Gamma^2\left(2h-1\right)\left(v_x-a_1\right)v_z}{\left(h-1\right)\left(a_5-a_1\right)}, \Delta_{55}\right], \tag{2.33}$$

where the auxiliary variables are defined as

$$\Delta_{12} = \frac{-\left[\Gamma^2\left(2h-1\right)\left(v^2-v_x^2\right)+1\right]\left(v_x-a_5\right)v_x}{\left(h-1\right)\left(1-v_x^2\right)\left(a_1-a_5\right)} + \frac{1}{a_1-a_5}, \tag{2.34}$$

$$\Delta_{15} = \frac{\left[\Gamma^2\left(2h-1\right)\left(v^2-v_x^2\right)+1\right]\left(v_x-a_5\right)}{\left(h-1\right)\left(1-v_x^2\right)\left(a_1-a_5\right)} - \frac{a_5}{a_1-a_5}, \tag{2.35}$$

$$\Delta_{52} = \frac{-\left[\Gamma^2\left(2h-1\right)\left(v^2-v_x^2\right)+1\right]\left(v_x-a_1\right)v_x}{\left(h-1\right)\left(1-v_x^2\right)\left(a_5-a_1\right)} + \frac{1}{a_5-a_1}, \tag{2.36}$$

$$\Delta_{55} = \frac{\left[\Gamma^2\left(2h-1\right)\left(v^2-v_x^2\right)+1\right]\left(v_x-a_1\right)}{\left(h-1\right)\left(1-v_x^2\right)\left(a_5-a_1\right)} - \frac{a_1}{a_5-a_1}. \tag{2.37}$$

The eigenvalues and eigenvectors of A_y and A_z can be obtained by properly redefining indices. We note that the eigenvalues are the same regardless of the choice of state or parameter vectors. But the right and left eigenvectors are different or can be presented in different forms.

2.3 One-Dimensional Functioning Code Based on the TVD Scheme

The TVD scheme we employ to build a one-dimensional functioning code is practically identical to that in Harten (1983) and Ryu et al. (1993). But for completeness, the procedure is shown here. The state vector $\vec{q}_i^n$ at the cell center i at the time step n is updated by calculating the modified flux vector $\vec{\bar{f}}_{x,i\pm1/2}$ along the x-direction at the cell interface $i\pm1/2$

as follows:

$$L_x \vec{q}_i^n = \vec{q}_i^n - \frac{\Delta t^n}{\Delta x}\left(\bar{\vec{f}}_{x,i+1/2} - \bar{\vec{f}}_{x,i-1/2}\right), \tag{2.38}$$

$$\bar{\vec{f}}_{x,i+1/2} = \frac{1}{2}\left[\vec{F}_x(\vec{q}_i^n) + \vec{F}_x(\vec{q}_{i+1}^n)\right] - \frac{\Delta x}{2\Delta t^n}\sum_{k=1}^{5}\beta_{k,i+1/2}\vec{R}_{k,i+1/2}^n, \tag{2.39}$$

$$\beta_{k,i+1/2} = Q_k(\frac{\Delta t^n}{\Delta x}a_{k,i+1/2}^n + \gamma_{k,i+1/2})\alpha_{k,i+1/2} - (g_{k,i} + g_{k,i+1}), \tag{2.40}$$

$$\gamma_{k,i+1/2} = \begin{cases} (g_{k,i+1} - g_{k,i})/\alpha_{k,i+1/2} & \text{for} \quad \alpha_{k,i+1/2} \neq 0, \\ 0 & \text{for} \quad \alpha_{k,i+1/2} = 0, \end{cases} \tag{2.41}$$

$$g_{k,i} = \text{sign}(\tilde{g}_{k,i+1/2})\max\{0, \min[|\tilde{g}_{k,i+1/2}|, \text{sign}(\tilde{g}_{k,i+1/2})\tilde{g}_{k,i-1/2}]\}, \tag{2.42}$$

$$\tilde{g}_{k,i+1/2} = \frac{1}{2}\left[Q_k(\frac{\Delta t^n}{\Delta x}a_{k,i+1/2}^n) - \left(\frac{\Delta t^n}{\Delta x}a_{k,i+1/2}^n\right)^2\right]\alpha_{k,i+1/2}, \tag{2.43}$$

$$\alpha_{k,i+1/2} = \vec{L}_{k,i+1/2}^n \cdot \left(\vec{q}_{i+1}^n - \vec{q}_i^n\right), \tag{2.44}$$

$$Q_k(x) = \begin{cases} x^2/4\varepsilon_k + \varepsilon_k & \text{for} \quad |x| < 2\varepsilon_k, \\ |x| & \text{for} \quad |x| \geq 2\varepsilon_k. \end{cases} \tag{2.45}$$

Here, $k = 1$ to 5 stand for the five characteristic modes. The internal parameters ε_k's are associated with numerical viscosity, and defined for $0 \leq \varepsilon_k \leq 0.5$; $\varepsilon_{1,5} = 0.1 - 0.3$ for the sound wave modes and $\varepsilon_{2-4} = 0 - 0.1$ for the entropy modes are reasonable choices.

We note that the flux limiter in equation (2.42) is the min-mod limiter. The min-mod limiter is known to be very stable but has the cost of additional diffusion. To reproduce sharper structures with less diffusion, other flux limiters, such as the monotonized central difference limiter (MC limiter)

$$g_{k,i} = \text{sign}(\tilde{g}_{k,i+1/2})\max\{0, \min[\frac{1}{2}(|\tilde{g}_{k,i+1/2}| + \text{sign}(\tilde{g}_{k,i+1/2})\tilde{g}_{k,i-1/2}), 2|\tilde{g}_{k,i+1/2}|,$$

$$2\text{sign}(\tilde{g}_{k,i+1/2})\tilde{g}_{k,i-1/2}]\}, \tag{2.46}$$

or the superbee limiter

$$g_{k,i} = \text{sign}(\tilde{g}_{k,i+1/2})\max\{0, \min[|\tilde{g}_{k,i+1/2}|, 2\text{sign}(\tilde{g}_{k,i+1/2})\tilde{g}_{k,i-1/2}], \min[2|\tilde{g}_{k,i+1/2}|,$$

$$\text{sign}(\tilde{g}_{k,i+1/2})\tilde{g}_{k,i-1/2}]\}, \tag{2.47}$$

may be used; however, these limiters are more susceptible to oscillations at discontinuities. In the tests described in Chapter 3, the min-mod limiter was used.

In order to define the physical quantities at the cell interfaces, the TVD scheme originally used Roe's linearization technique (Harten, 1983). Although it is possible to implement this linearization technique in the relativistic domain in a computationally feasible way (see Eulderink & Mellema, 1995), there is unlikely to be a significant advantage from the computational point of view. Instead, we simply calculate the algebraic averages of quantities at two adjacent cell centers to define the physical quantities at the cell interfaces;

$$v_{x,i+1/2} = \frac{v_{x,i} + v_{x,i+1}}{2}, \; v_{y,i+1/2} = \frac{v_{y,i} + v_{y,i+1}}{2}, \; v_{z,i+1/2} = \frac{v_{z,i} + v_{z,i+1}}{2}, \tag{2.48}$$

$$h_{i+1/2} = \frac{h_i + h_{i+1}}{2}, \tag{2.49}$$

$$c_{s,i+1/2} = \left[\frac{(\gamma - 1)\left(h_{i+1/2} - 1\right)}{h_{i+1/2}}\right]^{1/2}. \tag{2.50}$$

2.4 Multidimensional Extension

To extend the one-dimensional code to multidimensions, the procedure described in the previous section is applied separately to the y and z-directions. Multiple spatial dimensions are treated through the Strang-type dimensional splitting (Strang, 1968). Then, the state vector is updated by

$$\vec{q}^{n+1} = L_z L_y L_x \vec{q}^{n}. \tag{2.51}$$

In order to maintain second-order accuracy in time, the order of the dimensional splitting is permuted as follows

$$L_z L_y L_x,\ L_x L_y L_z,\ L_x L_z L_y,\ L_y L_z L_x,\ L_y L_x L_z,\ L_z L_x L_y. \tag{2.52}$$

The time step Δt^n is restricted by the usual Courant stability condition which presents variations in any quantity from being advected past any cell,

$$\Delta t^n = \min \left[\frac{C_{\rm Cour}\Delta x}{\max(a^n_{k,i+1/2})_x}, \frac{C_{\rm Cour}\Delta y}{\max(a^n_{k,i+1/2})_y}, \frac{C_{\rm Cour}\Delta z}{\max(a^n_{k,i+1/2})_z} \right]. \tag{2.53}$$

The Courant constant should be $C_{\rm Cour} < 1$. We typically use $C_{\rm Cour} \lesssim 0.9$. The time step is calculated at the beginning of a permutation sequence and used through the complete sequence.

2.5 Lorentz Transformation

In the code, the conserved quantities D, M_i, and E in the reference frame are evolved in time, but the physical quantities ρ, v_j, and e in the local rest frame are needed for fluxes to be estimated. The quantities ρ, v_j, and e can be obtained through Lorentz transformation of equations (2.5)–(2.7) at each time step. Schneider et al. (1993) showed that the transformation is reduced to a single quartic equation for v

$$f(v) = \left[\gamma v \left(E - Mv\right) - M\left(1 - v^2\right)\right]^2 - \left(1 - v^2\right) v^2 \left(\gamma - 1\right)^2 D^2 = 0, \tag{2.54}$$

where $M^2 = M_x^2 + M_y^2 + M_z^2$. They also showed that the physically meaningful solution for v is between the lower limit, v_1, and the upper limit, v_2,

$$v_1 = \frac{\gamma E - \sqrt{\left(\gamma E\right)^2 - 4\left(\gamma - 1\right) M^2}}{2\left(\gamma - 1\right) M},\ v_2 = \frac{M}{E}, \tag{2.55}$$

and that the solution is unique. Once v is known, the quantities ρ, v_j, and e can be straightforwardly calculated from the following relations

$$\rho = \frac{D}{\Gamma}, \tag{2.56}$$

$$v_x = \frac{M_x}{M} v,\ v_y = \frac{M_y}{M} v,\ v_z = \frac{M_z}{M} v, \tag{2.57}$$

$$e = E - M_x v_x - M_y v_y - M_z v_z. \tag{2.58}$$

Equation (2.54) could be solved using a numerical procedure such as the Newton-Raphson root-finding method, as suggested in Schneider et al. (1993). A problem with this numerical approach is that iterations can fail to converge. For instance, convergence can fail if one of the relativistic conditions is violated due to numerical errors, e.g., $M > E$, in a cell. This occurs mostly in extreme regimes. In addition, we found that convergence is often slow or sometimes fails in the limit $M \ll E$. On the other hand, quartic equations have analytic solutions. The general form of the four roots can be found in standard books such as Abramowitz & Stegun (1972) or on websites such as `http://mathworld.wolfram.com`. Although it is too difficult to prove analytically, we found numerically that for the physical meaningful values of v and c_s, $v < 1$ and $c_s < \sqrt{\gamma - 1}$, among the four roots of equation (2.54), two are complex and the other two are real. While the smaller real root is smaller than the lower limit v_1, the larger real root is between the two limits v_1 and v_2. So the larger real root is the one we are looking for, and we use its analytic formula in our code. The advantages of the analytic approach are obvious. It always gives a solution we are looking for, and it is easier to predict and deal with unphysical situations if one of the relativistic conditions is violated due to numerical errors.

Chapter 3

Numerical Tests

3.1 Relativistic Shock Tube

We have performed two sets of relativistic shock tube tests in the one, two, and three-dimensional computational boxes with $x = [0, 1]$, $y = [0, 1]$, and $z = [0, 1]$. Initially two different physical states are set up perpendicular to the direction along which waves propagate; along the x-axis in the one-dimensional calculation, along the diagonal line connecting $(0, 0)$ and $(1, 1)$ in the two-dimensional calculation, and along the diagonal line connecting $(0, 0, 0)$ and $(1, 1, 1)$ in the three-dimensional calculation. The initial states of the first test are

$$(\rho, v_x, v_y, v_z, p) = \begin{cases} (10, 0, 0, 0, 13.3) & 0 \leq x \leq 1/2, \\ (1, 0, 0, 0, 10^{-6}) & 1/2 < x \leq 1. \end{cases} \tag{3.1}$$

The initial states of the second test are

$$(\rho, v_x, v_y, v_z, p) = \begin{cases} (1, 0, 0, 0, 10^{3}) & 0 \leq x \leq 1/2, \\ (1, 0, 0, 0, 10^{-2}) & 1/2 < x \leq 1. \end{cases} \tag{3.2}$$

In equations (3.1) and (3.2), the expressions for the x's within the inequalities are appropriate for one dimension and are substituted by $(x + y)/2$ and $(x + y + z)/3$ for two and three dimensions, respectively. The first test involves a mildly relativistic flow and the second test involves a highly relativistic flow. In both tests, we assume the adiabatic index $\gamma = 5/3$

and the outflow condition is used for the x, y, and z-boundaries. Both tests were previously considered by several authors (e.g., Martí & Müller, 1996). The estimation of accuracy was done by comparing the numerical solutions with the exact solutions described in Thompson (1986) and Martí & Müller (1994). In Figures 3.1(a) and (b), our numerical solutions are shown as open circles and the exact solutions are represented by solid lines.

Figure 3.1(a) shows the mildly relativistic shock tube test done using 256, 256^2, and 256^3 cells with a Courant constant $C_{\rm Cour} = 0.9$ and the parameters $\varepsilon_{1,5} = 0.1$ and $\varepsilon_{2-4} = 0$. The plots of one, two, and three-dimensions correspond to times $t = 0.4$, $0.4\sqrt{2}$, and $0.4\sqrt{3}$, respectively. Structures such as the shock front (at $x = 0.83$), contact discontinuity (at $x = 0.78$) and rarefaction wave (ending at $x = 0.56$) are accurately produced. There are actually slight improvements in accuracy in the multidimensional calculations. Figure 3.1(b) shows the highly relativistic shock tube test done again using 256, 256^2, and 256^3 cells with a Courant constant $C_{\rm Cour} = 0.6$ and the parameters $\varepsilon_{1,5} = 0.1$ and $\varepsilon_{2-4} = 0$. The plots of one, two, and three-dimensions correspond to times $t = 0.4$, $0.4\sqrt{2}$, and $0.4\sqrt{3}$, respectively. The flow is more extreme, but the structure is correctly reproduced without spurious oscillations. But in the rest mass density profile the peak does not reach the value of the exact solution due to the coarseness of the computational cells. According to our tests, in a one-dimensional calculation, the peak can be very accurately reproduced when 2048 numerical cells are used. There are also improvements in accuracy in the multidimensional calculations.

For a more quantitative comparison, we have calculated the norm errors of the rest mass density, velocity, and pressure for different dimensions. The errors shown in Table 3.1 are calculated at the same times as in Figure 3.1. The errors are gradually reduced as the dimensionality increases and demonstrate a good agreement between the numerical and

exact solutions. Note that the values of $\|E(p)\|$ exceeding unity are still acceptable because these are from the initial large value of pressure.

Table 3.1: Norm errors for the relativistic shock tube tests

RST	$n_{\rm cell}$	$\|E(\rho)\|$	$\|E(v)\|$	$\|E(p)\|$
(a) 1D	256	1.1688E−01	6.0952E−02	9.3517E−02
2D	256^2	1.1264E−01	6.0586E−02	9.6789E−02
3D	256^3	9.1309E−02	5.8222E−02	8.7047E−02
(b) 1D	256	1.7506E−01	2.6591E−02	5.2191E+00
2D	256^2	1.6375E−01	1.9552E−02	4.3126E+00
3D	256^3	1.3840E−01	1.3533E−02	2.8773E+00

$\|E(\rho)\| = \sum_{i,j,k} |\rho^{\rm numer}_{i,j,k} - \rho^{\rm exact}_{i,j,k}| \Delta x_{i,j,k}$.

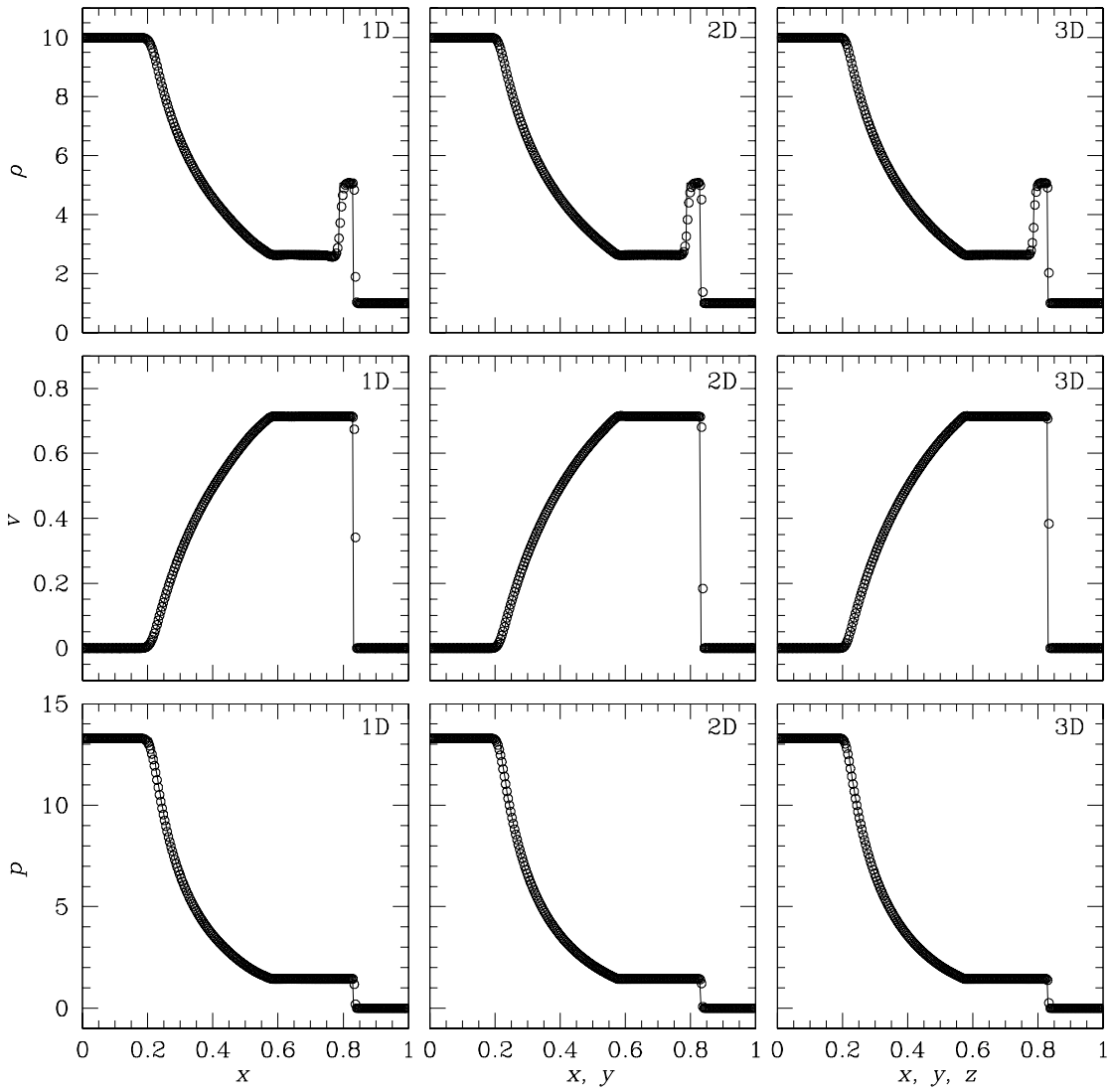

Figure 3.1: (a) 1D, 2D, and 3D mildly relativistic shock tube tests. The calculations have been done with the initial states in equation (3.1) using 256, 256^2, and 256^3 cells. The numerical solutions (open circles) and the exact solutions (solid lines) are plotted at $t = 0.4$, $0.4\sqrt{2}$, and $0.4\sqrt{3}$.

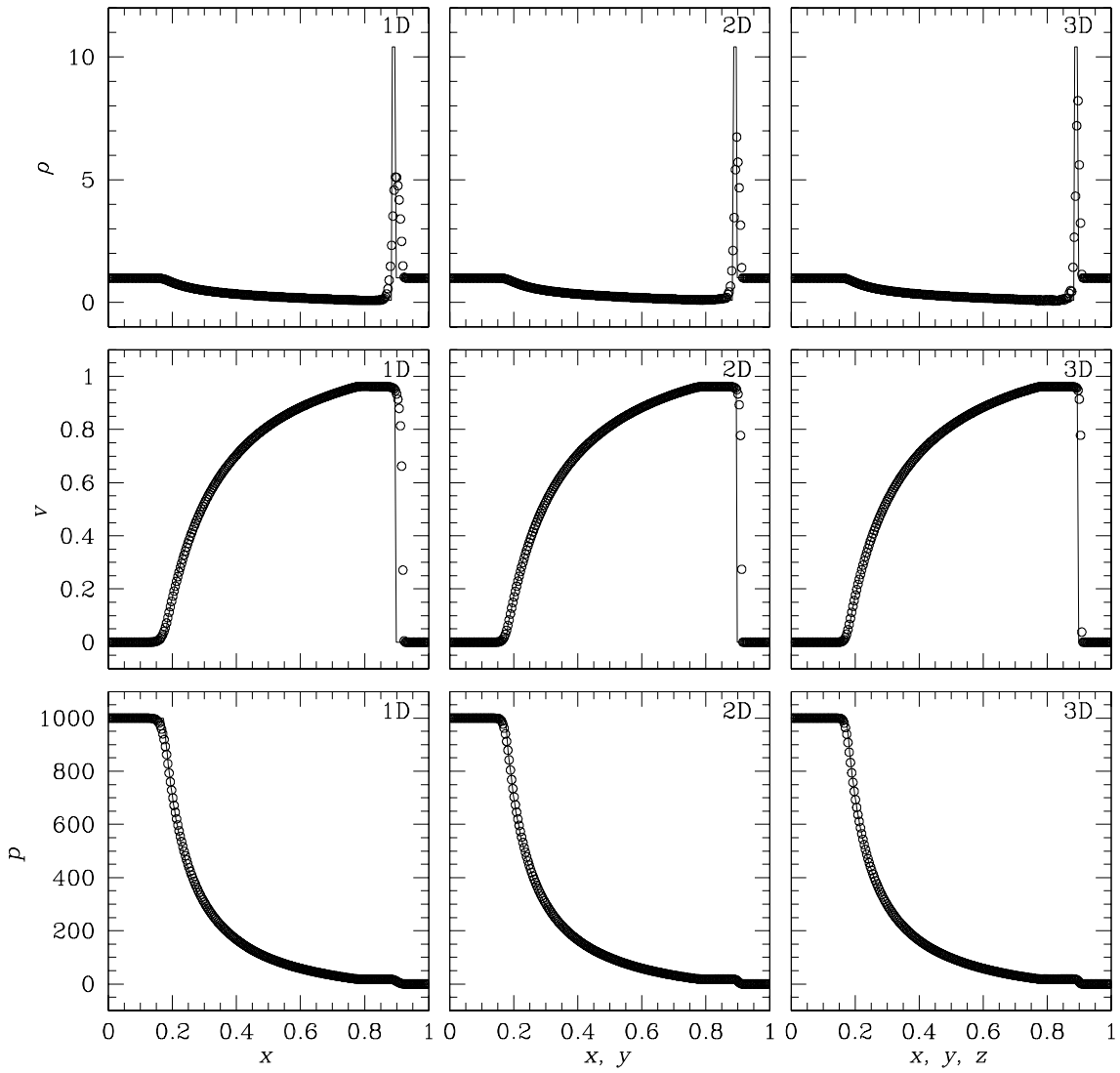

Figure 3.1: (b) 1D, 2D, and 3D highly relativistic shock tube tests. The calculations have been done with the initial states in equation (3.2) using 256, 256^2, and 256^3 cells. The numerical solutions (open circles) and the exact solutions (solid lines) are plotted at $t = 0.4$, $0.4\sqrt{2}$, and $0.4\sqrt{3}$.

3.2 Relativistic Wall Shock

A one-dimensional relativistic wall shock test has been performed in the computational box of $x = [0, 1]$. Initially a gas with extreme velocity occupying all numerical cells propagates along the x-axis against a reflecting wall placed at $x = 1$. As the gas hits the wall, it is compressed and heated and eventually a reverse shock is generated. The initial condition of this test is

$$(\rho, v_x, v_y, v_z, p) = \left(1, 0.999999, 0, 0, 10^{-4}\right) \qquad 0 \leq x \leq 1. \tag{3.3}$$

The adiabatic index $\gamma = 5/3$ is assumed and the inflow boundary condition is used at $x = 0$. This is another test which was widely used by several authors (e.g., Donat et al., 1998).

The relativistic jump condition for strong shocks with negligible preshock pressure is given by Blandford & McKee (1976)

$$v_s = -\frac{(\gamma - 1)\,\Gamma v}{\Gamma + 1}, \tag{3.4}$$

$$\rho^* = \rho \frac{\gamma\Gamma + 1}{\gamma - 1}, \tag{3.5}$$

$$v^* = 0, \tag{3.6}$$

$$p^* = \rho\,(\Gamma - 1)\,(\gamma\Gamma + 1)\,. \tag{3.7}$$

Here, v_s is the shock velocity and the superscript $*$ represents the postshock quantities, while the quantities without any superscript refer to the preshock gas.

Figure 3.2 shows the structure at $t = 0.75$ when the reverse shock is located at $x = 0.5$. The calculation has been done using 512 computational cells with a Courant constant

$C_{\rm Cour} = 0.9$ and the parameters $\varepsilon_{1,5} = 0.3$ and $\varepsilon_{2-4} = 0.1$. The numerical solution is drawn with open circles and the exact solution is represented by solid lines. The numerical and exact solutions match exactly without any oscillation or overshoot in the rest mass density, velocity, and pressure profiles.

We have calculated the mean errors in the rest mass density, velocity, and pressure for several different inflow velocities. The errors are calculated for the same time as in Figure 3.2 and given in Table 3.2. Note that the order of the mean errors is 10^{-3}, and that the accuracy does not depend systematically on the investigated Lorentz factor although the errors are smallest for the highest Lorentz factor we investigated. The mean error in the rest mass density is $\lesssim 0.5\%$ for all the Lorentz factors and about 0.25% for the maximum Lorentz factor. This accuracy is comparable to or better than that of other published upwind scheme codes.

Table 3.2: Mean errors for the relativistic wall shock tests

v	Γ	$\bar{E}(\rho)$	$\bar{E}(v)$	$\bar{E}(p)$
0.9	2.3	4.7423E−03	3.1483E−03	5.8100E−03
0.99	7.1	3.1938E−03	2.3634E−03	2.5168E−03
0.999	22.4	3.1876E−03	2.6687E−03	2.5015E−03
0.9999	70.7	5.0532E−03	4.1790E−03	3.8529E−03
0.99999	223.6	2.8425E−03	2.4914E−03	2.1466E−03
0.999999	707.1	2.4855E−03	2.0237E−03	1.8747E−03

$\bar{E}(\rho) = \sum_i |\rho_i^{\text{numer}} - \rho_i^{\text{exact}}| / \sum_i |\rho_i^{\text{exact}}|$.

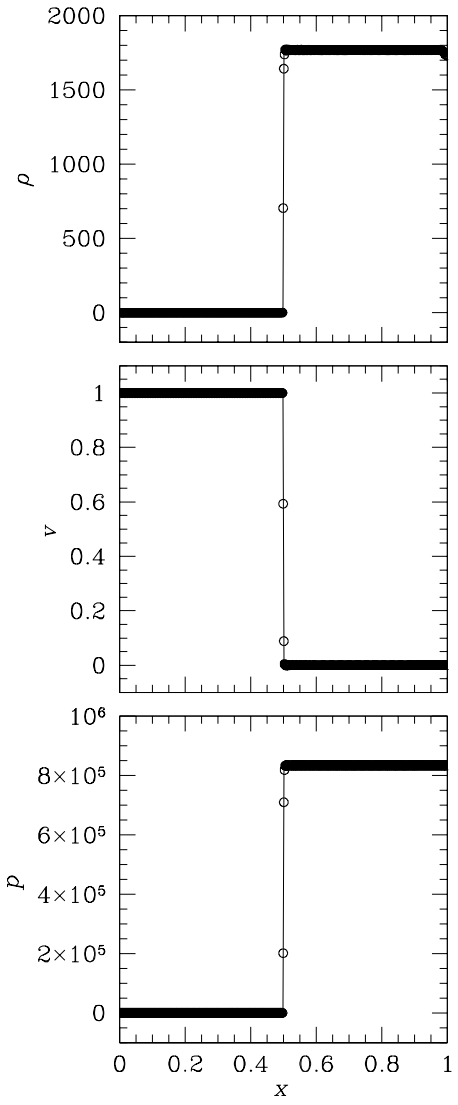

Figure 3.2: One-dimensional relativistic wall shock test. The calculation has been done with the initial states in equation (3.3) using 512 cells. The numerical solutions (open circles) and the exact solutions (solid lines) are plotted at $t = 0.75$.

3.3 Relativistic Blast Wave

The propagation of a relativistic blast wave has been tested in the two-dimensional computational box with $x = [0, 1]$ and $y = [0, 1]$. A gas of high density and pressure is initially confined in a spherical region and the subsequent explosion is allowed to evolve. This makes a spherical blast wave propagate outward. The initial condition of this test is

$$(\rho, v_x, v_y, v_z, p) = \begin{cases} (10, 0, 0, 0, 10^3) & 0 \leq \sqrt{x^2 + y^2} \leq 1/2, \\ (1, 0, 0, 0, 1) & \text{outside.} \end{cases} \tag{3.8}$$

The adiabatic index is taken to be $\gamma = 4/3$ and the reflecting and outflow boundary conditions are used.

The calculation was done using 512^2 cells with a Courant constant $C_{\rm Cour} = 0.6$ and the parameters $\varepsilon_{1,5} = 0.1$ and $\varepsilon_{2-4} = 0$. To test the symmetry properties of the code, the calculation was stopped before a reverse shock reaches the inner reflecting boundary. Figure 3.3 shows the profiles of the rest mass density, velocity, and pressure measured along the diagonal line connecting $(0, 0)$ and $(1, 1)$ at $t = 0.7$. The spherical blast wave successfully propagates to a larger radius, and we have found that all structures in it preserve the initial symmetry.

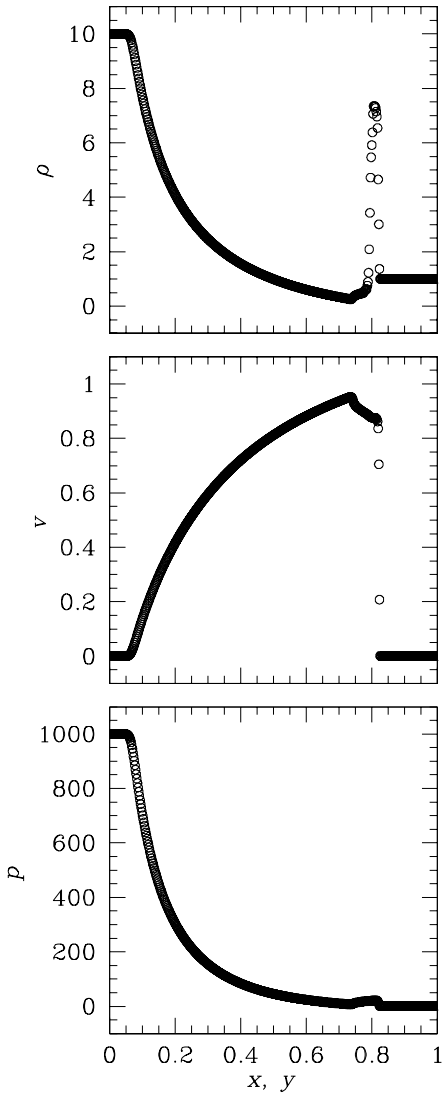

Figure 3.3: Two-dimensional relativistic blast wave test. The calculation has been done with the initial states in equation (3.8) using 512^2 cells. The numerical solutions (open circles) are plotted at $t = 0.7$.

3.4 Relativistic Hawley-Zabusky Shock

In order to test the applicability of the code to complex relativistic flows, we have performed a two-dimensional test simulation of the relativistic version of the Hawley-Zabusky shock. The test was originally suggested by Hawley & Zabusky (1989) for non-relativistic hydrodynamics. Almost the same physical values as in that original paper are used here. Initially a plane-parallel shock with a Mach number 1.2 propagates along the x-axis into two regions of different density. The regions are separated by an oblique discontinuity with an inclination of 30° with respect to the x-axis. The density jumps three times across the discontinuity. The initial configuration is summarized as

$$(\rho, v_x, v_y, v_z, p) = \begin{cases} (1, 0.6, 0, 0, 0.48) & 0 \le x \le 1/16,\ 0 \le y \le 1, \\ (1, 0, 0, 0, 0.48) & 1/16 < x \le \sqrt{3}y + 1/4,\ 0 \le y \le 1, \\ (3, 0, 0, 0, 0.48) & \text{outside.} \end{cases} \tag{3.9}$$

The adiabatic index $\gamma = 1.4$ is used. Inflow and outflow conditions are used at the x-boundaries and reflecting conditions are used at the y-boundaries.

The simulation has been done in the two-dimensional computational box with $x = [0, 8]$ and $y = [0, 1]$ using a uniform numerical grid of 2048×256 cells. A Courant constant $C_{\rm Cour} = 0.9$ and the parameters $\varepsilon_{1,5} = 0.1$ and $\varepsilon_{2-4} = 0$ were used. We have simulated this test until $t = 20$ in order to see the long term evolution. The passage of the planar shock through the discontinuity causes the Kelvin-Helmholtz instability to occur along the discontinuity and leads to the formation of vortices. The vortices roll up, interact, and merge during the simulation; the detailed morphology and the number of vortices formed

are somewhat sensitive to numerical resolution. Figure 3.4 shows the gray-scale images of the rest mass density at different times (t = 2, 11, and 20). Because all the structures are dragged to the right boundary as time goes on, only the left, middle, and right halves of the computational box are shown at t = 2, 11, and 20, respectively. The vortices along the discontinuity are clearly formed and overall the morphology is similar to that of the non-relativistic simulation.

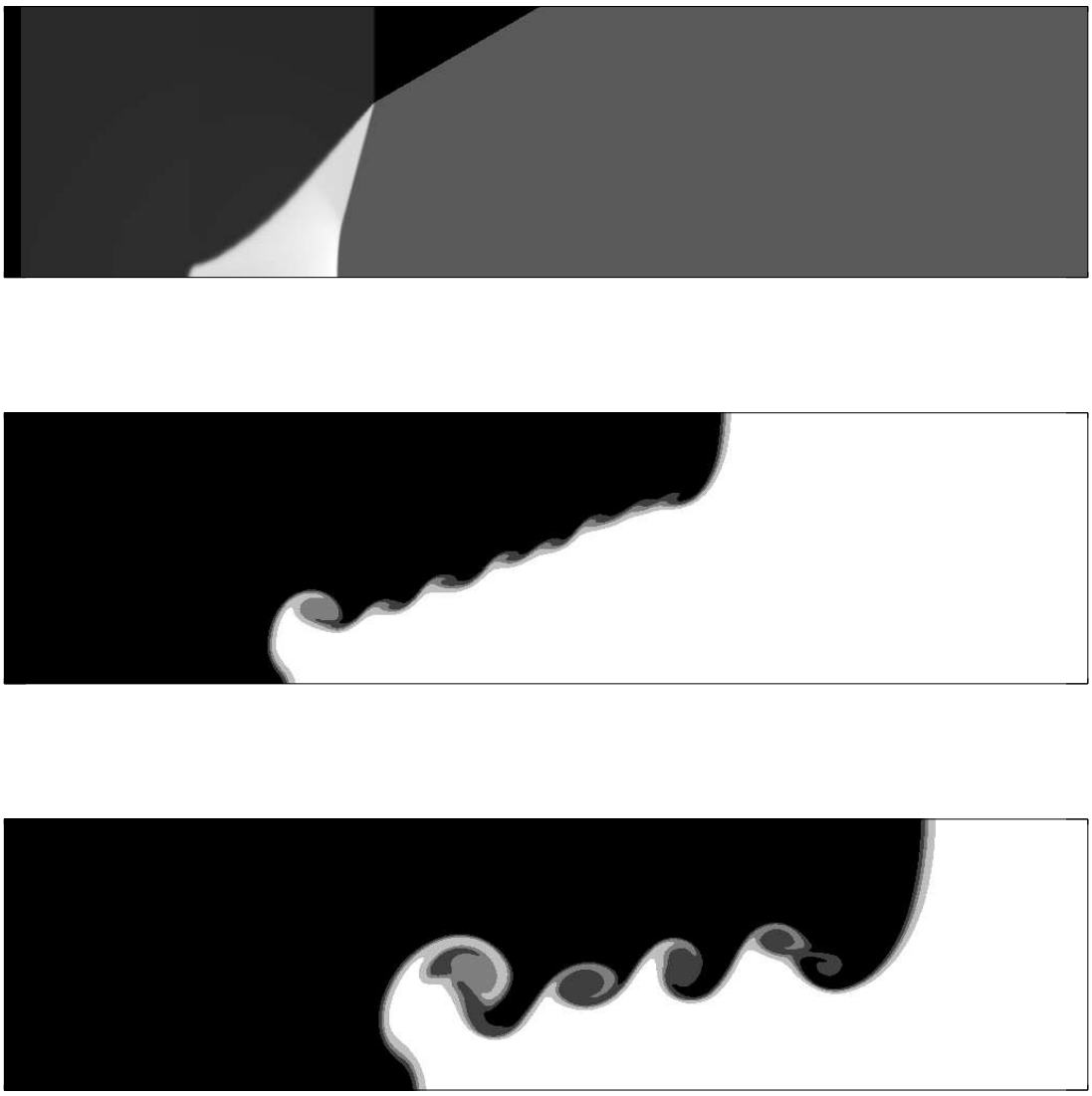

Figure 3.4: Two-dimensional relativistic Hawley-Zabusky shock. The simulation has been carried out with the initial configurations in equation (3.9) using 2048×256 cells. Gray-scale images show the rest mass density at $t = 2$, 11, and 20 (top to bottom), using linear scales that range from 1.0 (black) to 6.75 (white). Only the half of the computational box is shown in each panel.

3.5 Relativistic Extragalactic Jets

Finally, in order to test the applicability of the code to realistic relativistic flows, we have simulated a two-dimensional relativistic extragalactic jet propagating into a homogeneous medium. The relativistic jet inflows with a velocity 0.99 to the computational box of $x = [0, 4]$ and $y = [0, 1]$. The jet has initially a radius of 1/8 (32 cells) and Mach number 8.76. The density ratio of the jet to the ambient medium is 0.1 and the pressure of the jet is in equilibrium with that of the ambient medium. The initial condition for jet inflow and ambient medium is summarized as

$$(\rho, v_x, v_y, v_z, p) = \begin{cases} (1, 0.99, 0, 0, 0.1) & 0 \le x \le 1/32,\ 0 \le y \le 1/8, \\ (10, 0, 0, 0, 0.1) & \text{outside.} \end{cases} \tag{3.10}$$

The adiabatic index $\gamma = 4/3$ is used. The inflow and outflow conditions are used at the x-boundaries and the reflecting and outflow conditions are used at the y-boundaries.

The simulation has been done using a uniform numerical grid of 1024×256 cells with a Courant constant $C_{\rm Cour} = 0.3$ and the parameters $\varepsilon_{1,5} = 0.3$ and $\varepsilon_{2-4} = 0.1$. Figure 3.5 shows the gray-scale images of logarithm of the rest mass density, pressure, and Lorentz factor at $t = 5$ when the bow shock reaches the right boundary. We can clearly see the dominant structures of bow shock, working surface, contact discontinuity, and cocoon. It is clear that the internal structure of the relativistic jet is less complex compared to that of a non-relativistic jet due to the effects of high Lorentz factor. The overall morphology and dynamics of our simulation match roughly with those of previous works, e.g., Duncan & Hughes (1994), although the initial conditions and the plotted epoch are different.

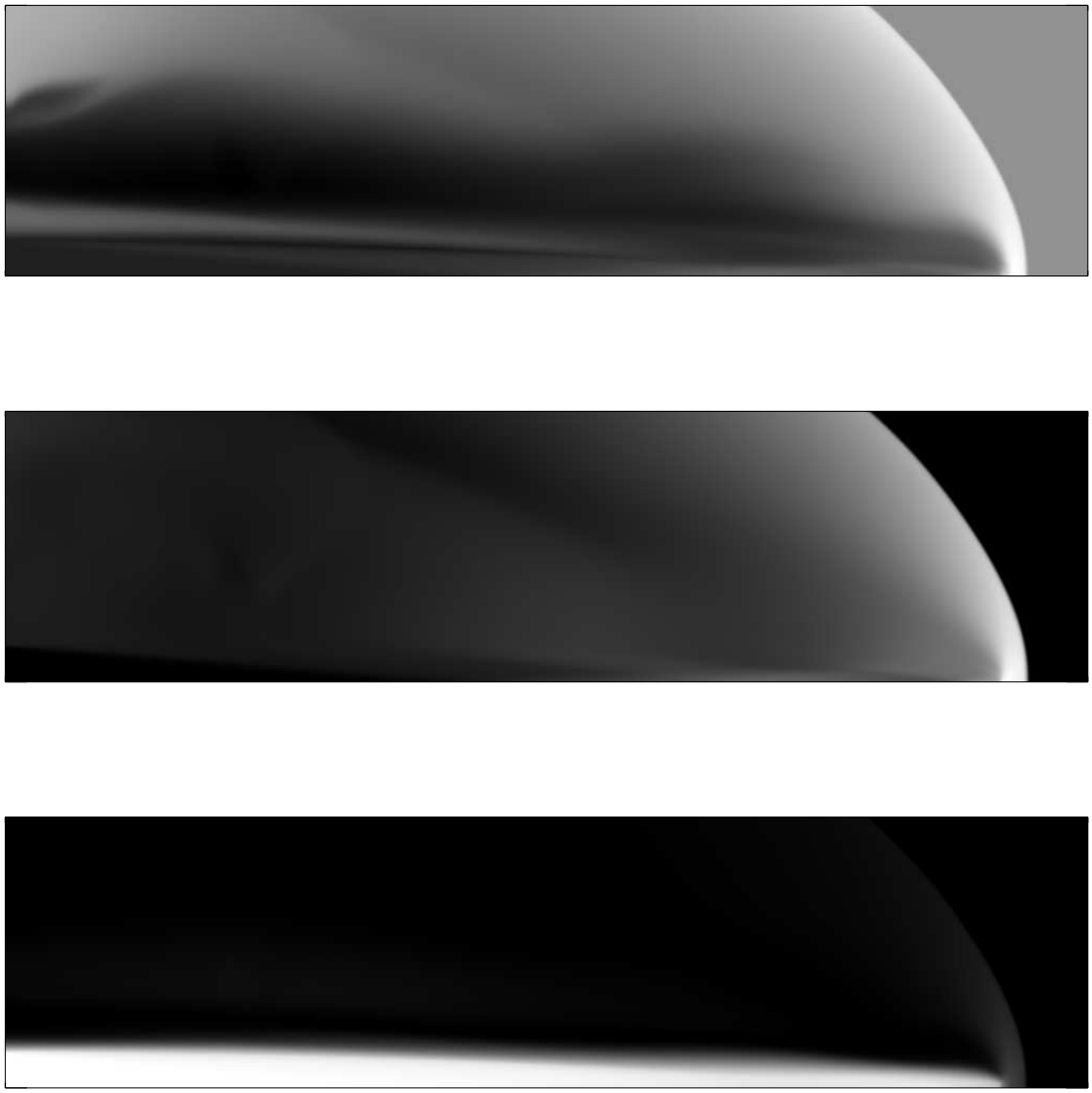

Figure 3.5: Two-dimensional relativistic extragalactic jet. The simulation has been carried out with the initial conditions in equation (3.10) using 1024×256 cells. Gray-scale images show logarithms of the rest mass density, pressure, and Lorentz factor (top to bottom) at $t = 5$, using logarithmic scales that range from -0.28 (black) to 1.98 (white) for log(density), -1.0 (black) to 1.30 (white) for log(pressure), and 0 (black) to 0.85 (white) for log(Lorentz factor).

Chapter 4

Problem Description

We now consider the three-dimensional interactions of relativistic jets with two-phase ambient media. These jets propagate through a denser ambient gas and then hit spherical clouds with densities higher than that of the ambient gas. The initial ratio of the cloud density, ρ_c, to the ambient medium density, ρ_a, and that of the beam density, ρ_b, to the ambient medium density, are respectively defined as

$$\chi \equiv \frac{\rho_c}{\rho_a}, \quad \eta \equiv \frac{\rho_b}{\rho_a}. \tag{4.1}$$

If we neglect complicating effects including radiative cooling and gravity and consider only hydrodynamic effects, then this problem can be relatively simple and depends only on a few hydrodynamic variables, the Mach number of the jet and the initial density contrasts given in equation (4.1). Any geometric effects, such as different impact zone sizes or cloud shapes, certainly will make differences in the evolutions of jets and clouds, but the overall dynamical evolutions should not be very sensitive to them. Thus, we focus on the evolutions of jets and clouds influenced by the above hydrodynamic effects.

The approximate propagation velocity of the jet through the homogeneous ambient medium can be obtained by the conservation of the momentum flux of the beam and ambient medium in the reference frame of the Mach disk (Martí et al., 1997). Assuming pressure equilibrium between the beam and the ambient medium, the conservation of the momentum

flux is $\rho_b h_b \Gamma_b'^2 v_b'^2 = \rho_a h_a \Gamma_a'^2 v_a'^2$, with the following relations, $v_b' = (v_b - v_a)/(1 - v_b v_a)$, $\Gamma_b' = \Gamma_b \Gamma_a (1 - v_b v_a)$, $v_a' = -v_a$, and $\Gamma_a' = \Gamma_a$. Here h is the specific enthalpy and v' and Γ' represent, respectively, velocity and Lorentz factor measured in the reference frame of the Mach disk, while v and Γ indicate those measured in the rest frame of the ambient medium. The subscripts b and a stand for the beam and the ambient medium, respectively. After substituting for the primed variables in terms of the unprimed ones, the conservation of the momentum flux is derived to be

$$\rho_b h_b \Gamma_b^2 (v_b - v_a)^2 = \rho_a h_a v_a^2. \tag{4.2}$$

Then the one-dimensional jet advance velocity, estimated in the rest frame of the ambient medium is

$$v_a = \frac{v_b}{\sqrt{1/\eta^*} + 1}, \tag{4.3}$$

where η^* is given by

$$\eta^* = \Gamma_b^2 \frac{\rho_b h_b}{\rho_a h_a}. \tag{4.4}$$

In the nonrelativistic limit ($h \to 1$, $\Gamma \to 1$), η^* approaches η, so that v_a represents the classical jet advance velocity through the ambient medium, i.e., $v_a = v_b/(\sqrt{1/\eta} + 1)$.

Based on this jet advance velocity, we define the dynamical timescale called the *beam crossing time*, $t_{\rm bc}$,

$$t_{\rm bc} \equiv \frac{2 r_c}{v_a}, \tag{4.5}$$

as the time taken for the beam to sweep a distance across the ambient medium equal to the diameter of a cloud with radius r_c. Since the timescale $t_{\rm bc}$ depends only on a single

variable v_a (for fixed cloud radius), it is extremely useful in comparing and characterizing the dynamical evolutions of both jets and clouds with different model parameters.

Although we use the beam crossing time as the primary timescale in this study, it is also interesting to estimate the cloud crushing and cooling timescales. The cloud crushing timescale is the time required for the beam to cross the cloud diameter during the phase of cloud compression, and if v_a is nonrelativistic, this timescale can be approximated as $t_{\rm cc} \sim 2\sqrt{\chi} r_c/v_a$ (Klein et al., 1994). Clearly, $t_{\rm bc} \simeq t_{\rm cc}$ in the absence of clouds, and for dense clouds ($\chi \gg 1$), $t_{\rm bc} < t_{\rm cc}$. Following Fragile et al. (2004), the cloud cooling timescale can be roughly estimated from $t_{\rm cool} \sim C v_a^3/(\chi^{3/2}\rho_c)$, where the constant $C = 7.0 \times 10^{-35}$ g cm^{-6} s^4. With values reasonable for kpc-scale extragalactic situations, $r_c = 1$ kpc, $v_a = 0.1c$, $\chi = 100$, and $\rho_c = 10^2 m_{\rm H}$ cm^{-3}, we find that $t_{\rm bc} < t_{\rm cc} \sim t_{\rm cool}$. Thus cooling will not be extremely important during the cloud compression phase for the chosen values. For fixed density and cloud radius, the cloud cooling timescale becomes longer compared to the cloud crushing timescale as v_a increases, so the effect of cooling is somewhat reduced for relativistic jets compared to nonrelativistic ones. For parameters more relevant to VLBI-scale jet/cloud collisions, $r_c = 0.5$ pc, $v_a = 0.5c$, $\chi = 10^4$, and $\rho_c = 10^6 m_{\rm H}$ cm^{-3}, we have $t_{\rm bc} \sim t_{\rm cool} < t_{\rm cc}$, so cooling would be more important in this case. A more detailed consideration of cooling timescales is beyond the scope of this study.

Three distinct evolutionary stages can be considered in this problem. There is an initial jet propagation stage where the jet advances through a homogeneous ambient medium with velocity v_a. Once a jet is launched, a bow shock propagates into the ambient medium; this is followed by a Mach disk shock in the beam which is quickly established during this stage. When the jet strikes the cloud, the jet transmits a shock into the cloud. If the

cloud/ambient density contrast is sufficiently large and the jet speed is relatively slow, the speed of the transmitted shock in the cloud is much slower than that of the bow shock of the jet. Thus the bow shock entirely encloses the cloud, which leads to the development of the Kelvin-Helmholtz instability at the cloud surface (e.g., Klein et al., 1994). The final stage is when the jet passes through the cloud. In this phase the cloud begins to reexpand just after the jet reaches the rear edge of the cloud. At the same time, the jet propagates in the original direction if it has dominated the cloud or in a new direction if the cloud was massive enough to deflect the jet.

Chapter 5

Numerical Simulations

5.1 Basic Equations

The special relativistic hydrodynamic equations are written in a covariant form (e.g., Landau & Lifshitz, 1959; Wilson & Mathews, 2003)

$$\partial_\alpha \left(\rho U^\alpha\right) = 0, \tag{5.1}$$

$$\partial_\alpha T^{\alpha\beta} = 0, \tag{5.2}$$

where the energy momentum tensor is given by

$$T^{\alpha\beta} = \left(e + p\right) U^\alpha U^\beta + p g^{\alpha\beta}. \tag{5.3}$$

Here, $\partial_\alpha = \partial/\partial x^\alpha$ is the covariant derivative with spacetime coordinates $x^\alpha = [t, x_j]$, $U^\alpha = [\Gamma, \Gamma v_j]$ is the normalized ($U^\alpha U_\alpha = -1$) four-velocity vector, and a metric tensor $g^{\alpha\beta}$ with a signature +2 is used. The mass density, velocity, internal plus mass energy density, and pressure in the local rest frame are denoted by ρ, v_j, e, and p, respectively. Greek indices (e.g., α, β) denote the spacetime components while Latin indices (e.g., i, j) indicate the spatial components, and the speed of light is still set to unity ($c \equiv 1$).

For our numerical purposes, it is convenient to rewrite the covariant equations (5.1)–(5.3) in the index form which gives a hyperbolic system of conservation equations

$$\frac{\partial D}{\partial t} + \frac{\partial}{\partial x_j}\left(D v_j\right) = 0, \tag{5.4}$$

$$\frac{\partial M_i}{\partial t} + \frac{\partial}{\partial x_j}\left(M_i v_j + p\delta_{ij}\right) = 0, \tag{5.5}$$

$$\frac{\partial E}{\partial t} + \frac{\partial}{\partial x_j}\left[\left(E + p\right) v_j\right] = 0, \tag{5.6}$$

where the equation of state (EOS) is given by

$$p = \left(\gamma - 1\right)\left(e - \rho\right). \tag{5.7}$$

Here, D, M_i, and E are the mass density, momentum density, and total energy density in the reference frame, respectively, and γ is the adiabatic index. We note that we restrict ourselves to an ideal gas EOS in this study although future expansions of this work could use a more general EOS (e.g., Ryu et al., 2006).

The quantities in the reference frame are related to those in the local rest frame via following transformations

$$D = \Gamma\rho, \tag{5.8}$$

$$M_i = \Gamma^2\left(e + p\right) v_i, \tag{5.9}$$

$$E = \Gamma^2\left(e + p\right) - p, \tag{5.10}$$

where the Lorentz factor is given by

$$\Gamma = \frac{1}{\sqrt{1 - v^2}} \tag{5.11}$$

with $v^2 = v_x^2 + v_y^2 + v_z^2$.

If an EOS is assumed, the local sound speed, c_s, and the specific enthalpy, h, are easily derived. For an ideal gas, they are given by

$$c_s^2 = \frac{1}{h}\frac{\partial p}{\partial \rho} + \frac{\partial p}{\partial e}, \quad h = 1 + \frac{\gamma}{\gamma - 1}\frac{p}{\rho}. \tag{5.12}$$

A γ-law gas such as an ideal gas has the local sound speed limit $c_s \leq \sqrt{\gamma - 1}$. Only in the ultrarelativistic case, $e \gg \rho$, does the local sound speed approach its limit (i.e., $c_s \to \sqrt{\gamma - 1}$).

5.2 Numerical Method and Setup

The system of equations (5.4)–(5.7) can be solved numerically with explicit finite difference upwind schemes which are based on exact or approximate Riemann solvers using the characteristic decomposition of these relativistic hydrodynamic conservation equations. Although the upwind schemes were originally developed for nonrelativistic hydrodynamics, some schemes have been extended to special relativistic hydrodynamics while retaining the advantages of the upwind schemes, including high accuracy and robustness.

We use the multidimensional code for solving the special relativistic hydrodynamic equations as a hyperbolic system of conservation laws based on the total variation diminishing (TVD) scheme (Harten, 1983) which was discussed in Chapters 2 and 3. The TVD scheme

is an explicit Eulerian finite difference upwind scheme and an extension of the Roe scheme to second-order accuracy in space and time. Our code uses a new set of conserved quantities, which lead to a new eigenstructure for special relativistic hydrodynamics and employs an analytic formula for the calculation of the local rest frame quantities from the reference frame quantities. The advantage of our code is that it is simple and fast, and yet it is accurate and reliable enough. The performance of the code was demonstrated through several standard tests, including relativistic shock tubes, a relativistic wall shock, and a relativistic blast wave, as well as test simulations of the relativistic version of the Hawley-Zabusky shock and a relativistic extragalactic jet given in Chapter 3. For our new simulations, we have parallelized this code using the message-passing interface (MPI) and this parallelized version is given in the Appendix. The simulations described here typically use 64 processors on a Linux cluster.

Table 5.1 lists the initial parameters of the four different relativistic jet-cloud interaction models we have investigated in this study. All models use the adiabatic index $\gamma = 4/3$ and assume pressure-matched jets and clouds, i.e., $p_b/p_a = p_c/p_a = 1$, where p_b, p_c, and p_a are the pressure of the beam, cloud, and ambient medium, respectively. We set $c = r_c = \rho_a \equiv 1$ in our models, so that all physical quantities are dimensionless and can be scaled to any specific physical model (e.g., $t \rightarrow tc/r_c$, $\rho \rightarrow \rho/\rho_a$). The initial Newtonian beam Mach number, $\mathcal{M}_b^N \equiv v_b/c_{s,b}$, where $c_{s,b}$ is the sound speed in the beam, as well as the initial relativistic beam Mach number, $\mathcal{M}_b^R \equiv (\Gamma_b/\Gamma_{s,b})\mathcal{M}_b^N$, where $\Gamma_{s,b}$ is the Lorentz factor associated with the beam sound speed are given in Table 5.1. As discussed in Königl (1980), in the context of relativistic gasdynamics the relativistic Mach number is the best analog of the Newtonian one for nonrelativistic flows, so we usually use this relativistic beam Mach number to describe

physical properties in our models. The initial density contrast between the beam and the ambient medium is fixed to $\eta = 0.1$, so that jets strike clouds with densities 100 to 1000 times higher than the initial beam density. While even smaller values of η would have been more realistic for most extragalactic jets, they lead to extremely short time steps when the jets strike the clouds, making them computationally unachievable. Previous studies have indicated that most important properties for fast jets are relatively insensitive to values of $\eta \lesssim 0.1$ (e.g., Rosen et al., 1999). In models M1 and M2, the clouds interact with the lower relativistic beam Mach number jets, so the relativistic effects are less dominant, with smaller beam velocities and internal energies. Model M1 is identical to model M2 except for the smaller density ratio of the cloud to the ambient medium. Models M3 and M4 have been chosen to study the cloud interactions with jets with higher relativistic beam Mach numbers, which have more dominant relativistic effects caused by larger beam velocities and internal energies. Again, the initial conditions of model M3 are the same as those of model M4 except for the smaller density ratio of the cloud to the ambient medium.

We set up the density gradient of the spherical cloud edge with a hyperbolic tangent function

$$\rho\left(r\right) = \frac{\rho_c + \rho_a}{2} + \frac{\rho_c - \rho_a}{2} \tanh\left(\frac{r_c - r}{\Delta r}\right), \tag{5.13}$$

where r is the distance from the center of the cloud and Δr is the scale parameter for the width of density transition ($\Delta r \ll r_c$). The presence of a true density discontinuity instead of this steep function would not affect the actual dynamics of jet-cloud interactions significantly, but the discontinuous cloud edge is approximated by this somewhat smoothed density profile to avoid numerical artifacts as the jet impacts the cloud. We assume that

other physical quantities such as pressure and velocity are constant across the transition width.

The simulations have been performed in the three-dimensional computational domain with $x = [0, 8]$, $y = [0, 8]$, and $z = [0, 8]$ using a uniform Cartesian grid of 256^3 cells. The beam, with a circular cross section of radius $r_b = 1/4$ (8 cells), is initially located at $(x, y, z) = (0, 4, 4)$ and propagates through the ambient medium along the positive x-direction. In order for the relativistic jet to collide off axis with the cloud at rest, the center of the cloud, with radius $r_c = 1$ (32 cells), is placed at $(x, y, z) = (4, 3.5, 4)$; hence the relativistic jet hits the spherical cloud with an impact angle of 30°. The outflow boundary condition is imposed on all boundaries of the computational domain except where the inflow boundary condition is used to maintain the continuous jet. We were able to assign the relativistic jet only 8 cells per initial beam radius and the cloud 32 cells per initial cloud radius due to the limitation of computational resources. This resolution is less than that of previously reported two-dimensional works which are related to this problem. Thus our three-dimensional simulations may not be fully converged and some quantities to be described may change if three-dimensional simulations with much higher resolutions can be performed in future studies; however, our tests of the code do indicate that simulations with this level of resolution should be reasonably accurate (see Chapter 3).

Table 5.1: Simulation parameters

Model	χ	η	v_b	Γ_b	$\mathcal{M}_b^N$	$\mathcal{M}_b^R$	$t_{\rm bc}$	$t_{\rm end}$
M1	10	0.1	0.9	2.29	2.92	6.36	4.86	$4t_{\rm bc}$
M2	100	0.1	0.9	2.29	2.92	6.36	4.86	$6t_{\rm bc}$
M3	10	0.1	0.99	7.09	1.92	11.6	2.50	$4t_{\rm bc}$
M4	100	0.1	0.99	7.09	1.92	11.6	2.50	$5t_{\rm bc}$

Here χ is the ratio of the cloud density to the ambient medium density, η is the ratio of the beam density to the ambient medium density, v_b is the initial beam velocity, Γ_b is the beam Lorentz factor, $\mathcal{M}_b^N$ is the Newtonian beam Mach number, $\mathcal{M}_b^R$ is the relativistic beam Mach number, $t_{\rm bc}$ is the beam crossing time, and $t_{\rm end}$ is the time at which the simulation is ended.

Chapter 6

Results

6.1 Morphology and Dynamics

The gray-scale images in Figures 6.1(a)–(d) show the distinct evolutionary phases of models M1–M4, respectively. These images show the $x-y$ plane with $z=4$ in the three-dimensional computational domain, thus providing a slice through the center of the jet and cloud. In each of these figures the top to bottom panels represent density, pressure, and Lorentz factor, respectively (in logarithmic scales) while the left to right panels represent evolutionary stages shown at three different times, $t=t_{\rm bc}$, $(t_{\rm bc}+t_{\rm end})/2$, and $t_{\rm end}$.

The early stages of the relativistic jet propagation through the uniform ambient medium until the jet is about to collide the cloud ($t/t_{\rm bc}\sim 1$) are basically similar to those found in earlier simulations (e.g., Martí et al., 1997; Aloy et al., 1999b). Several key features are clear from the left panels of Figures 6.1(a)–(d). In all the models a bow shock that separates the jet from the external medium is driven, the beam itself is terminated by a Mach disk (terminal shock) where most of the beam kinetic energy is converted into its internal energy, and shocked jet material flows backward into a cocoon within the contact discontinuity that separates the shocked external gas and the shocked jet gas. There is no difference between models M1 and M2 and between models M3 and M4 at this stage because of the same initial conditions of the jets and the same ambient media properties for these two pairs of simulations.

The relativistic beam Mach number of the jet is associated with the shape of the bow shock. In models M1 and M2, the lower relativistic beam Mach number jets, with a lower propagation velocity ($v_a \sim 0.42$) and internal energy, have bow shocks with narrower conical shapes, and the Mach disk is quite close to the bow shock. This conical shape of the bow shock tends to be broader as the relativistic beam Mach number of the jet increases, as seen for models M3 and M4; these higher relativistic beam Mach number jets, with a higher propagation velocity ($v_a \sim 0.78$) and internal energy, also have the Mach disk standing off farther from the bow shock. The shapes of the bow shocks are also connected with the sizes of the impact cross section when the jets begin to interact with the cloud. The low relativistic beam Mach number jets in models M1 and M2 feature relatively thick cocoons while the high relativistic beam Mach number jets in models M3 and M4 have thinner cocoons. This dependence of the cocoon morphology on the relativistic beam Mach number is consistent with previous results (see e.g., Martí et al., 1997). Although the structural differences in the jet head and the cocoon are evident by this early stage of the evolution, internal structures within the beam and backflows are not yet dominant and are barely distinguishable at this stage.

In every model, the relativistic jet begins to partially deflect as a direct response to its interactions with the clouds. This is seen in the middle column of panels of Figures 6.1(a)–(d), and most clearly visible in the middle bottom panels where fast streams emerge from the Mach disk at significant angles with respect to the jet axis. These deflection features are stronger in models M2 and M4, which have higher ratio of the density of the cloud to that of the ambient medium ($\chi = 100$). The deflection angles of the portion of the post-Mach shock flows with respect to the beam propagation axis are very time-dependent. In our models

these angles peak when the jets cross over approximately half the clouds (at $t/t_{\rm bc} \sim 2.5$, 3.5, 2.5, and 3 for models M1–M4, respectively). For these comparable dynamical times, models M2 and M4, both with $\chi = 100$ but having different beam Mach numbers, show $80^\circ - 90^\circ$ deflection angles, while models M1 and M3, with the same beam Mach numbers as the models M2 and M4, respectively, but with $\chi = 10$, show smaller deflection angles of about 45°. This indicates that the deflection angle is more strongly influenced by the density contrast, χ, than by the beam Mach number of the jet. For an off-axis collision there are weak deflection features on the other side of the jet axis, where the deflection of the outflow from the beam is significantly suppressed by the dense cloud. That suppression leads to the production of a strong oblique shock within the beam. As seen in the figures, the oblique shocks are quite strong in models M2 and M4, but in models M1 and M3 there are only relatively weak oblique shocks in the beam. Comparing at this stage models M1 and M3 with models M2 and M4, we note that the bow shocks enclose less of the cloud in models M1 and M3 because of their lower density contrast, χ. That implies quicker penetration of the clouds by these jets, so the strengths of the oblique shocks in these beams are reduced.

Some additional properties of the simulations at this stage are shown in Figures 6.2 and 6.3. Figure 6.2 illustrates one-dimensional flow structures of density, pressure, and Lorentz factor along the beam propagation axis for models M1 and M3 at the same epochs as in Figures 6.1(a) and (c). In both models there are spikes in the density and pressure associated with the impact by the incident jets, while there is little change in beam Lorentz factor. Figure 6.3 shows the images of the logarithm of the Lorentz factor projected at the viewing angle of 0° for models M2 and M4, at $t/t_{\rm bc} = 3.5$ and 3, respectively. These projection images clearly show the anisotropic distribution and directions of the deflected

gas induced by the jet. This gas is an admixture of jet and cloud material, but only a small fraction of the cloud gas is shown in these projection images since the mean cloud velocity computed in each component (refer to Section 6.2) is $\langle v_i \rangle \lesssim 0.01$ and 0.06 for models M2 and M4, respectively, at the same epoch as in Figure 6.3. This implies that this deflected gas consists predominantly of jet material although small amounts of cloud material are entrained in these deflected structures. This presence of deflected gas accelerating toward a terminal velocity strongly suggests that such deflected and accelerated gas is responsible for at least some of the outflowing gas observed in the vicinity of AGNs.

Once the jet passes through the cloud, it begins to accelerate, causing a change in the shape of the bow shock. As visible in the right panels of Figures 6.1(a)–(d) shown when the jet head nearly reaches the boundary of the computational cube (at $t/t_{\rm bc} = 4$, 6, 4, and 5 for models M1–M4, respectively), the shapes of the bow shocks change more clearly in the low relativistic beam Mach number jets than in the high relativistic beam Mach number jets. That reflects the fact that the acceleration of the jets is somewhat faster in low relativistic beam Mach number jets. That reacceleration occurs in essentially the original propagation direction or in a somewhat new direction. In our simulations there is a trend for the flow of the jet to be bent more when a lower relativistic beam Mach number jet interacts with a denser cloud, with the least bending seen for model M3 and the most for model M2. We see in the right panels of Figure 6.1(b) that the beam is bent by about 10° with respect to the original jet axis. The bent jet still remains stable and collimated over the several dynamical times we could follow its development.

After the jet head passes the cloud, the amount of strongly deflected gas gradually reduces and the oblique shocks continue to develop in the beam. These oblique shocks are

unlikely to play a major role in slowing the jets because we do not find any significant deceleration features during this stage. Although a significant portion of the momentum flux of the jets is transferred to the deflected gas and the cloud through the collision events, the jets in our simulations are still stable and well collimated over several dynamical times after collisions even if the jet is bent. This stable, collimated condition is quantitatively apparent in the flow structures of density, pressure, and Lorentz factor shown in Figure 6.2 at $t/t_{\rm bc} = 4$ (corresponding to the dashed lines) for models M1 and M3, respectively. There are only slight fluctuations in the flow structures at these late stages.

In comparing our simulations with hydrodynamic simulations of nonrelativistic jet-cloud interactions (de Gouveia Dal Pino, 1999; Higgins et al., 1999; Wang et al., 2000) we can only note some fairly basic similarities and differences between our relativistic models and the roughly corresponding nonrelativistic models. The lack of good overlap between the $\mathcal{M}_b^R$ values for the relativistic jets and the standard Mach number for the nonrelativistic jets as well as differences between cloud size to jet-width ratios considered here and in that earlier nonrelativistic work prevents us from making quantitative comparisons. Relativistic jets interacting with dense clouds certainly do show general morphological features such as deflections of some gas and bent structures of jets similar to those found in some of the nonrelativistic jet-cloud interactions. The slower relativistic jet shows a bent structure after interaction, which is similar to that found in nonrelativistic simulations involving "weak" jets while the faster relativistic jet effectively plows through the clouds. Higher power nonrelativistic jets also can plow through, and apparently completely destroy, clouds. However, some major differences arise because of the larger propagation velocity of the relativistic jets. Because of this, moderately light ($\eta = 0.1$) relativistic jets are not effectively decelerated and

disrupted by the dense ($\chi = 100$) clouds, whereas nonrelativistic jets assaulting clouds of similar density ratios typically are disrupted. Our relativistic jets are rather reaccelerated in either a slightly new or essentially the original direction after their interactions with clouds. The large propagation velocity also suppresses the development of hydrodynamic instabilities in the jets, so that the jets still remain stable and collimated even after the jets smash into much denser clouds. In addition, as discussed in Section 6.2, clouds impaled by relativistic jets also appear to survive somewhat better than do those hit by strong nonrelativistic jets.

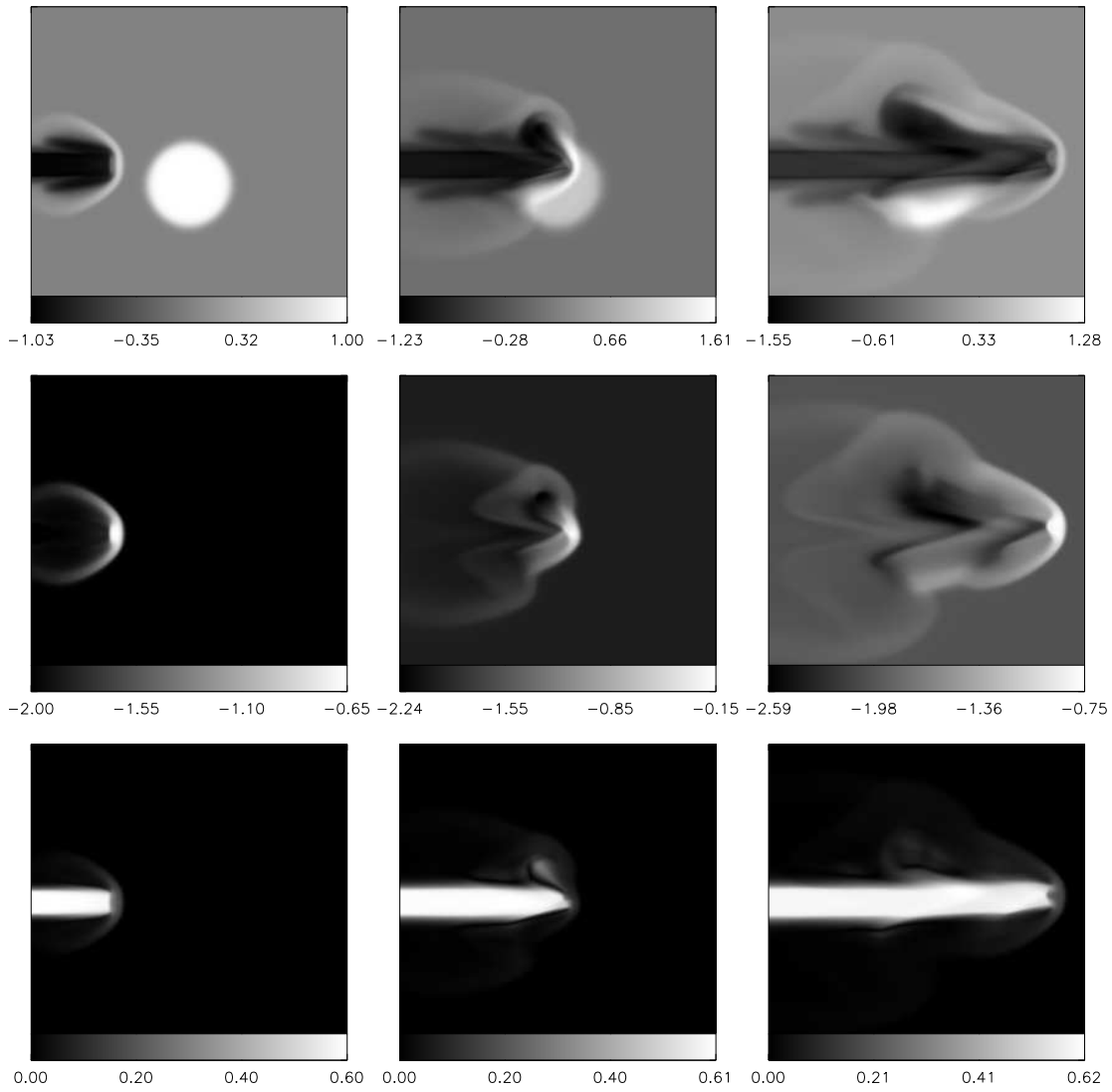

Figure 6.1: (a) Gray-scale images of density, pressure, and Lorentz factor (top to bottom) for model M1 at three different times, $t/t_{\rm bc} = 1$, 2.5, and 4 (left to right). The image scales are logarithmic for ρ and p but the square-root of the logarithm for Γ so as to enhance visibility of intermediate values; the images show the $x - y$ plane with $z = 4$ in the three-dimensional computational domain.

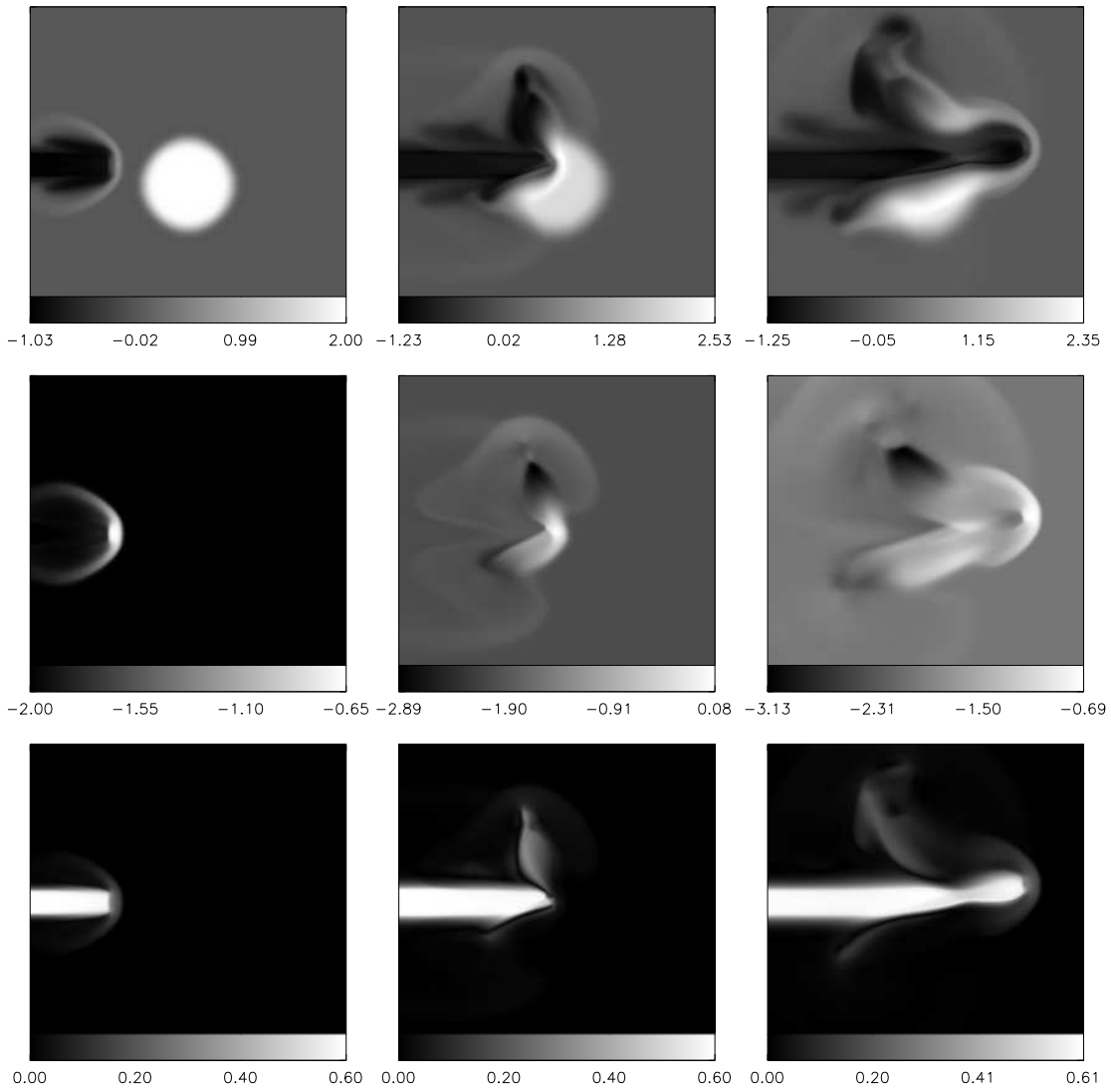

Figure 6.1: (b) Same as Fig. 6.1(a) except for model M2 at $t/t_{\rm bc} = 1$, 3.5, and 6.

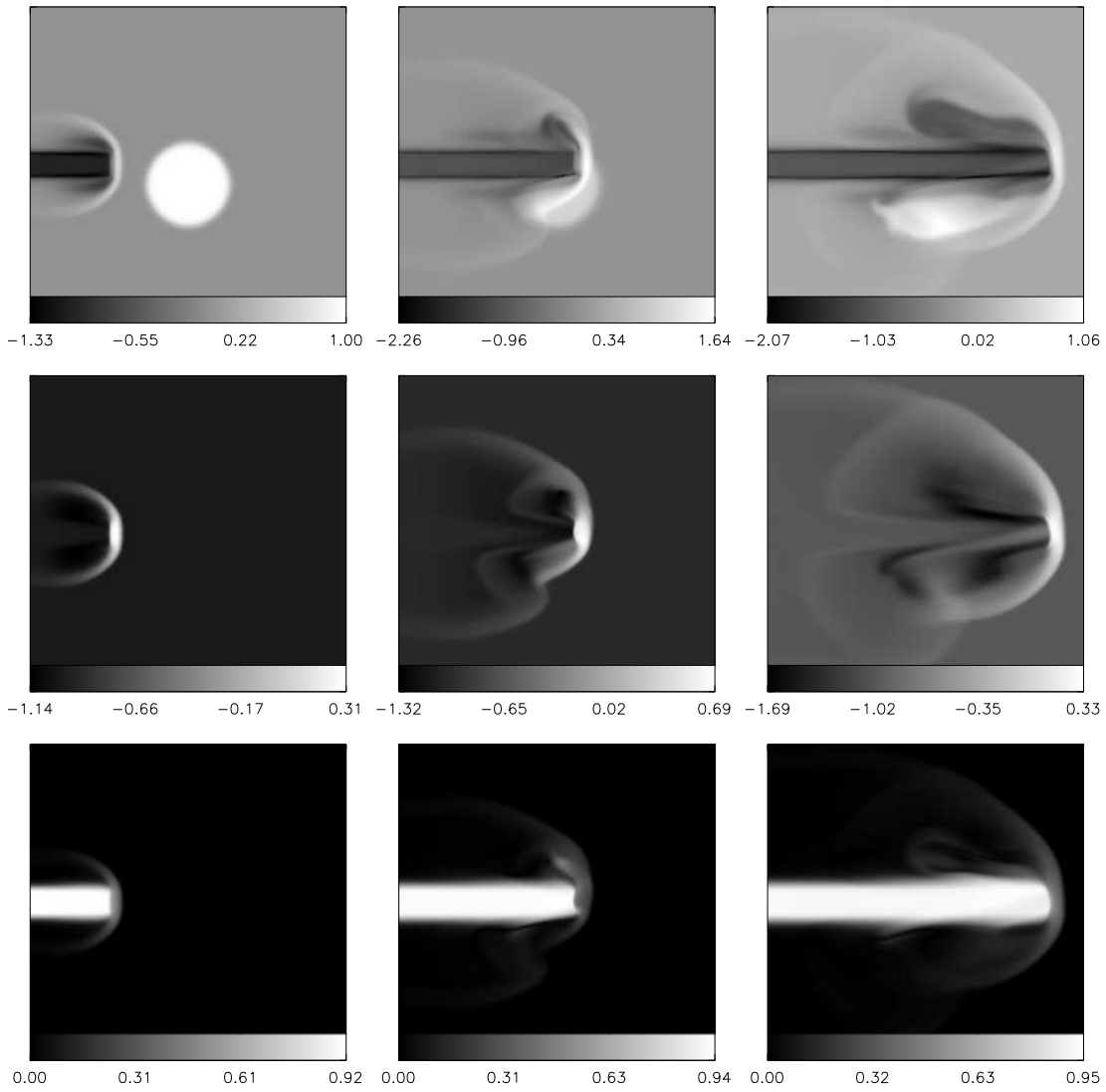

Figure 6.1: (c) Same as Fig. 6.1(a) except for model M3 at $t/t_{\rm bc} = 1$, 2.5, and 4.

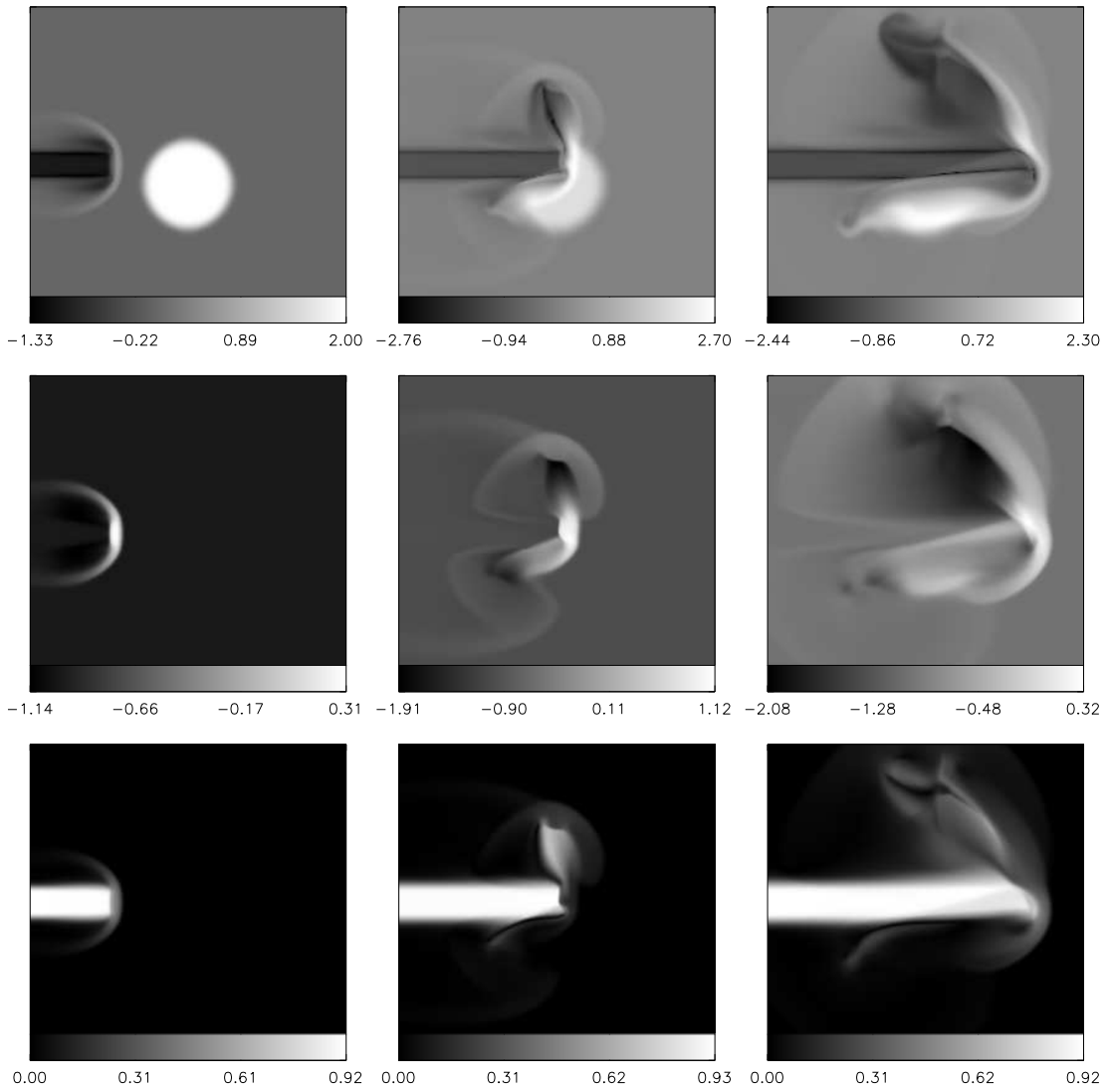

Figure 6.1: (d) Same as Fig. 6.1(a) except for model M4 at $t/t_{\rm bc} = 1$, 3, and 5.

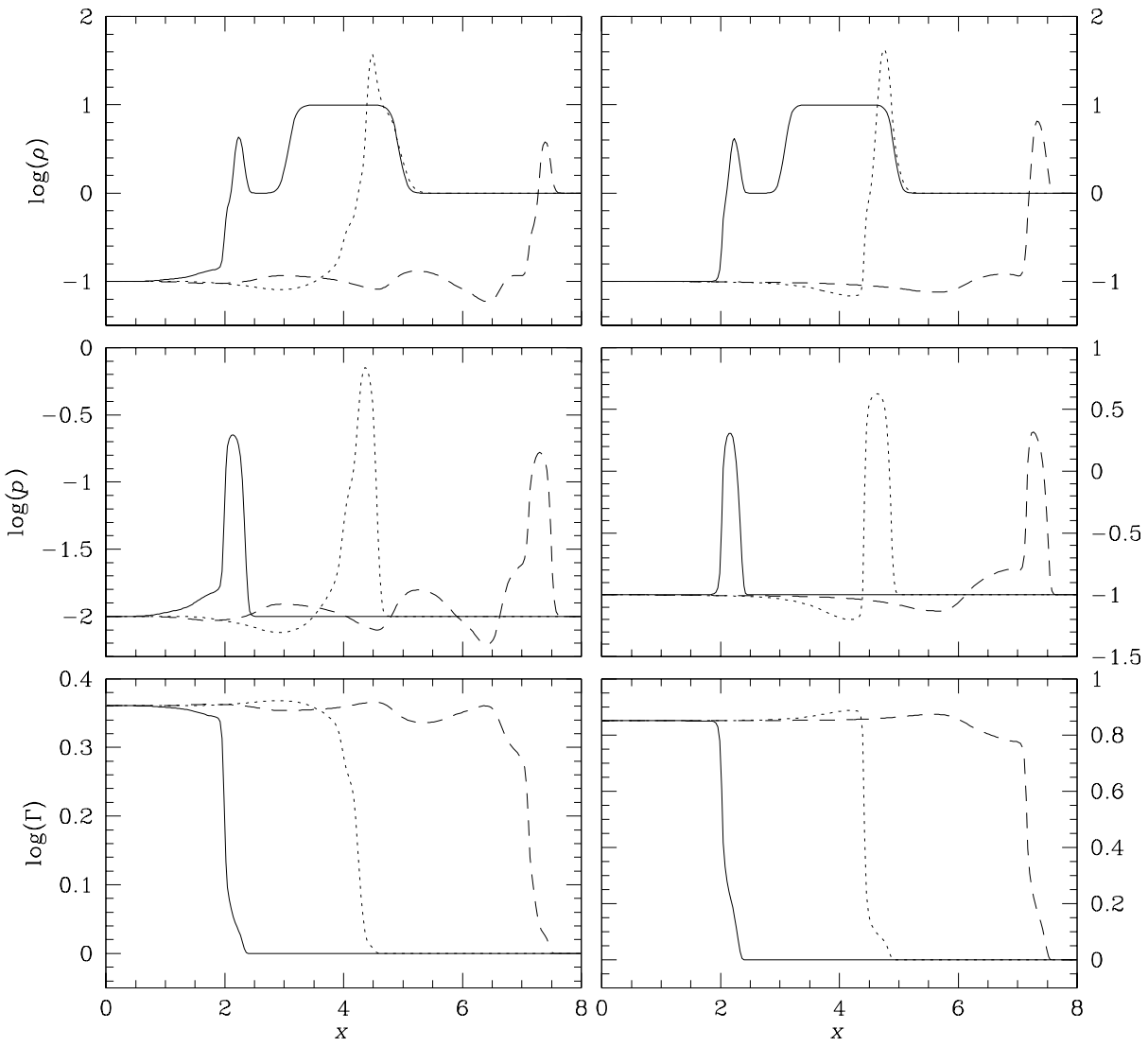

Figure 6.2: Distributions of density, pressure, and Lorentz factor along the beam propagation axis with $y = z = 4$ for models M1 (left) and M3 (right) at the same epochs as in Figs. 6.1(a) and (c). The solid lines correspond to $t/t_{\rm bc} = 1$, dotted lines represent profiles at $t/t_{\rm bc} = 2.5$, and dashed lines illustrate quantities at $t/t_{\rm bc} = 4$.

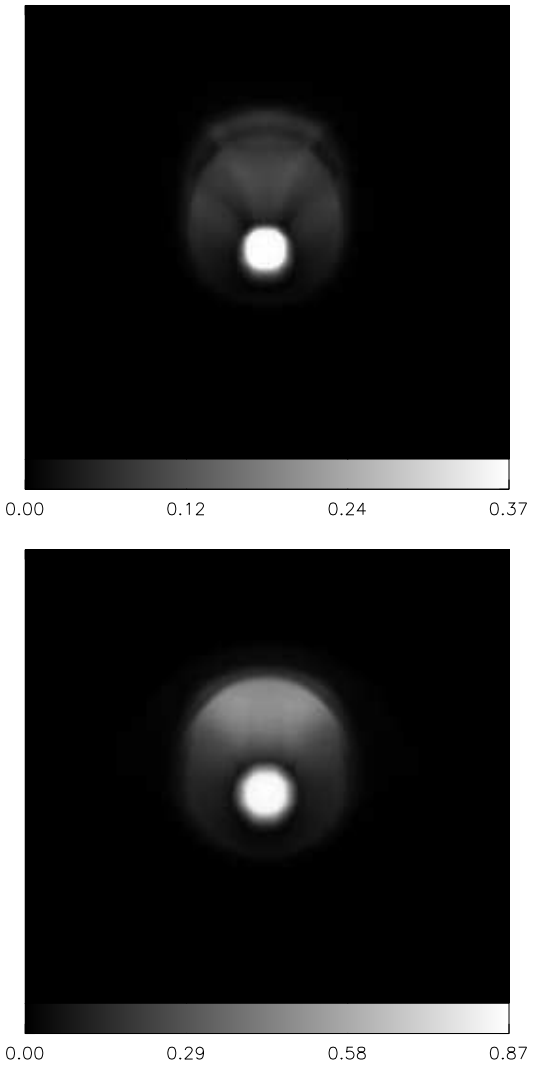

Figure 6.3: Projection images of Lorentz factors at viewing angles of 0° for models M2 (top) and M4 (bottom) at $t/t_{\rm bc} = 3.5$ (M2) and 3 (M4). The image scales are logarithmic and the images are projected on the $y - z$ plane in the three-dimensional computational domain.

6.2 Cloud Evolutions

Although previous studies of jet interactions with clouds mainly emphasized the dynamical and morphological features of the jet itself, it is also important to follow the evolution of the clouds during and after the off-axis collisions with relativistic jets. One key reason for investigating the fate of the clouds is that the leftover cloud material is a strong candidate for star formation regions in the vicinity of AGNs (e.g., Rees, 1989; Gopal-Krishna & Wiita, 2001). The cloud is expected to undergo a somewhat different evolution in our case compared with the evolution of the clouds struck by the nonrelativistic planar shocks considered in earlier work (e.g., Klein et al., 1994; Xu & Stone, 1995; Mellema et al., 2002; Fragile et al., 2004).

In order to describe the evolution of a cloud quantitatively, we introduce a conserved variable f called a Lagrangian tracer (e.g., Jones et al., 1996) which is updated along with the primitive hydrodynamic variables in our simulations. The evolution of the Lagrangian tracer is followed by the conservation equation

$$\frac{\partial Df}{\partial t} + \frac{\partial}{\partial x_j}\left(Dfv_j\right) = 0. \tag{6.1}$$

Since the above conservation equation is almost identical to the mass conservation equation (5.4), it is separately solved using the same TVD routine adopted for solving the mass conservation equation. Initially the tracer variable is set to unity ($f_c = 1$) inside the cloud while the variable is set to zero ($f_c = 0$) everywhere outside the cloud, so that the density of cloud material is given as $D_c = Df_c$ (i.e., $\rho_c = \rho f_c$) for a given tracer f_c in any zone. Then

the total mass of the cloud is computed by the integration over the entire volume V,

$$m_c = \int_V \rho_c dV, \tag{6.2}$$

where $dV = dxdydz$. This enables us to compute the several useful mass-weighted quantities such as the mean square radius of the cloud and the mean velocity of the cloud, computed with respect to the original center of the cloud

$$\langle r_i^2 \rangle = \frac{1}{m_c} \int_V \rho_c r_i^2 dV, \tag{6.3}$$

$$\langle v_i \rangle = \frac{1}{m_c} \int_V \rho_c v_i dV. \tag{6.4}$$

The index i given above represents each spatial component. Another useful mass-weighted quantity is the mean thermal energy inside the cloud $\langle e_{\rm th} \rangle$. This is also computed using the same volume integration given above.

We show in Figure 6.4 the volume-rendering images of cloud density for model M4 at three different times, $t/t_{\rm bc} = 1$, 3, and 5. As a direct consequence of the impact on the cloud by the jet the cloud develops a cavity in the cloud body as shown in the figure. The cloud cavity continues to grow until the jet completely penetrates the cloud, elongating the cloud material outside the cavity along the bow shock of the jet. Unlike the cases studied earlier where a cloud interacts with a plane-parallel shock (Klein et al., 1994; Xu & Stone, 1995), the cloud material is not completely destroyed by the impact of the jet. Some cloud mass is carried into the deflected material of the jet, eroding the cloud body, but much of the cloud mass remains in a large, coherent blob for at least a few beam crossing times. This

enhancement of the cloud durability is apparently primarily due to the geometric influence of an off-axis collision. Computational resource limits prescribe that we can accurately investigate the clouds for only a few beam crossing times, which is less than the many dynamical times for which it would be desirable to follow their evolutions.

Figure 6.5 shows for every model the time evolutions of the root mean square radius of cloud, the mean cloud velocity, and the mean thermal energy of the cloud. In every model the clouds remain in the initial root mean square radius, $\langle r_i^2 \rangle^{1/2} = 0.44$ until $t/t_{\rm bc} \sim 1.5$. When the jet hits the cloud, the cloud is first crushed in the x-direction, along which the jet propagates, and then it begins to expand beyond its initial size. The initial compressions in the y- and z-directions are very small and the cloud soon gradually expands in both these transverse directions. By the end of these simulations the root mean square radii of the clouds have expanded to about $1.5 - 2$ times their initial values.

After $t/t_{\rm bc} \sim 1.5$ the high pressure inside the cloud generated by the incident jet causes the entire cloud to accelerate. Unsurprisingly, the acceleration is faster in the x-direction for the faster jets in models M3 and M4 and for the lighter clouds in models M1 and M3. The mean velocity of the clouds peaks at values between 0.01 and 0.15 in the x-direction and 0.005 and 0.05 in the y-direction. Note that the mean velocity of the clouds in the z-direction is zero because of symmetry in this direction. The maximum velocity of the cloud is always rather modest even if the incident jet has a relativistic speed, though if we had considered less massive clouds they obviously could have been accelerated to higher speeds.

As we expect, the mean thermal energy of the cloud increases while the jet strikes the cloud. The maximum mean thermal energy of the cloud reaches about $5 - 15$ times its initial value, depending upon the model. In each model the peaks of the mean thermal energy

inside the cloud and the acceleration of the mean velocity of the cloud take place nearly at the same time. Note that at this point the cloud reexpands after the cloud reaches the maximum compression in the x-direction.

As mentioned earlier, the jet interaction with the cloud shows that the beam penetrates through the cloud body, which may begin a fragmentation process. A strong shear layer developing at the cloud boundary as a result of the interaction with the jet may lead to Kelvin-Helmholtz instabilities which enable the disrupted cloud body to fragment. So eventually the gas cloud might be broken into small pieces. However, the Kelvin-Helmholtz instability becomes inefficient if the density contrast of two slipping fluids is large or if the flow is supersonic (Chandrasekhar, 1961), so we may not see rapid fragmentation in the clouds. Although the fragmentation timescale is difficult to estimate, our simulations show no significant cloud fragmentation by $t/t_{\rm bc} \sim 4 - 6$. This indicates that the high density contrast between clouds and beams and the supersonic velocity of the clouds induced by the relativistic jets do indeed lower the growth rate of the Kelvin-Helmholtz instabilities.

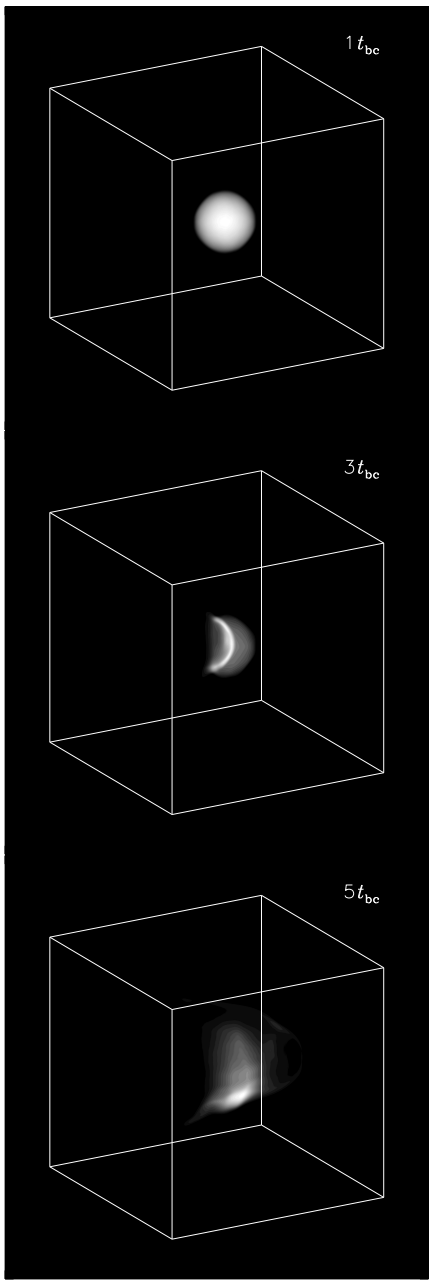

Figure 6.4: Volume-rendering images of cloud density for model M4 at three different times, $t/t_{\rm bc} = 1$, 3, and 5. The image scales are linear and the viewing area is rotated 20° about the x-axis and 30° about the z-axis. Black represents the lowest values which are ~ 0 at each epoch and white the highest values which are ~ 100 at $t/t_{\rm bc} = 1$, ~ 464 at $t/t_{\rm bc} = 3$, and ~ 229 at $t/t_{\rm bc} = 5$.

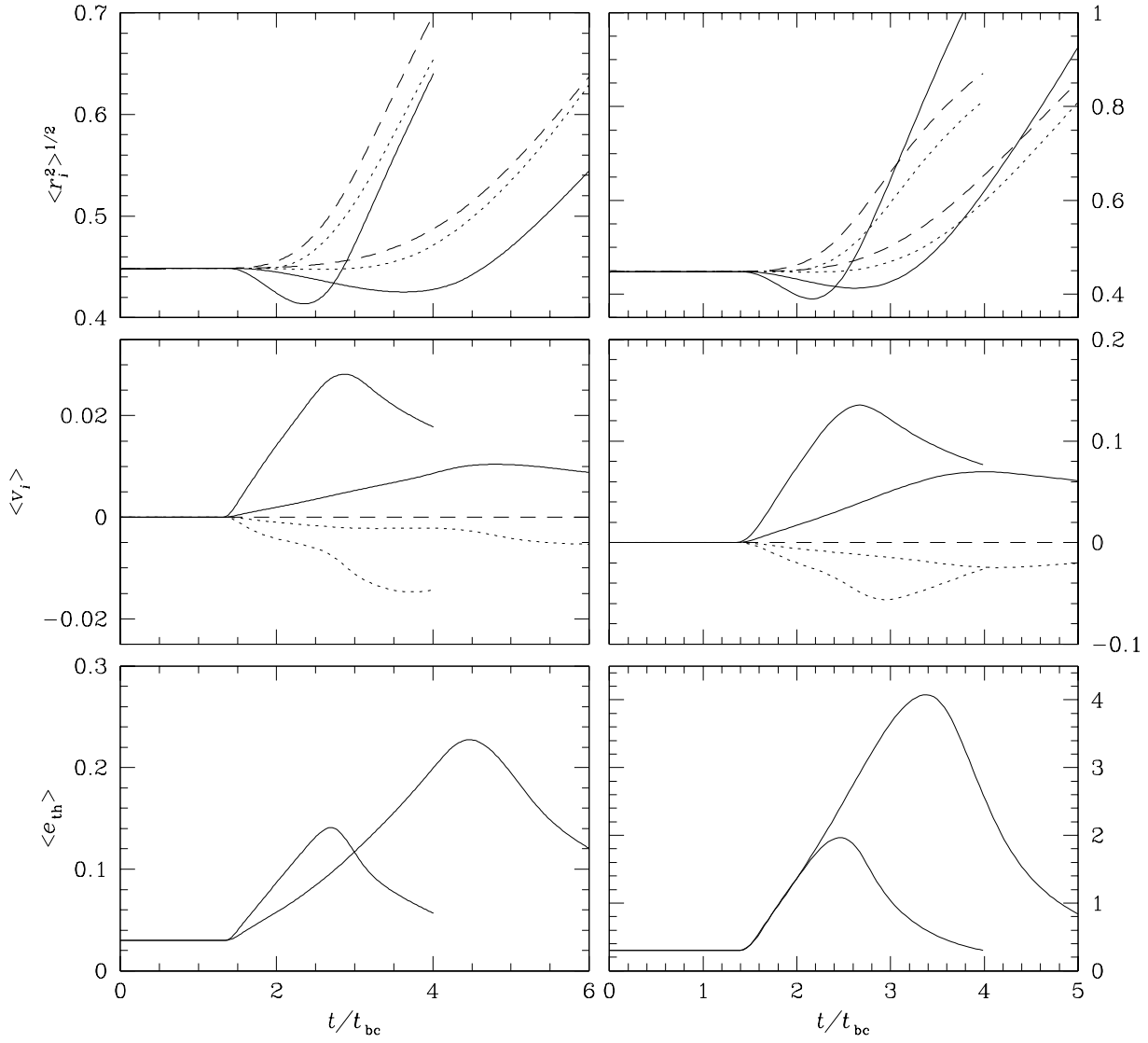

Figure 6.5: Time evolutions of the root mean square radius of the cloud, the mean cloud velocity, and the mean thermal energy of the cloud for models M1 (curves ending at $t/t_{\rm bc} = 4$) and M2 (curves ending at $t/t_{\rm bc} = 6$) in the left panels and for models M3 (curves ending at $t/t_{\rm bc} = 4$) and M4 (curves ending at $t/t_{\rm bc} = 5$) in the right panels. The solid, dotted, and dashed lines in the root mean square radius and the mean velocity panels represent the x-, y-, and z-components, respectively.

6.3 Synchrotron Emission

Propagating relativistic jets produce nonthermal radio (synchrotron) emission which originates from relativistic high-energy particles accelerated at the shock front. Jones et al. (1999) and Tregillis et al. (2001) calculated the synchrotron emission in extragalactic jets by explicitly calculating the acceleration of electrons at shocks and following the evolution of magnetic field. However, they assumed nonrelativistic jets, and hence the emissivity needs to be further examined using relativistic jets. To compute the synchrotron emission from relativistic jets, other relativistic hydrodynamic simulations have worked with a simpler approximation (Gómez et al., 1997; Komissarov & Falle, 1997; Mioduszewski et al., 1997; Aloy et al., 2000). Using this same simple model we now calculate the synchrotron emission in our simulations in order to estimate how the relativistic jet interaction with a cloud would appear in emission as a extragalactic radio source. We make the usual assumptions that the jet is optically thin and only the jet material radiates. Thus, in order to separate the jet material from the ambient medium and the cloud, we include an additional tracer variable f_b (see Section 6.2) which is initially set to unity inside the jet ($f_b = 1$) and zero everywhere outside the jet ($f_b = 0$).

The relativistic high-energy electrons responsible for the synchrotron emission are assumed to have a power-law energy distribution. Given the spectral index α, the high-energy particle number density N_0, and the magnetic field intensity B, the synchrotron emissivity at frequency ν is then approximated by the power-law distribution (see e.g., Mioduszewski et al., 1997)

$$j_\nu \propto N_0 B^{\alpha+1} \nu^{-\alpha}. \tag{6.5}$$

The high-energy particle number density, N_0, is assumed to be proportional to the relativistic electron energy density, u_e, from the integration of the power-law energy distribution over some energy range, and u_e is also taken to be proportional to the hydrodynamic pressure. Then we have $N_0 \propto u_e \propto p$. Assuming that there is an equipartition of the magnetic field energy density u_B and the relativistic electron energy density ($u_B = u_e$), then $u_B \propto p$. This leads to $B \propto u_B^{1/2} \propto p^{1/2}$. Therefore, equation (6.5) becomes

$$j_\nu \propto p^{(\alpha+3)/2} \nu^{-\alpha}. \tag{6.6}$$

This equation shows that the local thermal pressure approximately reflects the local synchrotron emissivity. We have used $\alpha = 0.75$ in our calculation. By integrating the synchrotron emissivity along the line of sight L through the emitting plasma at a viewing angle θ, we can compute the synchrotron intensity on the surface projected onto the line of sight at that viewing angle to be

$$I_\nu = \int_L \mathcal{D}^2 j_\nu dL, \tag{6.7}$$

where the Doppler boosting factor is given by

$$\mathcal{D} = \frac{1}{\Gamma\,(1 - v\cos\theta)}. \tag{6.8}$$

Other relativistic effects, including light aberration and time dilation, have not been included in this calculation, as we can reasonably assume that these effects are negligible in comparison to the Doppler boosting.

Figure 6.6 shows the synchrotron intensity maps of models M1 and M2 at the viewing angles of 90°, 45°, and 0° with respect to the jet propagation axis. These maps are shown at $t/t_{\rm bc} = 2.5$ for M1 and 3.5 for M2 when the jet is colliding with the cloud. The peak intensity in this figure varies with the models and the angles of view. Doppler boosting has a modest negative effect on the emission of the jet at the viewing angle 90°, so that the observed emission is quite closely related to the intrinsic emissivity in this case. At smaller viewing angles (e.g., 45° and 0°), however, the emission morphology is determined to a large degree by Doppler boosting. The synchrotron emission is dominated by the bright hotspot, which can correspond to the compact emission knot in VLBI radio maps. Although the beam and the deflected material show only weak emission features, there is a faint secondary spot seen from deflected material in model M2.

The time evolution of the total synchrotron intensity for models M1–M4 at the viewing angles of 90°, 45°, and 0° are shown in Figure 6.7. The total synchrotron intensities computed here are in arbitrary units. There are significant quantitative differences among the models, but the intensity curves qualitatively show the same trends. The total synchrotron intensity is much amplified at smaller viewing angles of 45° and 0° because Doppler boosting plays a role in the amplification of the intensity in these cases. As expected, the passage of the jet over a cloud enhances the synchrotron intensity; there are high amplitude bumps in the intensity curves during the interactions. The total intensity steeply increases at the moment of the impact by the jet, and then gradually increases until the jet crosses over the cloud. This tells us that the compression of the plasma in this region produces higher synchrotron emission in this approximation where it is tied to the pressure. The peak synchrotron

intensity occurs shortly after the jet passes through the entire cloud, and after that the intensity falls off slowly because the compression is weaker.

Although we have not computed the thermal X-ray emission in detail, we can briefly discuss it. Since the free-free emission (bremsstrahlung) is proportional to ρ^2, the total X-ray luminosity due to thermal bremsstrahlung is most sensitive to the density of gas, provided that the gas is, or becomes, hot enough to emit X-rays. The relativistic jet itself is not expected to emit thermal X-rays because of its low density, and only in some cases does the synchrotron spectrum extend far enough to produce nonthermal X-ray emission (Harris & Krawczynski, 2006). The dense cloud is very unlikely to start out hot enough to emit X-rays. However, during the jet-cloud interaction the density and pressure of the cloud become so high that the total X-ray emission may be larger inside the cloud than elsewhere, for example, in the bow shock of the jet. An estimate of the increase in the total X-ray luminosity of the cloud is given by $L_x/L_{x,0} \simeq (\rho_c/\rho_{c,0})^2(T_c/T_{c,0})^{1/2}(V_c/V_{c,0})$, where L_x is the total X-ray luminosity of the cloud, T_c is the mean cloud temperature, V_c is the mean cloud volume, and the subscript 0 represents the initial (preshocked) value. If we simply assume an ideal gas so $T_c \propto p_c/\rho_c$, we have $L_x/L_{x,0} \sim (\rho_c/\rho_{c,0})^{3/2}(p_c/p_{c,0})^{1/2}(V_c/V_{c,0})$, allowing us to estimate the total X-ray luminosity of the cloud with respect to its preshocked X-ray luminosity. In model M4, for example, $\rho_c/\rho_{c,0} \approx 3$, $p_c/p_{c,0} \approx 12$, and $V_c/V_{c,0} \approx 1$ at $t/t_{\rm bc} = 3$ (as can be roughly estimated from Figs. 6.4 and 6.5), so that $L_x/L_{x,0} \sim 18$. Thus, the shocked cloud could possibly be a important source of thermal X-rays depending upon the various physical parameters such as the incident jet velocity, the cloud density, and, most importantly, the initial cloud temperature, which is not explicitly specified in our scaled models. However

any significant thermal X-ray luminosity should subside rapidly after the interaction as the cloud is diffusive and quickly attains equilibrium with the postshocked ambient pressure.

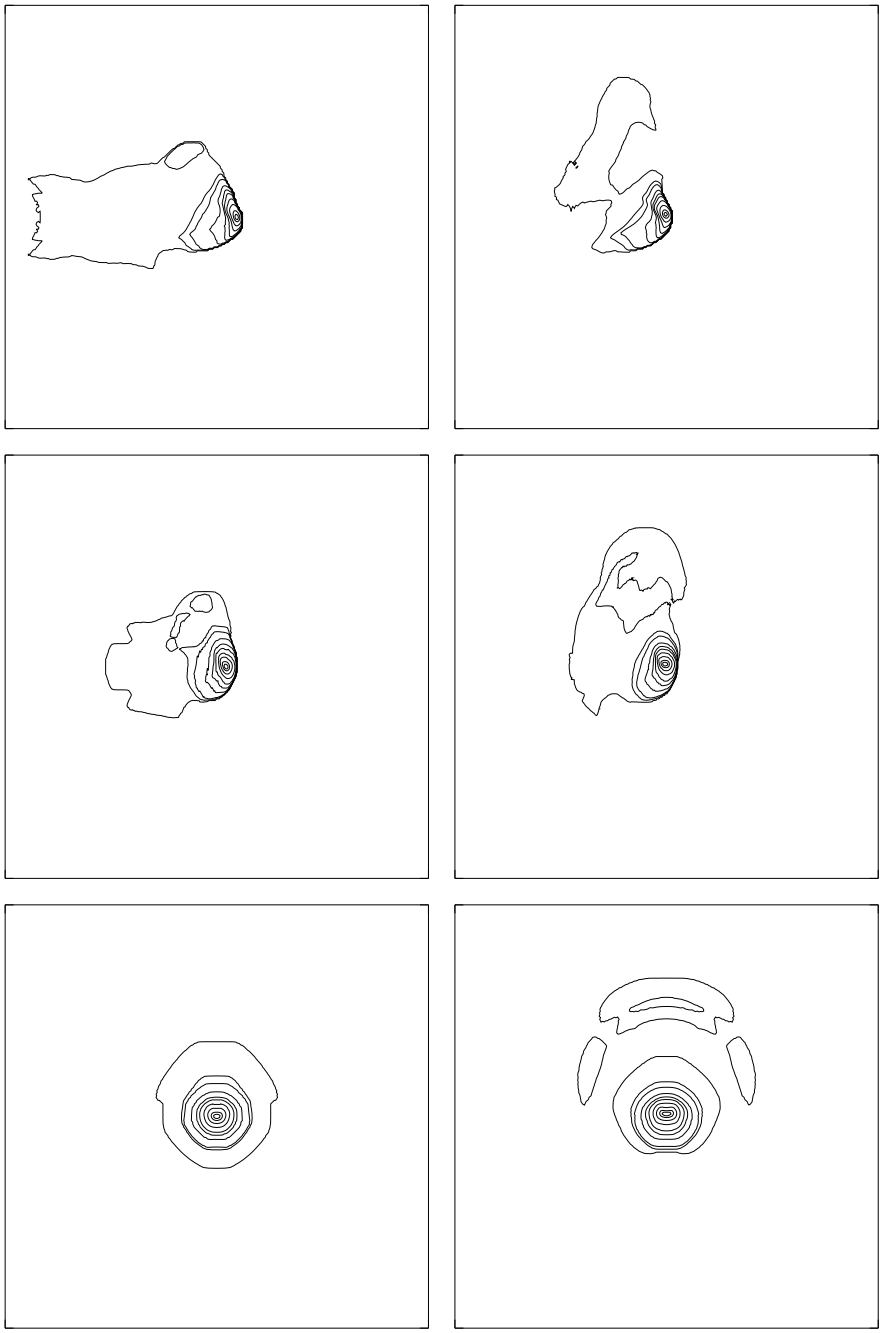

Figure 6.6: Contour maps of synchrotron intensity at the viewing angles of 90°, 45°, and 0° (top to bottom) for models M1 (left) and M2 (right) at $t/t_{\rm bc} = 2.5$ (M1) and 3.5 (M2). Maximum synchrotron intensities are 0.17 (90°), 0.13 (45°), and 0.27 (0°) for model M1 and 0.57 (90°), 0.35 (45°), and 0.55 (0°) for model M2, and the contour levels are 0.1%, 0.5%, 1%, 3%, 6%, 10%, 20%, 40%, 70%, and 90% of the maximum synchrotron intensity.

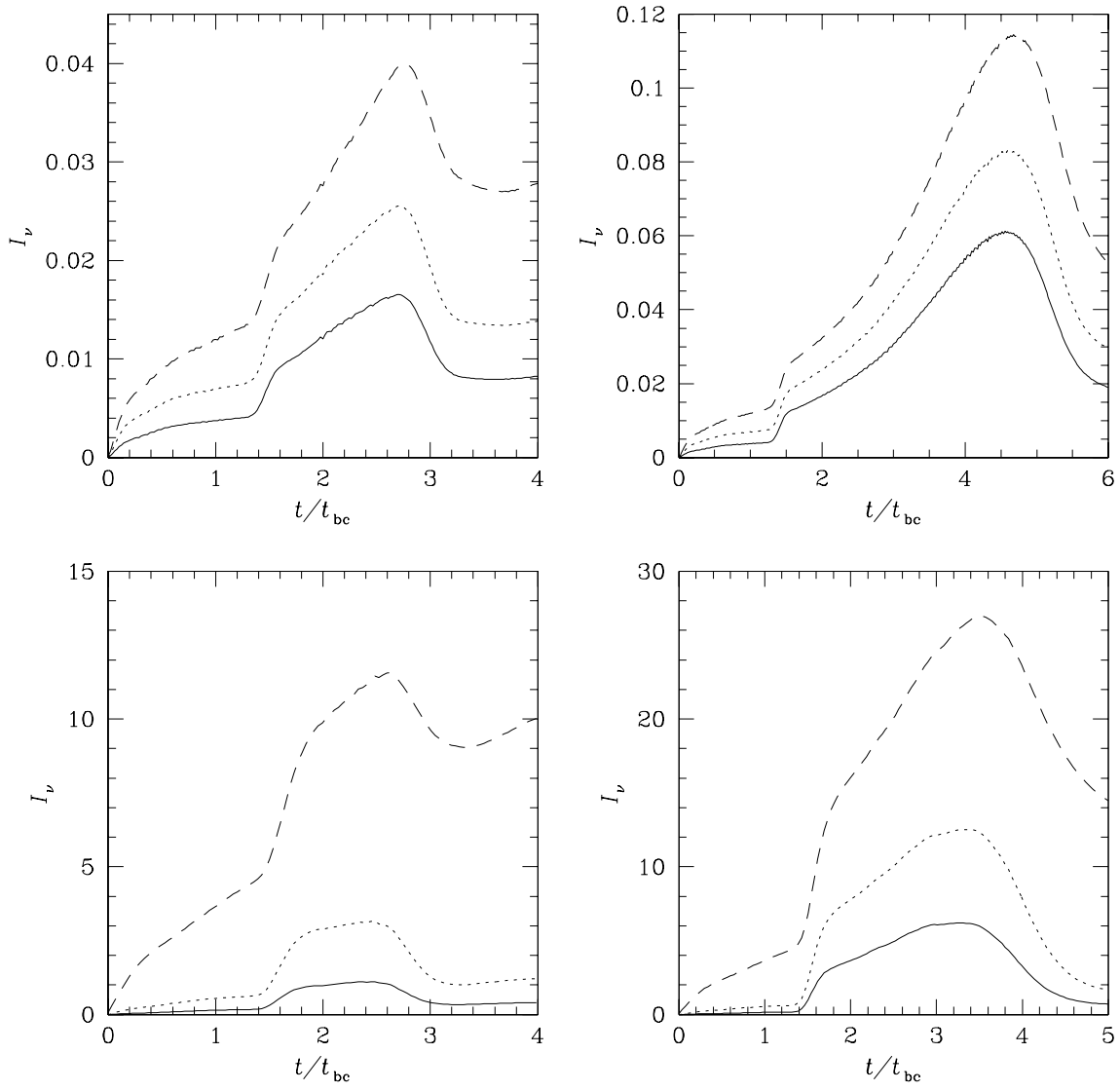

Figure 6.7: Total synchrotron intensity curves for models M1 (top left), M2 (top right), M3 (bottom left), and M4 (bottom right). The solid, dotted, and dashed lines correspond to the viewing angles of 90°, 45°, and 0°, respectively.

Chapter 7

Summary and Discussion

A multidimensional code for special relativistic hydrodynamics was described. It differs from previous codes in the following aspects: 1) It is based on the total variation diminishing (TVD) scheme (Harten, 1983), which is an explicit Eulerian finite difference upwind scheme and an extension of the Roe scheme to second-order accuracy in space and time. 2) It employs a new set of conserved quantities, and so the dissertation describes a new eigenstructure for special relativistic hydrodynamics. 3) For the Lorentz transformation from the conserved quantities in the reference frame to the physical quantities in the local rest frame, an analytic formula is used.

To demonstrate the performance of the code, several tests were presented, including relativistic shock tubes, a relativistic wall shock, a relativistic blast wave, the relativistic version of the Hawley-Zabusky shock, and a relativistic extragalactic jet. The relativistic shock tube tests showed that the code clearly resolves mildly relativistic and highly relativistic shocks within $2-4$ numerical cells, although it requires more cells for resolving contact discontinuities. The relativistic wall shock test showed that the code correctly captures very strong shocks with very high Lorentz factors. The relativistic blast wave test showed that blast waves propagate through the ambient medium while preserving the symmetry. The test simulations of the relativistic version of the Hawley-Zabusky shock and a relativistic extragalactic jet proved the robustness and flexibility of the code, and that the code can be applied to studies of practical astrophysical problems.

The strong points of the new code include the following: 1) Based on the TVD scheme, the code is simple and fast. The core routine of the TVD relativistic hydrodynamics is only about 300 lines long in the three-dimensional version. It runs only about $1.5-2$ times slower than the non-relativistic counterpart (per time step). Yet, tests have shown that the code is accurate and reliable enough to be suited for astrophysical applications. In addition, the use of an analytic formula for Lorentz transformation makes the code robust, so it ran for all the tests we have performed without failing to converge. 2) The code has been built in a way to be completely parallel to the non-relativistic counterpart. So it can be easily understood and used, once one is familiar with the non-relativistic code. In addition, the techniques developed for the non-relativistic code such as parallelization can be imported transparently.

We have used this code to perform three-dimensional relativistic hydrodynamic simulations to study relativistic jet interactions with dense clouds, focusing on the influence of special relativistic effects. We have investigated clouds struck by both low and high relativistic beam Mach number jets which have less and more dominant relativistic effects, respectively, and have compared our results to the extent possible with nonrelativistic simulations which have been published previously. We also have studied the evolution of the assaulted clouds and have estimated the synchrotron emission from the relativistic jets interacting with the clouds.

In our models, the partial deflections of the jets due to the interactions with clouds are seen more clearly when denser clouds are involved, and the deflection angle is more strongly influenced by the density contrast of the cloud to the ambient medium than by the beam Mach number of the jet. The streams of deflected gas from the jet induced by the interactions move outward much faster compared to nonrelativistic models. If our models

can be generalized, this suggests that the relativistic jet-cloud interactions are an effective mechanism of producing at least some of the outflows observed in the vicinity of AGNs (e.g., Emonts et al., 2005; Morganti et al., 2005). After the relativistic jets interact with the dense clouds, we find that the slower relativistic jets can be bent by modest angles and that these bent jets still remain stable and collimated over fairly extended timescales. This trend is similar to the results from nonrelativistic simulations.

The impact of the jet erodes the cloud, but much of the cloud mass survives as a large coherent body rather than being completely destroyed. This enhancement of the cloud durability compared to interactions with planar shocks appears to be primarily due to the geometric influence of an off-axis collision. Compared to head-on collisions, off-axis collisions damage the cloud less, increasing the chance of survival of a large portion of the cloud. Another likely reason for the enhancement of the cloud durability is that the rate of cloud fragmentation through Kelvin-Helmholtz instabilities is lowered since the relativistic flows reduce the growth rate of the instabilities compared to similar off-axis blows by nonrelativistic jets. This leftover tenacious cloud material could be a candidate for a strong star formation region in the vicinity of AGNs, particularly when the cooling timescales are sufficiently short.

The synchrotron intensity "maps" show that at the jet impact on a cloud, the synchrotron emission comes dominantly from a bright hotspot which could correspond to the form of the compact emission knots seen in many VLBI radio maps. Although the emission from the deflected jet material is relatively weak, sometimes there is a secondary synchrotron spot visible from this deflected material. This emission feature may represent some of the distorted emission seen in many VLBI radio maps. The passage of the jet over a cloud significantly enhances the total synchrotron intensity of the jet. We find that the synchrotron

emission is steeply enhanced shortly after the jet hits the cloud, but the emission peaks right before the jet passes through the cloud. The next big step in performing these calculations would be to include magnetic fields. Such relativistic magnetohydrodynamical simulations would allow for better estimates of synchrotron emission and would allow the examination of the polarization of emission arising from more complicated shock structures. Such polarization structures could be a useful diagnostic of the dynamics.

Although most astrophysical simulations based on relativistic hydrodynamics, including this study, have assumed the ideal EOS, it is well known that the ideal EOS is correct only if the gas is assumed to be entirely nonrelativistic ($\gamma = 5/3$) or ultrarelativistic ($\gamma = 4/3$). If a local transition between nonrelativistic gas and relativistic gas is involved, the ideal EOS will produce incorrect results in that regime. Recently, Ryu et al. (2006) have studied this issue of the EOS in numerical relativistic hydrodynamics and propose a new EOS which is simple and yet approximates closely the EOS of a perfect gas in the relativistic regime, having an accuracy in enthalpy better than 0.8%. Future numerical simulations using this new EOS should produce even better results concerning the problem of relativistic jet interactions with clouds.

References

Abramowitz, M. A., & Stegun, I. A. 1972, Handbook of Mathematical Functions (Dover: Dover Publishing Company)

Aloy, M. A., Gómez, J. L., Ibáñez, J. M., Martí, J. M., & Müller, E. 2000, ApJ, 528, L85

Aloy, M. A., Ibáñez, J. M., Martí, J. M., Gómez, J. L., & Müller, E. 1999b, ApJ, 523, L125

Aloy, M. A., Ibáñez, J. M., Martí, J. M., & Müller, E. 1999a, ApJS, 122, 151

Bicknell, G. V. 1995, ApJS, 101, 29

Blandford, R. D., & McKee, C. F. 1976, Phys. Fluids, 19, 1130

Burrows, A. 2000, Nature, 403, 727

Chandrasekhar, S. 1961, Hydrodynamic and Hydromagnetic Stability (Oxford: Clarendon Press)

Choi, E., & Ryu, D. 2005, NewA, 11, 116

Choi, E., Wiita, P. J., & Ryu, D. 2007, ApJ, 655, 769

Colella, P., & Woodward, P. R. 1984, J. Comput. Phys., 54, 174

de Gouveia Dal Pino, E. M. 1999, ApJ, 526, 862

Dolezal, A., & Wong, S. S. M. 1995, J. Comput. Phys., 120, 266

Donat, R., Font, J. A., Ibáñez, J. M., & Marquina, A. 1998, J. Comput. Phys., 146, 58

Duncan, G. C., & Hughes, P. A. 1994, ApJ, 436, L119

Emonts, B. H. C., Morganti, R., Tadhunter, C. N., Oosterloo, T. A., Holt, J., & van der Hulst, J. M. 2005, MNRAS, 362, 931

Eulderink, F., & Mellema, G. 1995, A&AS, 110, 587

Fanaroff, B. L., & Riley, J. M. 1974, MNRAS, 167, 31P

Fragile, P. C., Murray, S. D., Anninos, P., & van Breugel, W. 2004, ApJ, 604, 74

Gómez, J. L., Martí, J. M., Marscher, A. P., Ibáñez, J. M., & Alberdi, A. 1997, ApJ, 482, L33

Gopal-Krishna, & Wiita, P. J. 2001, ApJ, 560, L115

Harris, D. E., & Krawczynski, H. 2006, ARA&A, 44, 463

Harten, A. 1983, J. Comput. Phys., 49, 357

Harten, A., Engquist, B., Osher, S., & Chakravarthy, S. R. 1987, J. Comput. Phys., 71, 231

Hawley, J. F., & Zabusky, N. J. 1989, Phys. Rev. Lett., 63, 1241

Higgins, S. W., O'Brien, T. J., & Dunlop, J. S. 1999, MNRAS, 309, 273

Hughes, P. A., Aller, H. D., & Aller, M. F. 1985, ApJ, 298, 301

Hughes, P. A., Miller, M. A., & Duncan, G. C. 2002, ApJ, 572, 713

Jones, T. W., Ryu, D., & Engel, A. 1999, ApJ, 512, 105

Jones, T. W., Ryu, D., & Tregillis, I. L. 1996, ApJ, 473, 365

Jorstad, S. G., Marscher, A. P., Mattox, J. R., Aller, M. F., Aller, H. D., Wehrle, A. E., & Bloom, S. D. 2001, ApJ, 556, 738

Klein, R. I., McKee, C. F., & Colella, P. 1994, ApJ, 420, 213

Komissarov, S. S., & Falle, S. A. E. G. 1997, MNRAS, 288, 833

Komissarov, S. S., & Falle, S. A. E. G. 1998, MNRAS, 297, 1087

Königl, A. 1980, Phys. Fluids, 23, 1083

Landau, L. D., & Lifshitz, E. M. 1959, Fluid Mechanics (London: Pergamon Press)

Martí, J. M., & Müller, E. 1994, J. Fluid Mech., 258, 317

Martí, J. M., & Müller, E. 1996, J. Comput. Phys., 123, 1

Martí, J. M., & Müller, E. 2003, Living Rev. Relativity, 6, 7

Martí, J. M., Müller, E., Font, J. A., Ibáñez, J. M., & Marquina, A. 1997, ApJ, 479, 151

Martí, J. M., Müller, E., & Ibáñez, J. M. 1994, A&A, 281, L9

Mellema, G., Kurk, J. D., & Röttgering, H. J. A. 2002, A&A, 395, L13

Mészáros, P. 2002, ARA&A, 40, 137

Mioduszewski, A. J., Hughes, P. A., & Duncan, G. C. 1997, ApJ, 476, 649

Mirabel, I. F., & Rodríguez, L. F. 1999, ARA&A, 37, 409

Mizuta, A., Yamada, S., & Takabe, H. 2004, ApJ, 606, 804

Morganti, R., Oosterloo, T. A., Tadhunter, C. N., van Moorsel, G., & Emonts, B. 2005, A&A, 439, 521

Mundell, C. G., Wrobel, J. M., Pedlar, A., & Gallimore, J. F. 2003, ApJ, 583, 192

Poludnenko, A. Y., Frank, A., & Blackman, E. G. 2002, ApJ, 576, 832

Rees, M. J. 1989, MNRAS, 239, 1P

Roe, P. L. 1981, J. Comput. Phys., 43, 357

Rosen, A., Hughes, P. A., Duncan, G. C., & Hardee, P. E. 1999, ApJ, 516, 729

Ryu, D., Chattopadhyay, I., & Choi, E. 2006, ApJS, 166, 410

Ryu, D., Ostriker, J. P., Kang, H., & Cen, R. 1993, ApJ, 414, 1

Saxton, C. J., Bicknell, G. V., Sutherland, R. S., & Midgley, S. 2005, MNRAS, 359, 781

Schneider, V., Katscher, U., Rischke, D. H., Waldhauser, B., Maruhn, J. A., & Munz, C.-D. 1993, J. Comput. Phys., 105, 92

Strang, G. 1968, SIAM J. Numer. Anal., 5, 506

Thompson, K. W. 1986, J. Fluid Mech., 171, 365

Tregillis, I. L., Jones, T. W., & Ryu, D. 2001, ApJ, 557, 475

Urry, C. M., & Padovani, P. 1995, PASP, 107, 803

van Putten, M. H. P. M. 1993, ApJ, 408, L21

Wang, Z., Wiita, P. J., & Hooda, J. S. 2000, ApJ, 534, 201

Wiita, P. J., & Norman, M. L. 1992, ApJ, 385, 478

Wiita, P. J., Rosen, A., & Norman, M. L. 1990, ApJ, 350, 545

Wilson, J. R., & Mathews, G. J. 2003, Relativistic Numerical Hydrodynamics (Cambridge: Cambridge Univ. Press)

Xu, J., & Stone, J. M. 1995, ApJ, 454, 172

Zensus, J. A. 1997, ARA&A, 35, 607

Appendix

This appendix includes the Fortran source code which solves the three-dimensional special relativistic hydrodynamic equations in Cartesian coordinates using our characteristic decomposition and the total variation diminishing (TVD) scheme. This code is the parallelized version using the message passing interface (MPI) and contains the following components:

`makefile`	Makefile which compiles the program
`common`	A file giving dimensions of arrays and common blocks
`rjets.f`	Main program for three-dimensional relativistic jets and clouds
`init.f`	Subroutine which sets up initial conditions
`bound.f`	Sets three-dimensional boundary conditions
`tvd3.f`	Subroutine for three-dimensional extension of the TVD scheme
`tstep.f`	Computes every time step
`rhdtvd.f`	Relativistic hydrodynamic TVD scheme based on Choi & Ryu (2005)
`lorenz.f`	Solves Lorentz transformations
`prot.f`	Protects against too low minimum values of density and pressure
`dump.f`	Subroutine which dumps output data

```
#-----------------------------------------------------------------------
#  makefile for rhd3b             written by Eunwoo Choi (December 2005)
#-----------------------------------------------------------------------
fc   = mpif77
opts =
dir  =
objs = rjets.o init.o tvd3.o tstep.o rhdtvd.o lorenz.o prot.o\
       bound.o dump.o
#-----------------------------------------------------------------------
#  rule
#-----------------------------------------------------------------------
rjets.x : $(objs)
  $(fc) $(opts) -o $(dir)rjets.x $(objs)
$(objs) : common
  $(fc) $(opts) -c $*.f
#-----------------------------------------------------------------------
#  end
#-----------------------------------------------------------------------
```

```
c     implicit none
c
      integer nx, ny, nz
      integer np, nxp, master
      parameter (nx=256,ny=256,nz=256)
      parameter (np=32,nxp=8,master=0)
c
      real*8 q1(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       q2(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       q3(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       q4(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       q5(-1:nxp+2,-1:ny+2,-1:nz+2)
      real*8 u1(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       u2(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       u3(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       u4(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       u5(-1:nxp+2,-1:ny+2,-1:nz+2)
      real*8 w0(5,-1:nx+2), w1(105,-1:nx+2),
     +       w2(75,-1:nx+2), w3(15,-1:nx+2),
     +       w4(5,-1:nx+2)
      real*8 f1(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       f2(-1:nxp+2,-1:ny+2,-1:nz+2),
     +       w5(2,-1:nx+2),
     +       w6(12,2,ny), w7(12,2,ny)
c
      integer nstep, nunit, nw
      real*8 xsize, ysize, zsize, dx, dy, dz, dr,
     +       t, tend, dt, gam, cour, eps1, eps2
      real*8 x, y, z, r
c
      integer pi, nps, status(MPI_STATUS_SIZE), err
c
      common /hydro1/ q1, q2, q3, q4, q5
      common /hydro2/ u1, u2, u3, u4, u5
      common /hydro3/ w0, w1, w2, w3, w4
      common /hydro4/ f1, f2, w5, w6, w7
c
      common /param1/ nstep, nunit, nw
      common /param2/ xsize, ysize, zsize, dx, dy, dz, dr,
     +                t, tend, dt, gam, cour, eps1, eps2
      common /param3/ x, y, z, r
c
      common /mpipar/ pi, nps, status, err
```

```
      program rjets
c-----------------------------------------------------------------------
c  3D relativistic jets and clouds
c  rjets -- init -- bound
c        -- tvd3 -- tstep
c                -- rhdtvd
c                -- lorenz
c                -- prot
c                -- bound
c        -- dump
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c-----------------------------------------------------------------------
c  MPI initialization
c-----------------------------------------------------------------------
      call MPI_INIT (err)
      call MPI_COMM_RANK (MPI_COMM_WORLD,pi,err)
      call MPI_COMM_SIZE (MPI_COMM_WORLD,nps,err)
c
      if (nps.ne.np) then
         write (*,*) 'processor number not equal to pre-defined number'
         stop
      endif
c-----------------------------------------------------------------------
c  files
c-----------------------------------------------------------------------
      if (pi.eq.master) then
         open (unit=10,file='tape10',status='unknown',form='formatted')
      endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  initial setup
c-----------------------------------------------------------------------
      call init
      nstep = 0
      t     = 0.d0
      if (pi.eq.master) then
         write (10,*) nstep, t
      endif
      nunit = 0
      call dump
```

```
c-----------------------------------------------------------------------
c  hydrodynamic steps
c-----------------------------------------------------------------------
 900  continue
c
      call tvd3
      nstep = nstep + 1
      t     = t     + dt
      if (pi.eq.master) then
         write (10,*) nstep, t, dt
      endif
      nunit = nunit + 1
      if (mod(nunit,10).eq.0) call dump
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      if (t.lt.tend) goto 900
c-----------------------------------------------------------------------
c  MPI finalization
c-----------------------------------------------------------------------
      call MPI_FINALIZE (err)
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      stop
      end
```

```
      subroutine init
c-----------------------------------------------------------------------
c  initial condition
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz
c-----------------------------------------------------------------------
c  parameters
c-----------------------------------------------------------------------
      xsize = 8.d0
      ysize = 8.d0
      zsize = 8.d0
      dx    = xsize/dble(nx)
      dy    = ysize/dble(ny)
      dz    = zsize/dble(nz)
      tend  = 20.d0
c
      gam  = 4.d0/3.d0
      cour = 0.3d0
      eps1 = 0.3d0
      eps2 = 0.1d0
c-----------------------------------------------------------------------
c  initial states
c-----------------------------------------------------------------------
      do 101 iz=1,nz
      do 101 iy=1,ny
      do 101 ix=1,nxp
         x = (dble(ix+nxp*pi)-0.5d0)*dx
         y = (dble(iy)-0.5d0)*dy
         z = (dble(iz)-0.5d0)*dz
         r = sqrt((x-4.d0)**2+(y-3.5d0)**2+(z-4.d0)**2)
c
         q1(ix,iy,iz) = 1.d0
         q2(ix,iy,iz) = 0.d0
         q3(ix,iy,iz) = 0.d0
         q4(ix,iy,iz) = 0.d0
         q5(ix,iy,iz) = 0.01d0/(gam-1.d0)+1.d0
c
         u1(ix,iy,iz) = 1.d0
         u2(ix,iy,iz) = 0.d0
         u3(ix,iy,iz) = 0.d0
```

```
         u4(ix,iy,iz) = 0.d0
         u5(ix,iy,iz) = 0.01d0/(gam-1.d0)+1.d0
c
         f1(ix,iy,iz) = q1(ix,iy,iz)*0.d0
         f2(ix,iy,iz) = q1(ix,iy,iz)*0.d0
c
         if (r.le.1.d0) then
c
         q1(ix,iy,iz) = 10.d0
         q2(ix,iy,iz) = 0.d0
         q3(ix,iy,iz) = 0.d0
         q4(ix,iy,iz) = 0.d0
         q5(ix,iy,iz) = 0.01d0/(gam-1.d0)+10.d0
c
         u1(ix,iy,iz) = 10.d0
         u2(ix,iy,iz) = 0.d0
         u3(ix,iy,iz) = 0.d0
         u4(ix,iy,iz) = 0.d0
         u5(ix,iy,iz) = 0.01d0/(gam-1.d0)+10.d0
c
         f1(ix,iy,iz) = q1(ix,iy,iz)*0.d0
         f2(ix,iy,iz) = q1(ix,iy,iz)*1.d0
c
         endif
 101  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call bound
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
      subroutine bound
c-----------------------------------------------------------------------
c  boundary condition
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz, nsize, ndest
c-----------------------------------------------------------------------
c  inflow x-boundary
c-----------------------------------------------------------------------
      if (pi.eq.master) then
c
      do 100 iz=1,nz
      do 100 iy=1,ny
      do 100 ix=1,nx/64
         y = (dble(iy)-0.5d0)*dy
         z = (dble(iz)-0.5d0)*dz
         r = sqrt((y-4.d0)**2+(z-4.d0)**2)
c
         if (r.le.1.d0/4.d0) then
c
         q1(ix,iy,iz) = 0.1d0/sqrt(1.d0-0.9d0*0.9d0)
         q2(ix,iy,iz) = (0.01d0/(gam-1.d0)+0.1d0+0.01d0)*0.9d0
     +                  /(1.d0-0.9d0*0.9d0)
         q3(ix,iy,iz) = 0.d0
         q4(ix,iy,iz) = 0.d0
         q5(ix,iy,iz) = (0.01d0/(gam-1.d0)+0.1d0+0.01d0)
     +                  /(1.d0-0.9d0*0.9d0)-0.01d0
c
         u1(ix,iy,iz) = 0.1d0
         u2(ix,iy,iz) = 0.9d0
         u3(ix,iy,iz) = 0.d0
         u4(ix,iy,iz) = 0.d0
         u5(ix,iy,iz) = 0.01d0/(gam-1.d0)+0.1d0
c
         f1(ix,iy,iz) = q1(ix,iy,iz)*1.d0
         f2(ix,iy,iz) = q1(ix,iy,iz)*0.d0
c
         endif
 100  continue
c
      endif
```

```
c-----------------------------------------------------------------------
c  outflow x-boundary
c-----------------------------------------------------------------------
      nsize = 12*2*ny
c
      if (pi.eq.master) then
         ndest = np-1
      else
         ndest = pi-1
      endif
c
      do 101 iz=1,nz
         do 102 iy=1,ny
            w6( 1,1,iy) = q1(1,iy,iz)
            w6( 1,2,iy) = q1(2,iy,iz)
            w6( 2,1,iy) = q2(1,iy,iz)
            w6( 2,2,iy) = q2(2,iy,iz)
            w6( 3,1,iy) = q3(1,iy,iz)
            w6( 3,2,iy) = q3(2,iy,iz)
            w6( 4,1,iy) = q4(1,iy,iz)
            w6( 4,2,iy) = q4(2,iy,iz)
            w6( 5,1,iy) = q5(1,iy,iz)
            w6( 5,2,iy) = q5(2,iy,iz)
c
            w6( 6,1,iy) = u1(1,iy,iz)
            w6( 6,2,iy) = u1(2,iy,iz)
            w6( 7,1,iy) = u2(1,iy,iz)
            w6( 7,2,iy) = u2(2,iy,iz)
            w6( 8,1,iy) = u3(1,iy,iz)
            w6( 8,2,iy) = u3(2,iy,iz)
            w6( 9,1,iy) = u4(1,iy,iz)
            w6( 9,2,iy) = u4(2,iy,iz)
            w6(10,1,iy) = u5(1,iy,iz)
            w6(10,2,iy) = u5(2,iy,iz)
c
            w6(11,1,iy) = f1(1,iy,iz)
            w6(11,2,iy) = f1(2,iy,iz)
            w6(12,1,iy) = f2(1,iy,iz)
            w6(12,2,iy) = f2(2,iy,iz)
 102     continue
c
         call MPI_SEND (w6,nsize,MPI_DOUBLE_PRECISION,ndest,pi,
     +                  MPI_COMM_WORLD,err)
c
         call MPI_BARRIER (MPI_COMM_WORLD,err)
```

```
c
         call MPI_RECV (w7,nsize,MPI_DOUBLE_PRECISION,MPI_ANY_SOURCE,
     +                  MPI_ANY_TAG,MPI_COMM_WORLD,status,err)
c
         do 103 iy=1,ny
            q1(nxp+1,iy,iz) = w7( 1,1,iy)
            q1(nxp+2,iy,iz) = w7( 1,2,iy)
            q2(nxp+1,iy,iz) = w7( 2,1,iy)
            q2(nxp+2,iy,iz) = w7( 2,2,iy)
            q3(nxp+1,iy,iz) = w7( 3,1,iy)
            q3(nxp+2,iy,iz) = w7( 3,2,iy)
            q4(nxp+1,iy,iz) = w7( 4,1,iy)
            q4(nxp+2,iy,iz) = w7( 4,2,iy)
            q5(nxp+1,iy,iz) = w7( 5,1,iy)
            q5(nxp+2,iy,iz) = w7( 5,2,iy)
c
            u1(nxp+1,iy,iz) = w7( 6,1,iy)
            u1(nxp+2,iy,iz) = w7( 6,2,iy)
            u2(nxp+1,iy,iz) = w7( 7,1,iy)
            u2(nxp+2,iy,iz) = w7( 7,2,iy)
            u3(nxp+1,iy,iz) = w7( 8,1,iy)
            u3(nxp+2,iy,iz) = w7( 8,2,iy)
            u4(nxp+1,iy,iz) = w7( 9,1,iy)
            u4(nxp+2,iy,iz) = w7( 9,2,iy)
            u5(nxp+1,iy,iz) = w7(10,1,iy)
            u5(nxp+2,iy,iz) = w7(10,2,iy)
c
            f1(nxp+1,iy,iz) = w7(11,1,iy)
            f1(nxp+2,iy,iz) = w7(11,2,iy)
            f2(nxp+1,iy,iz) = w7(12,1,iy)
            f2(nxp+2,iy,iz) = w7(12,2,iy)
 103     continue
 101  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      if (pi.eq.np-1) then
         ndest = master
      else
         ndest = pi+1
      endif
c
      do 104 iz=1,nz
         do 105 iy=1,ny
            w6( 1,1,iy) = q1(nxp-1,iy,iz)
```

```
            w6( 1,2,iy) = q1(nxp  ,iy,iz)
            w6( 2,1,iy) = q2(nxp-1,iy,iz)
            w6( 2,2,iy) = q2(nxp  ,iy,iz)
            w6( 3,1,iy) = q3(nxp-1,iy,iz)
            w6( 3,2,iy) = q3(nxp  ,iy,iz)
            w6( 4,1,iy) = q4(nxp-1,iy,iz)
            w6( 4,2,iy) = q4(nxp  ,iy,iz)
            w6( 5,1,iy) = q5(nxp-1,iy,iz)
            w6( 5,2,iy) = q5(nxp  ,iy,iz)
c
            w6( 6,1,iy) = u1(nxp-1,iy,iz)
            w6( 6,2,iy) = u1(nxp  ,iy,iz)
            w6( 7,1,iy) = u2(nxp-1,iy,iz)
            w6( 7,2,iy) = u2(nxp  ,iy,iz)
            w6( 8,1,iy) = u3(nxp-1,iy,iz)
            w6( 8,2,iy) = u3(nxp  ,iy,iz)
            w6( 9,1,iy) = u4(nxp-1,iy,iz)
            w6( 9,2,iy) = u4(nxp  ,iy,iz)
            w6(10,1,iy) = u5(nxp-1,iy,iz)
            w6(10,2,iy) = u5(nxp  ,iy,iz)
c
            w6(11,1,iy) = f1(nxp-1,iy,iz)
            w6(11,2,iy) = f1(nxp  ,iy,iz)
            w6(12,1,iy) = f2(nxp-1,iy,iz)
            w6(12,2,iy) = f2(nxp  ,iy,iz)
 105     continue
c
         call MPI_SEND (w6,nsize,MPI_DOUBLE_PRECISION,ndest,pi,
     +                  MPI_COMM_WORLD,err)
c
         call MPI_BARRIER (MPI_COMM_WORLD,err)
c
         call MPI_RECV (w7,nsize,MPI_DOUBLE_PRECISION,MPI_ANY_SOURCE,
     +                  MPI_ANY_TAG,MPI_COMM_WORLD,status,err)
c
         do 106 iy=1,ny
            q1(-1,iy,iz) = w7( 1,1,iy)
            q1( 0,iy,iz) = w7( 1,2,iy)
            q2(-1,iy,iz) = w7( 2,1,iy)
            q2( 0,iy,iz) = w7( 2,2,iy)
            q3(-1,iy,iz) = w7( 3,1,iy)
            q3( 0,iy,iz) = w7( 3,2,iy)
            q4(-1,iy,iz) = w7( 4,1,iy)
            q4( 0,iy,iz) = w7( 4,2,iy)
            q5(-1,iy,iz) = w7( 5,1,iy)
```

```
            q5( 0,iy,iz) = w7( 5,2,iy)
c
            u1(-1,iy,iz) = w7( 6,1,iy)
            u1( 0,iy,iz) = w7( 6,2,iy)
            u2(-1,iy,iz) = w7( 7,1,iy)
            u2( 0,iy,iz) = w7( 7,2,iy)
            u3(-1,iy,iz) = w7( 8,1,iy)
            u3( 0,iy,iz) = w7( 8,2,iy)
            u4(-1,iy,iz) = w7( 9,1,iy)
            u4( 0,iy,iz) = w7( 9,2,iy)
            u5(-1,iy,iz) = w7(10,1,iy)
            u5( 0,iy,iz) = w7(10,2,iy)
c
            f1(-1,iy,iz) = w7(11,1,iy)
            f1( 0,iy,iz) = w7(11,2,iy)
            f2(-1,iy,iz) = w7(12,1,iy)
            f2( 0,iy,iz) = w7(12,2,iy)
 106     continue
 104  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      if (pi.eq.master) then
         do 107 iz=1,nz
         do 107 iy=1,ny
            q1(-1,iy,iz) = q1(1,iy,iz)
            q1( 0,iy,iz) = q1(1,iy,iz)
            q2(-1,iy,iz) = q2(1,iy,iz)
            q2( 0,iy,iz) = q2(1,iy,iz)
            q3(-1,iy,iz) = q3(1,iy,iz)
            q3( 0,iy,iz) = q3(1,iy,iz)
            q4(-1,iy,iz) = q4(1,iy,iz)
            q4( 0,iy,iz) = q4(1,iy,iz)
            q5(-1,iy,iz) = q5(1,iy,iz)
            q5( 0,iy,iz) = q5(1,iy,iz)
c
            u1(-1,iy,iz) = u1(1,iy,iz)
            u1( 0,iy,iz) = u1(1,iy,iz)
            u2(-1,iy,iz) = u2(1,iy,iz)
            u2( 0,iy,iz) = u2(1,iy,iz)
            u3(-1,iy,iz) = u3(1,iy,iz)
            u3( 0,iy,iz) = u3(1,iy,iz)
            u4(-1,iy,iz) = u4(1,iy,iz)
            u4( 0,iy,iz) = u4(1,iy,iz)
            u5(-1,iy,iz) = u5(1,iy,iz)
```

```
            u5( 0,iy,iz) = u5(1,iy,iz)
c
            f1(-1,iy,iz) = f1(1,iy,iz)
            f1( 0,iy,iz) = f1(1,iy,iz)
            f2(-1,iy,iz) = f2(1,iy,iz)
            f2( 0,iy,iz) = f2(1,iy,iz)
 107     continue
      endif
c
      if (pi.eq.np-1) then
         do 108 iz=1,nz
         do 108 iy=1,ny
            q1(nxp+1,iy,iz) = q1(nxp,iy,iz)
            q1(nxp+2,iy,iz) = q1(nxp,iy,iz)
            q2(nxp+1,iy,iz) = q2(nxp,iy,iz)
            q2(nxp+2,iy,iz) = q2(nxp,iy,iz)
            q3(nxp+1,iy,iz) = q3(nxp,iy,iz)
            q3(nxp+2,iy,iz) = q3(nxp,iy,iz)
            q4(nxp+1,iy,iz) = q4(nxp,iy,iz)
            q4(nxp+2,iy,iz) = q4(nxp,iy,iz)
            q5(nxp+1,iy,iz) = q5(nxp,iy,iz)
            q5(nxp+2,iy,iz) = q5(nxp,iy,iz)
c
            u1(nxp+1,iy,iz) = u1(nxp,iy,iz)
            u1(nxp+2,iy,iz) = u1(nxp,iy,iz)
            u2(nxp+1,iy,iz) = u2(nxp,iy,iz)
            u2(nxp+2,iy,iz) = u2(nxp,iy,iz)
            u3(nxp+1,iy,iz) = u3(nxp,iy,iz)
            u3(nxp+2,iy,iz) = u3(nxp,iy,iz)
            u4(nxp+1,iy,iz) = u4(nxp,iy,iz)
            u4(nxp+2,iy,iz) = u4(nxp,iy,iz)
            u5(nxp+1,iy,iz) = u5(nxp,iy,iz)
            u5(nxp+2,iy,iz) = u5(nxp,iy,iz)
c
            f1(nxp+1,iy,iz) = f1(nxp,iy,iz)
            f1(nxp+2,iy,iz) = f1(nxp,iy,iz)
            f2(nxp+1,iy,iz) = f2(nxp,iy,iz)
            f2(nxp+2,iy,iz) = f2(nxp,iy,iz)
 108     continue
      endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  outflow y-boundary
c-----------------------------------------------------------------------
```

```
      do 109 ix=1,nxp
      do 109 iz=1,nz
         q1(ix,  -1,iz) = q1(ix, 1,iz)
         q1(ix,   0,iz) = q1(ix, 1,iz)
         q1(ix,ny+1,iz) = q1(ix,ny,iz)
         q1(ix,ny+2,iz) = q1(ix,ny,iz)
         q2(ix,  -1,iz) = q2(ix, 1,iz)
         q2(ix,   0,iz) = q2(ix, 1,iz)
         q2(ix,ny+1,iz) = q2(ix,ny,iz)
         q2(ix,ny+2,iz) = q2(ix,ny,iz)
         q3(ix,  -1,iz) = q3(ix, 1,iz)
         q3(ix,   0,iz) = q3(ix, 1,iz)
         q3(ix,ny+1,iz) = q3(ix,ny,iz)
         q3(ix,ny+2,iz) = q3(ix,ny,iz)
         q4(ix,  -1,iz) = q4(ix, 1,iz)
         q4(ix,   0,iz) = q4(ix, 1,iz)
         q4(ix,ny+1,iz) = q4(ix,ny,iz)
         q4(ix,ny+2,iz) = q4(ix,ny,iz)
         q5(ix,  -1,iz) = q5(ix, 1,iz)
         q5(ix,   0,iz) = q5(ix, 1,iz)
         q5(ix,ny+1,iz) = q5(ix,ny,iz)
         q5(ix,ny+2,iz) = q5(ix,ny,iz)
c
         u1(ix,  -1,iz) = u1(ix, 1,iz)
         u1(ix,   0,iz) = u1(ix, 1,iz)
         u1(ix,ny+1,iz) = u1(ix,ny,iz)
         u1(ix,ny+2,iz) = u1(ix,ny,iz)
         u2(ix,  -1,iz) = u2(ix, 1,iz)
         u2(ix,   0,iz) = u2(ix, 1,iz)
         u2(ix,ny+1,iz) = u2(ix,ny,iz)
         u2(ix,ny+2,iz) = u2(ix,ny,iz)
         u3(ix,  -1,iz) = u3(ix, 1,iz)
         u3(ix,   0,iz) = u3(ix, 1,iz)
         u3(ix,ny+1,iz) = u3(ix,ny,iz)
         u3(ix,ny+2,iz) = u3(ix,ny,iz)
         u4(ix,  -1,iz) = u4(ix, 1,iz)
         u4(ix,   0,iz) = u4(ix, 1,iz)
         u4(ix,ny+1,iz) = u4(ix,ny,iz)
         u4(ix,ny+2,iz) = u4(ix,ny,iz)
         u5(ix,  -1,iz) = u5(ix, 1,iz)
         u5(ix,   0,iz) = u5(ix, 1,iz)
         u5(ix,ny+1,iz) = u5(ix,ny,iz)
         u5(ix,ny+2,iz) = u5(ix,ny,iz)
c
         f1(ix,  -1,iz) = f1(ix, 1,iz)
```

```
         f1(ix,   0,iz) = f1(ix, 1,iz)
         f1(ix,ny+1,iz) = f1(ix,ny,iz)
         f1(ix,ny+2,iz) = f1(ix,ny,iz)
         f2(ix,  -1,iz) = f2(ix, 1,iz)
         f2(ix,   0,iz) = f2(ix, 1,iz)
         f2(ix,ny+1,iz) = f2(ix,ny,iz)
         f2(ix,ny+2,iz) = f2(ix,ny,iz)
 109  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  outflow z-boundary
c-----------------------------------------------------------------------
      do 110 iy=1,ny
      do 110 ix=1,nxp
         q1(ix,iy,  -1) = q1(ix,iy, 1)
         q1(ix,iy,   0) = q1(ix,iy, 1)
         q1(ix,iy,nz+1) = q1(ix,iy,nz)
         q1(ix,iy,nz+2) = q1(ix,iy,nz)
         q2(ix,iy,  -1) = q2(ix,iy, 1)
         q2(ix,iy,   0) = q2(ix,iy, 1)
         q2(ix,iy,nz+1) = q2(ix,iy,nz)
         q2(ix,iy,nz+2) = q2(ix,iy,nz)
         q3(ix,iy,  -1) = q3(ix,iy, 1)
         q3(ix,iy,   0) = q3(ix,iy, 1)
         q3(ix,iy,nz+1) = q3(ix,iy,nz)
         q3(ix,iy,nz+2) = q3(ix,iy,nz)
         q4(ix,iy,  -1) = q4(ix,iy, 1)
         q4(ix,iy,   0) = q4(ix,iy, 1)
         q4(ix,iy,nz+1) = q4(ix,iy,nz)
         q4(ix,iy,nz+2) = q4(ix,iy,nz)
         q5(ix,iy,  -1) = q5(ix,iy, 1)
         q5(ix,iy,   0) = q5(ix,iy, 1)
         q5(ix,iy,nz+1) = q5(ix,iy,nz)
         q5(ix,iy,nz+2) = q5(ix,iy,nz)
c
         u1(ix,iy,  -1) = u1(ix,iy, 1)
         u1(ix,iy,   0) = u1(ix,iy, 1)
         u1(ix,iy,nz+1) = u1(ix,iy,nz)
         u1(ix,iy,nz+2) = u1(ix,iy,nz)
         u2(ix,iy,  -1) = u2(ix,iy, 1)
         u2(ix,iy,   0) = u2(ix,iy, 1)
         u2(ix,iy,nz+1) = u2(ix,iy,nz)
         u2(ix,iy,nz+2) = u2(ix,iy,nz)
         u3(ix,iy,  -1) = u3(ix,iy, 1)
```

```
         u3(ix,iy,   0) = u3(ix,iy, 1)
         u3(ix,iy,nz+1) = u3(ix,iy,nz)
         u3(ix,iy,nz+2) = u3(ix,iy,nz)
         u4(ix,iy,  -1) = u4(ix,iy, 1)
         u4(ix,iy,   0) = u4(ix,iy, 1)
         u4(ix,iy,nz+1) = u4(ix,iy,nz)
         u4(ix,iy,nz+2) = u4(ix,iy,nz)
         u5(ix,iy,  -1) = u5(ix,iy, 1)
         u5(ix,iy,   0) = u5(ix,iy, 1)
         u5(ix,iy,nz+1) = u5(ix,iy,nz)
         u5(ix,iy,nz+2) = u5(ix,iy,nz)
c
         f1(ix,iy,  -1) = f1(ix,iy, 1)
         f1(ix,iy,   0) = f1(ix,iy, 1)
         f1(ix,iy,nz+1) = f1(ix,iy,nz)
         f1(ix,iy,nz+2) = f1(ix,iy,nz)
         f2(ix,iy,  -1) = f2(ix,iy, 1)
         f2(ix,iy,   0) = f2(ix,iy, 1)
         f2(ix,iy,nz+1) = f2(ix,iy,nz)
         f2(ix,iy,nz+2) = f2(ix,iy,nz)
 110  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
      subroutine tvd3
c-----------------------------------------------------------------------
c  3d tvd
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz, nmod, ncycle, iflag(0:5,0:2)
c-----------------------------------------------------------------------
c  order of integrations
c-----------------------------------------------------------------------
      nmod = mod(nstep,6)
      ncycle = 0
      if ((nmod.eq.0).or.(nstep.eq.1)) call tstep
c
      iflag(0,0) = 1
      iflag(0,1) = 2
      iflag(0,2) = 3
      iflag(1,0) = 3
      iflag(1,1) = 2
      iflag(1,2) = 1
      iflag(2,0) = 2
      iflag(2,1) = 3
      iflag(2,2) = 1
      iflag(3,0) = 1
      iflag(3,1) = 3
      iflag(3,2) = 2
      iflag(4,0) = 3
      iflag(4,1) = 1
      iflag(4,2) = 2
      iflag(5,0) = 2
      iflag(5,1) = 1
      iflag(5,2) = 3
c
 204  continue
      if (ncycle.ge.3) then
         goto 205
      elseif (iflag(nmod,ncycle).eq.1) then
         goto 201
      elseif (iflag(nmod,ncycle).eq.2) then
         goto 202
      elseif (iflag(nmod,ncycle).eq.3) then
         goto 203
```

```
      else
         stop 'iflag not equal to 1, 2 or 3'
      endif
c-----------------------------------------------------------------------
c  integration along x
c-----------------------------------------------------------------------
 201  continue
c
      nw = nxp
      dr = dx
c
      do 101 iz=1,nz
      do 101 iy=1,ny
         do 102 ix=-1,nxp+2
            w0(1,ix) = q1(ix,iy,iz)
            w0(2,ix) = q2(ix,iy,iz)
            w0(3,ix) = q3(ix,iy,iz)
            w0(4,ix) = q4(ix,iy,iz)
            w0(5,ix) = q5(ix,iy,iz)
c
            w4(1,ix) = u1(ix,iy,iz)
            w4(2,ix) = u2(ix,iy,iz)
            w4(3,ix) = u3(ix,iy,iz)
            w4(4,ix) = u4(ix,iy,iz)
            w4(5,ix) = u5(ix,iy,iz)
c
            w5(1,ix) = f1(ix,iy,iz)
            w5(2,ix) = f2(ix,iy,iz)
 102     continue
c
         call rhdtvd
         call lorenz
c
         do 103 ix=1,nxp
            q1(ix,iy,iz) = w0(1,ix)
            q2(ix,iy,iz) = w0(2,ix)
            q3(ix,iy,iz) = w0(3,ix)
            q4(ix,iy,iz) = w0(4,ix)
            q5(ix,iy,iz) = w0(5,ix)
c
            u1(ix,iy,iz) = w4(1,ix)
            u2(ix,iy,iz) = w4(2,ix)
            u3(ix,iy,iz) = w4(3,ix)
            u4(ix,iy,iz) = w4(4,ix)
            u5(ix,iy,iz) = w4(5,ix)
```

```
c
            f1(ix,iy,iz) = w5(1,ix)
            f2(ix,iy,iz) = w5(2,ix)
 103     continue
 101  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call prot
      call bound
c
      ncycle = ncycle+1
      goto 204
c-------------------------------------------------------------------------
c  integration along y
c-------------------------------------------------------------------------
 202  continue
c
      nw = ny
      dr = dy
c
      do 111 ix=1,nxp
      do 111 iz=1,nz
         do 112 iy=-1,ny+2
            w0(1,iy) = q1(ix,iy,iz)
            w0(2,iy) = q3(ix,iy,iz)
            w0(3,iy) = q4(ix,iy,iz)
            w0(4,iy) = q2(ix,iy,iz)
            w0(5,iy) = q5(ix,iy,iz)
c
            w4(1,iy) = u1(ix,iy,iz)
            w4(2,iy) = u3(ix,iy,iz)
            w4(3,iy) = u4(ix,iy,iz)
            w4(4,iy) = u2(ix,iy,iz)
            w4(5,iy) = u5(ix,iy,iz)
c
            w5(1,iy) = f1(ix,iy,iz)
            w5(2,iy) = f2(ix,iy,iz)
 112     continue
c
         call rhdtvd
         call lorenz
c
         do 113 iy=1,ny
            q1(ix,iy,iz) = w0(1,iy)
```

```
            q3(ix,iy,iz) = w0(2,iy)
            q4(ix,iy,iz) = w0(3,iy)
            q2(ix,iy,iz) = w0(4,iy)
            q5(ix,iy,iz) = w0(5,iy)
c
            u1(ix,iy,iz) = w4(1,iy)
            u3(ix,iy,iz) = w4(2,iy)
            u4(ix,iy,iz) = w4(3,iy)
            u2(ix,iy,iz) = w4(4,iy)
            u5(ix,iy,iz) = w4(5,iy)
c
            f1(ix,iy,iz) = w5(1,iy)
            f2(ix,iy,iz) = w5(2,iy)
 113     continue
 111  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call prot
      call bound
c
      ncycle = ncycle+1
      goto 204
c-------------------------------------------------------------------------
c  integration along z
c-------------------------------------------------------------------------
 203  continue
c
      nw = nz
      dr = dz
c
      do 121 iy=1,ny
      do 121 ix=1,nxp
         do 122 iz=-1,nz+2
            w0(1,iz) = q1(ix,iy,iz)
            w0(2,iz) = q4(ix,iy,iz)
            w0(3,iz) = q2(ix,iy,iz)
            w0(4,iz) = q3(ix,iy,iz)
            w0(5,iz) = q5(ix,iy,iz)
c
            w4(1,iz) = u1(ix,iy,iz)
            w4(2,iz) = u4(ix,iy,iz)
            w4(3,iz) = u2(ix,iy,iz)
            w4(4,iz) = u3(ix,iy,iz)
            w4(5,iz) = u5(ix,iy,iz)
```

```
c
            w5(1,iz) = f1(ix,iy,iz)
            w5(2,iz) = f2(ix,iy,iz)
 122     continue
c
         call rhdtvd
         call lorenz
c
         do 123 iz=1,nz
            q1(ix,iy,iz) = w0(1,iz)
            q4(ix,iy,iz) = w0(2,iz)
            q2(ix,iy,iz) = w0(3,iz)
            q3(ix,iy,iz) = w0(4,iz)
            q5(ix,iy,iz) = w0(5,iz)
c
            u1(ix,iy,iz) = w4(1,iz)
            u4(ix,iy,iz) = w4(2,iz)
            u2(ix,iy,iz) = w4(3,iz)
            u3(ix,iy,iz) = w4(4,iz)
            u5(ix,iy,iz) = w4(5,iz)
c
            f1(ix,iy,iz) = w5(1,iz)
            f2(ix,iy,iz) = w5(2,iz)
 123     continue
 121  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call prot
      call bound
c
      ncycle = ncycle+1
      goto 204
c-----------------------------------------------------------------------
c  end of integrations
c-----------------------------------------------------------------------
 205  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
      subroutine tstep
c-----------------------------------------------------------------------
c  time step
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz, i
      real*8 vxmax, vymax, vzmax, dtp
c
      vxmax = 0.d0
      vymax = 0.d0
      vzmax = 0.d0
c-----------------------------------------------------------------------
c  along x
c-----------------------------------------------------------------------
      do 101 iz=1,nz
      do 101 iy=1,ny
c-----------------------------------------------------------------------
c  assignment
c-----------------------------------------------------------------------
      do 102 ix=-1,nxp+2
         w4(1,ix) = u1(ix,iy,iz)
         w4(2,ix) = u2(ix,iy,iz)
         w4(3,ix) = u3(ix,iy,iz)
         w4(4,ix) = u4(ix,iy,iz)
         w4(5,ix) = u5(ix,iy,iz)
 102  continue
c-----------------------------------------------------------------------
c  vx(i), vy(i), vz(i), p(i), h(i)
c-----------------------------------------------------------------------
      do 103 i=-1,nxp+2
         w2(1,i) = w4(2,i)
         w2(2,i) = w4(3,i)
         w2(3,i) = w4(4,i)
         w2(4,i) = (gam-1.d0)*(w4(5,i)-w4(1,i))
         w2(5,i) = (w4(5,i)+w2(4,i))/w4(1,i)
 103  continue
c-----------------------------------------------------------------------
c  vx(i+1/2), vy(i+1/2), vz(i+1/2), h(i+1/2), cs(i+1/2)
c-----------------------------------------------------------------------
      do 104 i=-1,nxp+1
         w2(11,i) = (w2(1,i)+w2(1,i+1))/2.d0
```

```
         w2(12,i) = (w2(2,i)+w2(2,i+1))/2.d0
         w2(13,i) = (w2(3,i)+w2(3,i+1))/2.d0
         w2(14,i) = (w2(5,i)+w2(5,i+1))/2.d0
         w2(15,i) = sqrt((gam-1.d0)*(w2(14,i)-1.d0)/w2(14,i))
 104  continue
c-----------------------------------------------------------------------
c  vxmax
c-----------------------------------------------------------------------
      do 105 i=0,nxp
         vxmax = max(vxmax,((1.d0-w2(15,i)**2)*abs(w2(11,i))
     +               +sqrt((1.d0-w2(11,i)**2-w2(12,i)**2-w2(13,i)**2)
     +               *w2(15,i)**2*(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2-(1.d0-w2(15,i)**2)
     +               *w2(11,i)**2)))/(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2))
 105  continue
c-----------------------------------------------------------------------
c  along x
c-----------------------------------------------------------------------
 101  continue
c-----------------------------------------------------------------------
c  along y
c-----------------------------------------------------------------------
      do 111 ix=1,nxp
      do 111 iz=1,nz
c-----------------------------------------------------------------------
c  assignment
c-----------------------------------------------------------------------
      do 112 iy=-1,ny+2
         w4(1,iy) = u1(ix,iy,iz)
         w4(2,iy) = u3(ix,iy,iz)
         w4(3,iy) = u4(ix,iy,iz)
         w4(4,iy) = u2(ix,iy,iz)
         w4(5,iy) = u5(ix,iy,iz)
 112  continue
c-----------------------------------------------------------------------
c  vx(i), vy(i), vz(i), p(i), h(i)
c-----------------------------------------------------------------------
      do 113 i=-1,ny+2
         w2(1,i) = w4(2,i)
         w2(2,i) = w4(3,i)
         w2(3,i) = w4(4,i)
         w2(4,i) = (gam-1.d0)*(w4(5,i)-w4(1,i))
         w2(5,i) = (w4(5,i)+w2(4,i))/w4(1,i)
 113  continue
```

```
c-----------------------------------------------------------------------
c  vx(i+1/2), vy(i+1/2), vz(i+1/2), h(i+1/2), cs(i+1/2)
c-----------------------------------------------------------------------
      do 114 i=-1,ny+1
         w2(11,i) = (w2(1,i)+w2(1,i+1))/2.d0
         w2(12,i) = (w2(2,i)+w2(2,i+1))/2.d0
         w2(13,i) = (w2(3,i)+w2(3,i+1))/2.d0
         w2(14,i) = (w2(5,i)+w2(5,i+1))/2.d0
         w2(15,i) = sqrt((gam-1.d0)*(w2(14,i)-1.d0)/w2(14,i))
 114  continue
c-----------------------------------------------------------------------
c  vymax
c-----------------------------------------------------------------------
      do 115 i=0,ny
         vymax = max(vymax,((1.d0-w2(15,i)**2)*abs(w2(11,i))
     +               +sqrt((1.d0-w2(11,i)**2-w2(12,i)**2-w2(13,i)**2)
     +               *w2(15,i)**2*(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2-(1.d0-w2(15,i)**2)
     +               *w2(11,i)**2)))/(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2))
 115  continue
c-----------------------------------------------------------------------
c  along y
c-----------------------------------------------------------------------
 111  continue
c-----------------------------------------------------------------------
c  along z
c-----------------------------------------------------------------------
      do 121 iy=1,ny
      do 121 ix=1,nxp
c-----------------------------------------------------------------------
c  assignment
c-----------------------------------------------------------------------
      do 122 iz=-1,nz+2
         w4(1,iz) = u1(ix,iy,iz)
         w4(2,iz) = u4(ix,iy,iz)
         w4(3,iz) = u2(ix,iy,iz)
         w4(4,iz) = u3(ix,iy,iz)
         w4(5,iz) = u5(ix,iy,iz)
 122  continue
c-----------------------------------------------------------------------
c  vx(i), vy(i), vz(i), p(i), h(i)
c-----------------------------------------------------------------------
      do 123 i=-1,nz+2
         w2(1,i) = w4(2,i)
```

```
         w2(2,i) = w4(3,i)
         w2(3,i) = w4(4,i)
         w2(4,i) = (gam-1.d0)*(w4(5,i)-w4(1,i))
         w2(5,i) = (w4(5,i)+w2(4,i))/w4(1,i)
 123  continue
c-----------------------------------------------------------------------
c  vx(i+1/2), vy(i+1/2), vz(i+1/2), h(i+1/2), cs(i+1/2)
c-----------------------------------------------------------------------
      do 124 i=-1,nz+1
         w2(11,i) = (w2(1,i)+w2(1,i+1))/2.d0
         w2(12,i) = (w2(2,i)+w2(2,i+1))/2.d0
         w2(13,i) = (w2(3,i)+w2(3,i+1))/2.d0
         w2(14,i) = (w2(5,i)+w2(5,i+1))/2.d0
         w2(15,i) = sqrt((gam-1.d0)*(w2(14,i)-1.d0)/w2(14,i))
 124  continue
c-----------------------------------------------------------------------
c  vzmax
c-----------------------------------------------------------------------
      do 125 i=0,nz
         vzmax = max(vzmax,((1.d0-w2(15,i)**2)*abs(w2(11,i))
     +               +sqrt((1.d0-w2(11,i)**2-w2(12,i)**2-w2(13,i)**2)
     +               *w2(15,i)**2*(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2-(1.d0-w2(15,i)**2)
     +               *w2(11,i)**2)))/(1.d0-(w2(11,i)**2+w2(12,i)**2
     +               +w2(13,i)**2)*w2(15,i)**2))
 125  continue
c-----------------------------------------------------------------------
c  along z
c-----------------------------------------------------------------------
 121  continue
c-----------------------------------------------------------------------
c  dtp
c-----------------------------------------------------------------------
      dtp = min(cour/vxmax*dx,cour/vymax*dy,cour/vzmax*dz)
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  dt
c-----------------------------------------------------------------------
      call MPI_REDUCE (dtp,dt,1,MPI_DOUBLE_PRECISION,MPI_MIN,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_BCAST (dt,1,MPI_DOUBLE_PRECISION,master,
     +                MPI_COMM_WORLD,err)
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
```

```
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
      subroutine rhdtvd
c-----------------------------------------------------------------------
c  rhd tvd code based on Harten (1983) and Ryu et al. (1993)
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer i
c-----------------------------------------------------------------------
c  vx(i), vy(i), vz(i), p(i), h(i)
c-----------------------------------------------------------------------
      do 101 i=-1,nw+2
         w2(1,i) = w4(2,i)
         w2(2,i) = w4(3,i)
         w2(3,i) = w4(4,i)
         w2(4,i) = (gam-1.d0)*(w4(5,i)-w4(1,i))
         w2(5,i) = (w4(5,i)+w2(4,i))/w4(1,i)
 101  continue
c-----------------------------------------------------------------------
c  vx(i+1/2), vy(i+1/2), vz(i+1/2), h(i+1/2), cs(i+1/2)
c-----------------------------------------------------------------------
      do 102 i=-1,nw+1
         w2(11,i) = (w2(1,i)+w2(1,i+1))/2.d0
         w2(12,i) = (w2(2,i)+w2(2,i+1))/2.d0
         w2(13,i) = (w2(3,i)+w2(3,i+1))/2.d0
         w2(14,i) = (w2(5,i)+w2(5,i+1))/2.d0
         w2(15,i) = sqrt((gam-1.d0)*(w2(14,i)-1.d0)/w2(14,i))
 102  continue
c-----------------------------------------------------------------------
c  gam(i+1/2)
c-----------------------------------------------------------------------
      do 103 i=-1,nw+1
         w2(16,i) = 1.d0/sqrt(1.d0-w2(11,i)**2-w2(12,i)**2-w2(13,i)**2)
 103  continue
c-----------------------------------------------------------------------
c  ak(i+1/2)
c-----------------------------------------------------------------------
      do 104 i=-1,nw+1
         w1(1,i) = ((1.d0-w2(15,i)**2)*w2(11,i)-sqrt((1.d0-w2(11,i)**2
     +             -w2(12,i)**2-w2(13,i)**2)*w2(15,i)**2*(1.d0
     +             -(w2(11,i)**2+w2(12,i)**2+w2(13,i)**2)*w2(15,i)**2
     +             -(1.d0-w2(15,i)**2)*w2(11,i)**2)))/(1.d0
     +             -(w2(11,i)**2+w2(12,i)**2+w2(13,i)**2)*w2(15,i)**2)
```

```
         w1(2,i) = w2(11,i)
         w1(3,i) = w2(11,i)
         w1(4,i) = w2(11,i)
         w1(5,i) = ((1.d0-w2(15,i)**2)*w2(11,i)+sqrt((1.d0-w2(11,i)**2
     +             -w2(12,i)**2-w2(13,i)**2)*w2(15,i)**2*(1.d0
     +             -(w2(11,i)**2+w2(12,i)**2+w2(13,i)**2)*w2(15,i)**2
     +             -(1.d0-w2(15,i)**2)*w2(11,i)**2)))/(1.d0
     +             -(w2(11,i)**2+w2(12,i)**2+w2(13,i)**2)*w2(15,i)**2)
 104  continue
c-----------------------------------------------------------------------
c  Rk(i+1/2)
c-----------------------------------------------------------------------
      do 105 i=-1,nw+1
         w1(11,i) = (1.d0-w2(11,i)*w1(1,i))
     +              /(w2(16,i)*w2(14,i)*(1.d0-w2(11,i)**2))
         w1(12,i) = w1(1,i)
         w1(13,i) = (1.d0-w2(11,i)*w1(1,i))*w2(12,i)/(1.d0-w2(11,i)**2)
         w1(14,i) = (1.d0-w2(11,i)*w1(1,i))*w2(13,i)/(1.d0-w2(11,i)**2)
         w1(15,i) = 1.d0
c
         w1(21,i) = -w2(16,i)*(2.d0*w2(14,i)-1.d0)*w2(12,i)/w2(14,i)
         w1(22,i) = 0.d0
         w1(23,i) = 1.d0
         w1(24,i) = 0.d0
         w1(25,i) = 0.d0
c
         w1(31,i) = (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+w2(14,i))/w2(16,i)/w2(14,i)
         w1(32,i) = w2(11,i)
         w1(33,i) = 0.d0
         w1(34,i) = 0.d0
         w1(35,i) = 1.d0
c
         w1(41,i) = -w2(16,i)*(2.d0*w2(14,i)-1.d0)*w2(13,i)/w2(14,i)
         w1(42,i) = 0.d0
         w1(43,i) = 0.d0
         w1(44,i) = 1.d0
         w1(45,i) = 0.d0
c
         w1(51,i) = (1.d0-w2(11,i)*w1(5,i))
     +              /(w2(16,i)*w2(14,i)*(1.d0-w2(11,i)**2))
         w1(52,i) = w1(5,i)
         w1(53,i) = (1.d0-w2(11,i)*w1(5,i))*w2(12,i)/(1.d0-w2(11,i)**2)
         w1(54,i) = (1.d0-w2(11,i)*w1(5,i))*w2(13,i)/(1.d0-w2(11,i)**2)
         w1(55,i) = 1.d0
```

```
 105  continue
c-----------------------------------------------------------------------
c  Lk(i+1/2)
c-----------------------------------------------------------------------
      do 106 i=-1,nw+1
         w1(61,i) = -w2(16,i)*w2(14,i)*(w2(11,i)-w1(5,i))
     +              /(w2(14,i)-1.d0)/(w1(1,i)-w1(5,i))
         w1(62,i) = -(w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+1.d0)*(w2(11,i)-w1(5,i))*w2(11,i)
     +              /((w2(14,i)-1.d0)*(1.d0-w2(11,i)**2)
     +              *(w1(1,i)-w1(5,i)))+1.d0/(w1(1,i)-w1(5,i))
         w1(63,i) = -w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(11,i)-w1(5,i))
     +              *w2(12,i)/(w2(14,i)-1.d0)/(w1(1,i)-w1(5,i))
         w1(64,i) = -w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(11,i)-w1(5,i))
     +              *w2(13,i)/(w2(14,i)-1.d0)/(w1(1,i)-w1(5,i))
         w1(65,i) = (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+1.d0)*(w2(11,i)-w1(5,i))
     +              /((w2(14,i)-1.d0)*(1.d0-w2(11,i)**2)
     +              *(w1(1,i)-w1(5,i)))-w1(5,i)/(w1(1,i)-w1(5,i))
c
         w1(71,i) = w2(16,i)*w2(14,i)*w2(12,i)/(w2(14,i)-1.d0)
         w1(72,i) = (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+w2(14,i))*w2(11,i)*w2(12,i)
     +              /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
         w1(73,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(12,i)**2
     +              /(w2(14,i)-1.d0)+1.d0
         w1(74,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(12,i)*w2(13,i)
     +              /(w2(14,i)-1.d0)
         w1(75,i) = -(w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+w2(14,i))*w2(12,i)
     +              /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
c
         w1(81,i) = w2(16,i)*w2(14,i)/(w2(14,i)-1.d0)
         w1(82,i) = (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+1.d0)*w2(11,i)
     +              /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
         w1(83,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(12,i)
     +              /(w2(14,i)-1.d0)
         w1(84,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(13,i)
     +              /(w2(14,i)-1.d0)
         w1(85,i) = -(w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +              +w2(13,i)**2)+1.d0)
     +              /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
c
         w1(91,i) = w2(16,i)*w2(14,i)*w2(13,i)/(w2(14,i)-1.d0)
```

```
          w1(92,i) = (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +               +w2(13,i)**2)+w2(14,i))*w2(11,i)*w2(13,i)
     +               /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
          w1(93,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(12,i)*w2(13,i)
     +               /(w2(14,i)-1.d0)
          w1(94,i) = w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*w2(13,i)**2
     +               /(w2(14,i)-1.d0)+1.d0
          w1(95,i) = -(w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +               +w2(13,i)**2)+w2(14,i))*w2(13,i)
     +               /(w2(14,i)-1.d0)/(1.d0-w2(11,i)**2)
c
          w1(101,i)= -w2(16,i)*w2(14,i)*(w2(11,i)-w1(1,i))
     +               /(w2(14,i)-1.d0)/(w1(5,i)-w1(1,i))
          w1(102,i)= -(w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +               +w2(13,i)**2)+1.d0)*(w2(11,i)-w1(1,i))*w2(11,i)
     +               /((w2(14,i)-1.d0)*(1.d0-w2(11,i)**2)
     +               *(w1(5,i)-w1(1,i)))+1.d0/(w1(5,i)-w1(1,i))
          w1(103,i)= -w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(11,i)-w1(1,i))
     +               *w2(12,i)/(w2(14,i)-1.d0)/(w1(5,i)-w1(1,i))
          w1(104,i)= -w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(11,i)-w1(1,i))
     +               *w2(13,i)/(w2(14,i)-1.d0)/(w1(5,i)-w1(1,i))
          w1(105,i)= (w2(16,i)**2*(2.d0*w2(14,i)-1.d0)*(w2(12,i)**2
     +               +w2(13,i)**2)+1.d0)*(w2(11,i)-w1(1,i))
     +               /((w2(14,i)-1.d0)*(1.d0-w2(11,i)**2)
     +               *(w1(5,i)-w1(1,i)))-w1(1,i)/(w1(5,i)-w1(1,i))
 106  continue
c-------------------------------------------------------------------------
c  alphak(i+1/2)
c-------------------------------------------------------------------------
       do 107 i=-1,nw+1
          w2(21,i) = w1(61,i)*(w0(1,i+1)-w0(1,i))
     +              +w1(62,i)*(w0(2,i+1)-w0(2,i))
     +               +w1(63,i)*(w0(3,i+1)-w0(3,i))
     +               +w1(64,i)*(w0(4,i+1)-w0(4,i))
     +               +w1(65,i)*(w0(5,i+1)-w0(5,i))
          w2(22,i) = w1(71,i)*(w0(1,i+1)-w0(1,i))
     +               +w1(72,i)*(w0(2,i+1)-w0(2,i))
     +               +w1(73,i)*(w0(3,i+1)-w0(3,i))
     +               +w1(74,i)*(w0(4,i+1)-w0(4,i))
     +               +w1(75,i)*(w0(5,i+1)-w0(5,i))
          w2(23,i) = w1(81,i)*(w0(1,i+1)-w0(1,i))
     +               +w1(82,i)*(w0(2,i+1)-w0(2,i))
     +               +w1(83,i)*(w0(3,i+1)-w0(3,i))
     +               +w1(84,i)*(w0(4,i+1)-w0(4,i))
     +               +w1(85,i)*(w0(5,i+1)-w0(5,i))
```

```
         w2(24,i) = w1(91,i)*(w0(1,i+1)-w0(1,i))
     +             +w1(92,i)*(w0(2,i+1)-w0(2,i))
     +             +w1(93,i)*(w0(3,i+1)-w0(3,i))
     +             +w1(94,i)*(w0(4,i+1)-w0(4,i))
     +             +w1(95,i)*(w0(5,i+1)-w0(5,i))
         w2(25,i) = w1(101,i)*(w0(1,i+1)-w0(1,i))
     +             +w1(102,i)*(w0(2,i+1)-w0(2,i))
     +             +w1(103,i)*(w0(3,i+1)-w0(3,i))
     +             +w1(104,i)*(w0(4,i+1)-w0(4,i))
     +             +w1(105,i)*(w0(5,i+1)-w0(5,i))
 107  continue
c-----------------------------------------------------------------------
c  ~gk(i+1/2)
c-----------------------------------------------------------------------
      do 108 i=-1,nw+1
         if (dt/dr*abs(w1(1,i)).ge.2.d0*eps1) then
            w2(31,i) = (dt/dr*abs(w1(1,i))-(dt/dr*w1(1,i))**2)
     +                 /2.d0*w2(21,i)
         else
            w2(31,i) = ((dt/dr*w1(1,i))**2/4.d0/eps1+eps1
     +                 -(dt/dr*w1(1,i))**2)/2.d0*w2(21,i)
         endif
         if (dt/dr*abs(w1(2,i)).ge.2.d0*eps2) then
            w2(32,i) = (dt/dr*abs(w1(2,i))-(dt/dr*w1(2,i))**2)
     +                 /2.d0*w2(22,i)
         else
            w2(32,i) = ((dt/dr*w1(2,i))**2/4.d0/eps2+eps2
     +                 -(dt/dr*w1(2,i))**2)/2.d0*w2(22,i)
         endif
         if (dt/dr*abs(w1(3,i)).ge.2.d0*eps2) then
            w2(33,i) = (dt/dr*abs(w1(3,i))-(dt/dr*w1(3,i))**2)
     +                 /2.d0*w2(23,i)
         else
            w2(33,i) = ((dt/dr*w1(3,i))**2/4.d0/eps2+eps2
     +                 -(dt/dr*w1(3,i))**2)/2.d0*w2(23,i)
         endif
         if (dt/dr*abs(w1(4,i)).ge.2.d0*eps2) then
            w2(34,i) = (dt/dr*abs(w1(4,i))-(dt/dr*w1(4,i))**2)
     +                 /2.d0*w2(24,i)
         else
            w2(34,i) = ((dt/dr*w1(4,i))**2/4.d0/eps2+eps2
     +                 -(dt/dr*w1(4,i))**2)/2.d0*w2(24,i)
         endif
         if (dt/dr*abs(w1(5,i)).ge.2.d0*eps1) then
            w2(35,i) = (dt/dr*abs(w1(5,i))-(dt/dr*w1(5,i))**2)
```

```
     +                 /2.d0*w2(25,i)
          else
             w2(35,i) = ((dt/dr*w1(5,i))**2/4.d0/eps1+eps1
     +                 -(dt/dr*w1(5,i))**2)/2.d0*w2(25,i)
          endif
 108  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
       do 109 i=0,nw+1
          w2(41,i) = sign(1.d0,w2(31,i))
     +               *max(0.d0,min(abs(w2(31,i)),
     +                         sign(1.d0,w2(31,i))*w2(31,i-1)))
          w2(42,i) = sign(1.d0,w2(32,i))
     +               *max(0.d0,min(abs(w2(32,i)),
     +                         sign(1.d0,w2(32,i))*w2(32,i-1)))
          w2(43,i) = sign(1.d0,w2(33,i))
     +               *max(0.d0,min(abs(w2(33,i)),
     +                         sign(1.d0,w2(33,i))*w2(33,i-1)))
          w2(44,i) = sign(1.d0,w2(34,i))
     +               *max(0.d0,min(abs(w2(34,i)),
     +                         sign(1.d0,w2(34,i))*w2(34,i-1)))
          w2(45,i) = sign(1.d0,w2(35,i))
     +               *max(0.d0,min(abs(w2(35,i)),
     +                         sign(1.d0,w2(35,i))*w2(35,i-1)))
 109  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
c       do 109 i=0,nw+1
c          w2(41,i) = sign(1.d0,w2(31,i))
c     +               *max(0.d0,min((abs(w2(31,i))
c     +                         +sign(1.d0,w2(31,i))*w2(31,i-1))/2.d0,
c     +                         2.d0*abs(w2(31,i)),
c     +                         2.d0*sign(1.d0,w2(31,i))*w2(31,i-1)))
c          w2(42,i) = sign(1.d0,w2(32,i))
c     +               *max(0.d0,min((abs(w2(32,i))
c     +                         +sign(1.d0,w2(32,i))*w2(32,i-1))/2.d0,
c     +                         2.d0*abs(w2(32,i)),
c     +                         2.d0*sign(1.d0,w2(32,i))*w2(32,i-1)))
c          w2(43,i) = sign(1.d0,w2(33,i))
c     +               *max(0.d0,min((abs(w2(33,i))
c     +                         +sign(1.d0,w2(33,i))*w2(33,i-1))/2.d0,
c     +                         2.d0*abs(w2(33,i)),
c     +                         2.d0*sign(1.d0,w2(33,i))*w2(33,i-1)))
```

```
c        w2(44,i) = sign(1.d0,w2(34,i))
c    +              *max(0.d0,min((abs(w2(34,i))
c    +                        +sign(1.d0,w2(34,i))*w2(34,i-1))/2.d0,
c    +                        2.d0*abs(w2(34,i)),
c    +                        2.d0*sign(1.d0,w2(34,i))*w2(34,i-1)))
c        w2(45,i) = sign(1.d0,w2(35,i))
c    +              *max(0.d0,min((abs(w2(35,i))
c    +                        +sign(1.d0,w2(35,i))*w2(35,i-1))/2.d0,
c    +                        2.d0*abs(w2(35,i)),
c    +                        2.d0*sign(1.d0,w2(35,i))*w2(35,i-1)))
c109  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
c     do 109 i=0,nw+1
c        w2(41,i) = sign(1.d0,w2(31,i))
c    +              *max(0.d0,min(abs(w2(31,i)),
c    +                        2.d0*sign(1.d0,w2(31,i))*w2(31,i-1)),
c    +                        min(2.d0*abs(w2(31,i)),
c    +                        sign(1.d0,w2(31,i))*w2(31,i-1)))
c        w2(42,i) = sign(1.d0,w2(32,i))
c    +              *max(0.d0,min(abs(w2(32,i)),
c    +                        2.d0*sign(1.d0,w2(32,i))*w2(32,i-1)),
c    +                        min(2.d0*abs(w2(32,i)),
c    +                        sign(1.d0,w2(32,i))*w2(32,i-1)))
c        w2(43,i) = sign(1.d0,w2(33,i))
c    +              *max(0.d0,min(abs(w2(33,i)),
c    +                        2.d0*sign(1.d0,w2(33,i))*w2(33,i-1)),
c    +                        min(2.d0*abs(w2(33,i)),
c    +                        sign(1.d0,w2(33,i))*w2(33,i-1)))
c        w2(44,i) = sign(1.d0,w2(34,i))
c    +              *max(0.d0,min(abs(w2(34,i)),
c    +                        2.d0*sign(1.d0,w2(34,i))*w2(34,i-1)),
c    +                        min(2.d0*abs(w2(34,i)),
c    +                        sign(1.d0,w2(34,i))*w2(34,i-1)))
c        w2(45,i) = sign(1.d0,w2(35,i))
c    +              *max(0.d0,min(abs(w2(35,i)),
c    +                        2.d0*sign(1.d0,w2(35,i))*w2(35,i-1)),
c    +                        min(2.d0*abs(w2(35,i)),
c    +                        sign(1.d0,w2(35,i))*w2(35,i-1)))
c109  continue
c-----------------------------------------------------------------------
c  gammak(i+1/2)
c-----------------------------------------------------------------------
      do 110 i=0,nw
```

```
         if (abs(w2(21,i)).ge.1.d-30) then
            w2(51,i) = (w2(41,i+1)-w2(41,i))/w2(21,i)
         else
            w2(51,i) = 0.d0
         endif
         if (abs(w2(22,i)).ge.1.d-30) then
            w2(52,i) = (w2(42,i+1)-w2(42,i))/w2(22,i)
         else
            w2(52,i) = 0.d0
         endif
         if (abs(w2(23,i)).ge.1.d-30) then
            w2(53,i) = (w2(43,i+1)-w2(43,i))/w2(23,i)
         else
            w2(53,i) = 0.d0
         endif
         if (abs(w2(24,i)).ge.1.d-30) then
            w2(54,i) = (w2(44,i+1)-w2(44,i))/w2(24,i)
         else
            w2(54,i) = 0.d0
         endif
         if (abs(w2(25,i)).ge.1.d-30) then
            w2(55,i) = (w2(45,i+1)-w2(45,i))/w2(25,i)
         else
            w2(55,i) = 0.d0
         endif
 110  continue
c-----------------------------------------------------------------------
c  betak(i+1/2)
c-----------------------------------------------------------------------
       do 111 i=0,nw
         if (abs(dt/dr*w1(1,i)+w2(51,i)).ge.2.d0*eps1) then
            w2(61,i) = abs(dt/dr*w1(1,i)+w2(51,i))*w2(21,i)
     +                 -(w2(41,i)+w2(41,i+1))
         else
            w2(61,i) = ((dt/dr*w1(1,i)+w2(51,i))**2/4.d0/eps1+eps1)
     +                 *w2(21,i)-(w2(41,i)+w2(41,i+1))
         endif
         if (abs(dt/dr*w1(2,i)+w2(52,i)).ge.2.d0*eps2) then
            w2(62,i) = abs(dt/dr*w1(2,i)+w2(52,i))*w2(22,i)
     +                 -(w2(42,i)+w2(42,i+1))
         else
            w2(62,i) = ((dt/dr*w1(2,i)+w2(52,i))**2/4.d0/eps2+eps2)
     +                 *w2(22,i)-(w2(42,i)+w2(42,i+1))
         endif
         if (abs(dt/dr*w1(3,i)+w2(53,i)).ge.2.d0*eps2) then
```

```
            w2(63,i) = abs(dt/dr*w1(3,i)+w2(53,i))*w2(23,i)
     +                 -(w2(43,i)+w2(43,i+1))
          else
            w2(63,i) = ((dt/dr*w1(3,i)+w2(53,i))**2/4.d0/eps2+eps2)
     +                 *w2(23,i)-(w2(43,i)+w2(43,i+1))
          endif
          if (abs(dt/dr*w1(4,i)+w2(54,i)).ge.2.d0*eps2) then
            w2(64,i) = abs(dt/dr*w1(4,i)+w2(54,i))*w2(24,i)
     +                 -(w2(44,i)+w2(44,i+1))
          else
            w2(64,i) = ((dt/dr*w1(4,i)+w2(54,i))**2/4.d0/eps2+eps2)
     +                 *w2(24,i)-(w2(44,i)+w2(44,i+1))
          endif
          if (abs(dt/dr*w1(5,i)+w2(55,i)).ge.2.d0*eps1) then
            w2(65,i) = abs(dt/dr*w1(5,i)+w2(55,i))*w2(25,i)
     +                 -(w2(45,i)+w2(45,i+1))
          else
            w2(65,i) = ((dt/dr*w1(5,i)+w2(55,i))**2/4.d0/eps1+eps1)
     +                 *w2(25,i)-(w2(45,i)+w2(45,i+1))
          endif
 111  continue
c-------------------------------------------------------------------------
c  -f(i+1/2)
c-------------------------------------------------------------------------
       do 112 i=0,nw
          w2(71,i) = (w0(1,i)*w2(1,i)+w0(1,i+1)*w2(1,i+1))/2.d0
          w2(72,i) = (w0(2,i)*w2(1,i)+w2(4,i)
     +               +w0(2,i+1)*w2(1,i+1)+w2(4,i+1))/2.d0
          w2(73,i) = (w0(3,i)*w2(1,i)+w0(3,i+1)*w2(1,i+1))/2.d0
          w2(74,i) = (w0(4,i)*w2(1,i)+w0(4,i+1)*w2(1,i+1))/2.d0
          w2(75,i) = (w0(5,i)*w2(1,i)+w2(4,i)*w2(1,i)
     +               +w0(5,i+1)*w2(1,i+1)+w2(4,i+1)*w2(1,i+1))/2.d0
c
          w2(71,i) = w2(71,i)-dr/dt*(w2(61,i)*w1(11,i)
     +                              +w2(62,i)*w1(21,i)
     +                              +w2(63,i)*w1(31,i)
     +                              +w2(64,i)*w1(41,i)
     +                              +w2(65,i)*w1(51,i))/2.d0
          w2(72,i) = w2(72,i)-dr/dt*(w2(61,i)*w1(12,i)
     +                              +w2(62,i)*w1(22,i)
     +                              +w2(63,i)*w1(32,i)
     +                              +w2(64,i)*w1(42,i)
     +                              +w2(65,i)*w1(52,i))/2.d0
          w2(73,i) = w2(73,i)-dr/dt*(w2(61,i)*w1(13,i)
     +                              +w2(62,i)*w1(23,i)
```

```
     +                              +w2(63,i)*w1(33,i)
     +                              +w2(64,i)*w1(43,i)
     +                              +w2(65,i)*w1(53,i))/2.d0
         w2(74,i) = w2(74,i)-dr/dt*(w2(61,i)*w1(14,i)
     +                              +w2(62,i)*w1(24,i)
     +                              +w2(63,i)*w1(34,i)
     +                              +w2(64,i)*w1(44,i)
     +                              +w2(65,i)*w1(54,i))/2.d0
         w2(75,i) = w2(75,i)-dr/dt*(w2(61,i)*w1(15,i)
     +                              +w2(62,i)*w1(25,i)
     +                              +w2(63,i)*w1(35,i)
     +                              +w2(64,i)*w1(45,i)
     +                              +w2(65,i)*w1(55,i))/2.d0
 112  continue
c-----------------------------------------------------------------------
c  q(i)
c-----------------------------------------------------------------------
      do 113 i=1,nw
         w0(1,i) = w0(1,i)-dt/dr*(w2(71,i)-w2(71,i-1))
         w0(2,i) = w0(2,i)-dt/dr*(w2(72,i)-w2(72,i-1))
         w0(3,i) = w0(3,i)-dt/dr*(w2(73,i)-w2(73,i-1))
         w0(4,i) = w0(4,i)-dt/dr*(w2(74,i)-w2(74,i-1))
         w0(5,i) = w0(5,i)-dt/dr*(w2(75,i)-w2(75,i-1))
 113  continue
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
c     return
c     end
c-----------------------------------------------------------------------
c  ak(i+1/2)
c-----------------------------------------------------------------------
      do 204 i=-1,nw+1
         w1(1,i) = w2(11,i)
         w1(2,i) = w2(11,i)
 204  continue
c-----------------------------------------------------------------------
c  Rk(i+1/2)
c-----------------------------------------------------------------------
      do 205 i=-1,nw+1
         w1(11,i) = 1.d0
         w1(12,i) = 1.d0
 205  continue
c-----------------------------------------------------------------------
c  Lk(i+1/2)
```

```
c-----------------------------------------------------------------------
      do 206 i=-1,nw+1
         w1(61,i) = 1.d0
         w1(62,i) = 1.d0
 206  continue
c-----------------------------------------------------------------------
c  alphak(i+1/2)
c-----------------------------------------------------------------------
      do 207 i=-1,nw+1
         w2(21,i) = w1(61,i)*(w5(1,i+1)-w5(1,i))
         w2(22,i) = w1(62,i)*(w5(2,i+1)-w5(2,i))
 207  continue
c-----------------------------------------------------------------------
c  ~gk(i+1/2)
c-----------------------------------------------------------------------
      do 208 i=-1,nw+1
         if (dt/dr*abs(w1(1,i)).ge.2.d0*eps1) then
            w2(31,i) = (dt/dr*abs(w1(1,i))-(dt/dr*w1(1,i))**2)
     +                 /2.d0*w2(21,i)
         else
            w2(31,i) = ((dt/dr*w1(1,i))**2/4.d0/eps1+eps1
     +                 -(dt/dr*w1(1,i))**2)/2.d0*w2(21,i)
         endif
         if (dt/dr*abs(w1(2,i)).ge.2.d0*eps1) then
            w2(32,i) = (dt/dr*abs(w1(2,i))-(dt/dr*w1(2,i))**2)
     +                 /2.d0*w2(22,i)
         else
            w2(32,i) = ((dt/dr*w1(2,i))**2/4.d0/eps1+eps1
     +                 -(dt/dr*w1(2,i))**2)/2.d0*w2(22,i)
         endif
 208  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
c     do 209 i=0,nw+1
c        w2(41,i) = sign(1.d0,w2(31,i))
c    +              *max(0.d0,min(abs(w2(31,i)),
c    +                        sign(1.d0,w2(31,i))*w2(31,i-1)))
c        w2(42,i) = sign(1.d0,w2(32,i))
c    +              *max(0.d0,min(abs(w2(32,i)),
c    +                        sign(1.d0,w2(32,i))*w2(32,i-1)))
c209  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
```

```
c     do 209 i=0,nw+1
c        w2(41,i) = sign(1.d0,w2(31,i))
c    +              *max(0.d0,min((abs(w2(31,i))
c    +                        +sign(1.d0,w2(31,i))*w2(31,i-1))/2.d0,
c    +                        2.d0*abs(w2(31,i)),
c    +                        2.d0*sign(1.d0,w2(31,i))*w2(31,i-1)))
c        w2(42,i) = sign(1.d0,w2(32,i))
c    +              *max(0.d0,min((abs(w2(32,i))
c    +                        +sign(1.d0,w2(32,i))*w2(32,i-1))/2.d0,
c    +                        2.d0*abs(w2(32,i)),
c    +                        2.d0*sign(1.d0,w2(32,i))*w2(32,i-1)))
c209  continue
c-----------------------------------------------------------------------
c  gk(i)
c-----------------------------------------------------------------------
      do 209 i=0,nw+1
         w2(41,i) = sign(1.d0,w2(31,i))
     +              *max(0.d0,min(abs(w2(31,i)),
     +                        2.d0*sign(1.d0,w2(31,i))*w2(31,i-1)),
     +                        min(2.d0*abs(w2(31,i)),
     +                        sign(1.d0,w2(31,i))*w2(31,i-1)))
         w2(42,i) = sign(1.d0,w2(32,i))
     +              *max(0.d0,min(abs(w2(32,i)),
     +                        2.d0*sign(1.d0,w2(32,i))*w2(32,i-1)),
     +                        min(2.d0*abs(w2(32,i)),
     +                        sign(1.d0,w2(32,i))*w2(32,i-1)))
 209  continue
c-----------------------------------------------------------------------
c  gammak(i+1/2)
c-----------------------------------------------------------------------
      do 210 i=0,nw
         if (abs(w2(21,i)).ge.1.d-30) then
            w2(51,i) = (w2(41,i+1)-w2(41,i))/w2(21,i)
         else
            w2(51,i) = 0.d0
         endif
         if (abs(w2(22,i)).ge.1.d-30) then
            w2(52,i) = (w2(42,i+1)-w2(42,i))/w2(22,i)
         else
            w2(52,i) = 0.d0
         endif
 210  continue
c-----------------------------------------------------------------------
c  betak(i+1/2)
c-----------------------------------------------------------------------
```

```
      do 211 i=0,nw
         if (abs(dt/dr*w1(1,i)+w2(51,i)).ge.2.d0*eps1) then
            w2(61,i) = abs(dt/dr*w1(1,i)+w2(51,i))*w2(21,i)
     +                 -(w2(41,i)+w2(41,i+1))
         else
            w2(61,i) = ((dt/dr*w1(1,i)+w2(51,i))**2/4.d0/eps1+eps1)
     +                 *w2(21,i)-(w2(41,i)+w2(41,i+1))
         endif
         if (abs(dt/dr*w1(2,i)+w2(52,i)).ge.2.d0*eps1) then
            w2(62,i) = abs(dt/dr*w1(2,i)+w2(52,i))*w2(22,i)
     +                 -(w2(42,i)+w2(42,i+1))
         else
            w2(62,i) = ((dt/dr*w1(2,i)+w2(52,i))**2/4.d0/eps1+eps1)
     +                 *w2(22,i)-(w2(42,i)+w2(42,i+1))
         endif
 211  continue
c-----------------------------------------------------------------------
c  -f(i+1/2)
c-----------------------------------------------------------------------
      do 212 i=0,nw
         w2(71,i) = (w5(1,i)*w2(1,i)+w5(1,i+1)*w2(1,i+1))/2.d0
         w2(72,i) = (w5(2,i)*w2(1,i)+w5(2,i+1)*w2(1,i+1))/2.d0
c
         w2(71,i) = w2(71,i)-dr/dt*w2(61,i)*w1(11,i)/2.d0
         w2(72,i) = w2(72,i)-dr/dt*w2(62,i)*w1(12,i)/2.d0
 212  continue
c-----------------------------------------------------------------------
c  q(i)
c-----------------------------------------------------------------------
      do 213 i=1,nw
         w5(1,i) = w5(1,i)-dt/dr*(w2(71,i)-w2(71,i-1))
         w5(2,i) = w5(2,i)-dt/dr*(w2(72,i)-w2(72,i-1))
 213  continue
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
CC    program lorenz
      SUBROUTINE LORENZ
c
c  Written by Dongsu Ryu, February 2005
c  Compile on linux with ifort -O2 -132
c
      implicit none
C
      INCLUDE "mpif.h"
      INCLUDE "common"
c
CC    real*16 gam, rho, vel, pre
CC    real*16 ddd, mmm, eee
CC    real*16 vt, vl1, vl2
CC    real*16 coff1, coff2, coff3, coff4, coff5
CC    complex*32 a, b, c, d
CC    complex*32 vs
CC    real*16 vsr, vsi, v
      INTEGER I
      REAL*8 DDD, MMM, EEE
      REAL*8 VT, VL1, VL2
      REAL*8 COFF1, COFF2, COFF3, COFF4, COFF5
      COMPLEX*16 A, B, C, D
      COMPLEX*16 VS
      REAL*8 VSR, VSI, V
c
c     write (*,*) 'gam, rho, vel, pre'
c     read (*,*) gam, rho, vel, pre
c     write (*,*)
c
c     ddd = rho/sqrt(1.d0-vel**2)
c     mmm = (rho+gam/(gam-1.d0)*pre)*vel/(1.d0-vel**2)
c     eee = (rho+gam/(gam-1.d0)*pre)/(1.d0-vel**2)-pre
c     write (*,*) 'ddd, mmm, eee'
c     write (*,*) ddd, mmm, eee
c     write (*,*)
c
CC    write (*,*) 'gam, ddd, mmm, eee'
CC    read (*,*) gam, ddd, mmm, eee
CC    write (*,*)
c
      DO 101 I=1,NW
C
      IF (W0(1,I).LE.1.D-30) THEN
         W0(1,I) = SQRT(1.D0-(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2)
```

```
     +             /W0(5,I)**2)*W0(5,I)/2.D0
      ENDIF
C
      DDD = W0(1,I)
      MMM = SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2)
      EEE = W0(5,I)
C
CC    if ((mmm/abs(ddd)).le.(1.d-10)) then
      IF ((MMM/DDD).LE.(1.D-10)) THEN
         v = 0.d0
CC       write (*,*) 'mmm=0: approximate solution'
CC       write (*,*) v
CC       stop
         GOTO 900
      endif
c
      if ((mmm/eee).ge.(1.d0-1.d-6)) then
CC       v = sqrt(1.d2-1.d0)/1.d1
         V = 1.D0-1.D-6
CC       write (*,*) 'mmm>=eee: approximate solution'
CC       write (*,*) v
CC       stop
         GOTO 900
      endif
c
      vt = (gam*eee)**2-4.d0*(gam-1.d0)*mmm**2
      if (vt.le.(0.d0)) then
         vl1 = 0.d0
      else
         vl1 = (gam*eee-sqrt(vt))/2.d0/(gam-1.d0)/mmm
      endif
      vl2 = mmm/eee
CC    write (*,*) 'limits'
CC    write (*,*) vl1, vl2
CC    write (*,*)
c
CC    if ((mmm/abs(ddd)).le.(1.d-6)) then
      IF ((MMM/DDD).LE.(1.D-6)) THEN
         v = (vl1+vl2)/2.d0
CC       write (*,*) 'very small mmm: approximate solution'
CC       write (*,*) v
CC       stop
         GOTO 900
      endif
c
```

```
      coff1 = mmm**2
      coff2 = -2.d0*eee*gam*mmm
      coff3 = 2.d0*(gam-1.d0)*mmm**2+eee**2*gam**2-ddd**2*(gam-1.d0)**2
      coff4 = -2.d0*eee*gam*(gam-1.d0)*mmm
      coff5 = mmm**2*(gam-1.d0)**2+ddd**2*(gam-1.d0)**2
c
      a = coff1/coff5
      b = coff2/coff5
      c = coff3/coff5
      d = coff4/coff5
c
      vs = ((SQRT(27.d0*a**2*d**4+(4.d0*b**3-18.d0*a*b*c)*d**3
     +     +(4.d0*a*c**3-b**2*c**2
     1   -144.d0*a**2*c+6.d0*a*b**2)*d**2+(80.d0*a*b*c**2
     +     -18.d0*b**3*c+192.d0*a**2*b)*d
     2   -16.d0*a*c**4+4.d0*b**2*c**3+128.d0*a**2*c**2
     +     -144.d0*a*b**2*c+27.d0*b**4-256.d0*a
     3   **3)/SQRT(3.d0)/6.d0-(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*d
     +     -2.d0*c**3-27.d0*b**2)/54.d0
     4   )**((-1.d0)/3.d0)*(36.d0*(SQRT(27.d0*a**2*d**4+(4.d0*b**3
     +     -18.d0*a*b*c)*d**3+(
     5   4.d0*a*c**3-b**2*c**2-144.d0*a**2*c+6.d0*a*b**2)*d**2
     +     +(80.d0*a*b*c**2-18.d0*b*
     6   *3*c+192.d0*a**2*b)*d-16.d0*a*c**4+4.d0*b**2*c**3
     +     +128.d0*a**2*c**2-144.d0*a*b*
     7   *2*c+27.d0*b**4-256.d0*a**3)/SQRT(3.d0)/6.d0-(a*(72.d0*c
     +     -27.d0*d**2)+9.d0*b*c*d-2.d0*
     8   c**3-27.d0*b**2)/54.d0)**(2.d0/3.d0)+(9.d0*d**2
     +     -24.d0*c)*(SQRT(27.d0*a**2*d**4
     9   +(4.d0*b**3-18.d0*a*b*c)*d**3+(4.d0*a*c**3-b**2*c**2
     +     -144.d0*a**2*c+6.d0*a*b**2
     :   )*d**2+(80.d0*a*b*c**2-18.d0*b**3*c+192.d0*a**2*b)*d
     +     -16.d0*a*c**4+4.d0*b**2*c*
     ;   *3+128.d0*a**2*c**2-144.d0*a*b**2*c+27.d0*b**4-256.d0*a**3)
     +     /SQRT(3.d0)/6.d0-(a*
     <   (72.d0*c-27.d0*d**2)+9.d0*b*c*d-2.d0*c**3-27.d0*b**2)
     +     /54.d0)**(1.d0/3.d0)-12.d0*b*d+
     =   4.d0*c**2+48.d0*a))**((-1.d0)/4.d0)*SQRT(-(SQRT(27.d0*a**2
     +     *d**4+(4.d0*b**3-18.d0
     >   *a*b*c)*d**3+(4.d0*a*c**3-b**2*c**2-144.d0*a**2*c
     +     +6.d0*a*b**2)*d**2+(80.d0*
     ?   a*b*c**2-18.d0*b**3*c+192.d0*a**2*b)*d-16.d0*a*c**4
     +     +4.d0*b**2*c**3+128.d0*a**2
     @   *c**2-144.d0*a*b**2*c+27.d0*b**4-256.d0*a**3)/SQRT(3.d0)
     +     /6.d0-(a*(72.d0*c-27.d0*d*
```

```
1    *2)+9.d0*b*c*d-2.d0*c**3-27.d0*b**2)/54.d0)**((-1.d0)/3.d0)
+      *((18.d0*(SQRT(27.d0*a
2    **2*d**4+(4.d0*b**3-18.d0*a*b*c)*d**3+(4.d0*a*c**3
+      -b**2*c**2-144.d0*a**2*c+
3    6.d0*a*b**2)*d**2+(80.d0*a*b*c**2-18.d0*b**3*c
+      +192.d0*a**2*b)*d-16.d0*a*c**4+4.d0
4    *b**2*c**3+128.d0*a**2*c**2-144.d0*a*b**2*c+27.d0*b**4
+      -256.d0*a**3)/SQRT(3.d0)
5    /6.d0-(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*d-2.d0*c**3
+      -27.d0*b**2)/54.d0)**(2.d0/3.d0)
6    +(24.d0*c-9.d0*d**2)*(SQRT(27.d0*a**2*d**4+(4.d0*b**3
+      -18.d0*a*b*c)*d**3+(4.d0*a*c
7    **3-b**2*c**2-144.d0*a**2*c+6.d0*a*b**2)*d**2+(80.d0*a*b*c**2
+      -18.d0*b**3*c+
8    192.d0*a**2*b)*d-16.d0*a*c**4+4.d0*b**2*c**3+128.d0*a**2*c**2
+      -144.d0*a*b**2*c+
9    27.d0*b**4-256.d0*a**3)/SQRT(3.d0)/6.d0-(a*(72.d0*c
+      -27.d0*d**2)+9.d0*b*c*d-2.d0*c**3-
:    27.d0*b**2)/54.d0)**(1.d0/3.d0)-6.d0*b*d+2.d0*c**2+24.d0*a)
+      *SQRT((SQRT(27.d0*a**2
;    *d**4+(4.d0*b**3-18.d0*a*b*c)*d**3+(4.d0*a*c**3-b**2*c**2
+      -144.d0*a**2*c+6.d0*a
<    *b**2)*d**2+(80.d0*a*b*c**2-18.d0*b**3*c+192.d0*a**2*b)*d
+      -16.d0*a*c**4+4.d0*b*
=    *2*c**3+128.d0*a**2*c**2-144.d0*a*b**2*c+27.d0*b**4
+      -256.d0*a**3)/SQRT(3.d0)/6
>    .d0-(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*d-2.d0*c**3-27.d0*b**2)
+      /54.d0)**((-1.d0)/3.d0)
?    *(36.d0*(SQRT(27.d0*a**2*d**4+(4.d0*b**3-18.d0*a*b*c)*d**3
+      +(4.d0*a*c**3-b**2*c
@    **2-144.d0*a**2*c+6.d0*a*b**2)*d**2+(80.d0*a*b*c**2
+      -18.d0*b**3*c+192.d0*a**2*b
1    )*d-16.d0*a*c**4+4.d0*b**2*c**3+128.d0*a**2*c**2-144.d0
+      *a*b**2*c+27.d0*b**4-25
2    6.d0*a**3)/SQRT(3.d0)/6.d0-(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*d
+      -2.d0*c**3-27.d0*b**2)/5
3    4.d0)**(2.d0/3.d0)+(9.d0*d**2-24.d0*c)*(SQRT(27.d0*a**2*d**4
+      +(4.d0*b**3-18.d0*a*b
4    *c)*d**3+(4.d0*a*c**3-b**2*c**2-144.d0*a**2*c
+      +6.d0*a*b**2)*d**2+(80.d0*a*b*
5    c**2-18.d0*b**3*c+192.d0*a**2*b)*d-16.d0*a*c**4+4.d0*b**2*c**3
+      +128.d0*a**2*c**
6    2-144.d0*a*b**2*c+27.d0*b**4-256.d0*a**3)/SQRT(3.d0)/6.d0
+      -(a*(72.d0*c-27.d0*d**2)+
7    9.d0*b*c*d-2.d0*c**3-27.d0*b**2)/54.d0)**(1.d0/3.d0)
```

```
     +     -12.d0*b*d+4.d0*c**2+48.d0*a))+(
     8   -27.d0*d**3+108.d0*c*d-216.d0*b)*(SQRT(27.d0*a**2*d**4
     +     +(4.d0*b**3-18.d0*a*b*c)*d*
     9   *3+(4.d0*a*c**3-b**2*c**2-144.d0*a**2*c+6.d0*a*b**2)*d**2
     +     +(80.d0*a*b*c**2-1
     :   8.d0*b**3*c+192.d0*a**2*b)*d-16.d0*a*c**4+4.d0*b**2*c**3
     +     +128.d0*a**2*c**2-144.d0*
     ;   a*b**2*c+27.d0*b**4-256.d0*a**3)/SQRT(3.d0)/6.d0
     +     -(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*
     <   d-2.d0*c**3-27.d0*b**2)/54.d0)**(1.d0/3.d0)))
     +     /SQRT(2.d0)/6.d0-SQRT((SQRT(27.d0*
     =   a**2*d**4+(4.d0*b**3-18.d0*a*b*c)*d**3+(4.d0*a*c**3
     +     -b**2*c**2-144.d0*a**2*c
     >   +6.d0*a*b**2)*d**2+(80.d0*a*b*c**2-18.d0*b**3*c
     +     +192.d0*a**2*b)*d-16.d0*a*c**4+
     ?   4.d0*b**2*c**3+128.d0*a**2*c**2-144.d0*a*b**2*c+27.d0*b**4
     +     -256.d0*a**3)/SQRT(3.d0
     @   )/6.d0-(a*(72.d0*c-27.d0*d**2)+9.d0*b*c*d-2.d0*c**3
     +     -27.d0*b**2)/54.d0)**((-1.d0)/
     1   3.d0)*(36.d0*(SQRT(27.d0*a**2*d**4+(4.d0*b**3
     +     -18.d0*a*b*c)*d**3+(4.d0*a*c**3-b*
     2   *2*c**2-144.d0*a**2*c+6.d0*a*b**2)*d**2+(80.d0*a*b*c**2
     +     -18.d0*b**3*c+192.d0*a*
     3   *2*b)*d-16.d0*a*c**4+4.d0*b**2*c**3+128.d0*a**2*c**2
     +     -144.d0*a*b**2*c+27.d0*b**
     4   4-256.d0*a**3)/SQRT(3.d0)/6.d0-(a*(72.d0*c-27.d0*d**2)
     +     +9.d0*b*c*d-2.d0*c**3-27.d0*b**
     5   2)/54.d0)**(2.d0/3.d0)+(9.d0*d**2-24.d0*c)
     +     *(SQRT(27.d0*a**2*d**4+(4.d0*b**3-18.d0
     6   *a*b*c)*d**3+(4.d0*a*c**3-b**2*c**2-144.d0*a**2*c
     +     +6.d0*a*b**2)*d**2+(80.d0*
     7   a*b*c**2-18.d0*b**3*c+192.d0*a**2*b)*d-16.d0*a*c**4
     +     +4.d0*b**2*c**3+128.d0*a**2
     8   *c**2-144.d0*a*b**2*c+27.d0*b**4-256.d0*a**3)/SQRT(3.d0)
     +     /6.d0-(a*(72.d0*c-27.d0*d*
     9   *2)+9.d0*b*c*d-2.d0*c**3-27.d0*b**2)/54.d0)**(1.d0/3.d0)
     +     -12.d0*b*d+4.d0*c**2+48.d0*a
     :   ))/12.d0-d/4.d0
c
CC    write (*,*) 'v from transformation equation'
CC    write (*,*) vs
CC    write (*,*)
c
CC    vsr = real(vs)
      VSR = DBLE(VS)
```

```
      vsi = imag(vs)
c
CC    if (vsr.le.(1.d-8)) then
      IF (VSR.LE.(1.D-10)) THEN
         v = 0.d0
CC       write (*,*) 'zero: approximate solution'
CC       write (*,*) v
CC       stop
         GOTO 900
      endif
c
      if (abs(vsi/vsr).le.(1.d-6)) then
         if ((vsr.ge.vl1).and.(vsr.le.vl2)) then
            v = vsr
CC          v = min(v,(sqrt(1.d2-1.d0)/1.d1))
            V = MIN(V,1.D0-1.D-6)
CC          write (*,*) 'well-behaving: solution'
CC          write (*,*) v
         else
            if (abs(vsr-vl1).lt.abs(vsr-vl2)) then
CC             v = vl1+1.d-4
               V = VL1+1.D-6
               if (v.ge.vl2) v = (vl1+vl2)/2.d0
CC             v = min(v,(sqrt(1.d2-1.d0)/1.d1))
               V = MIN(V,1.D0-1.D-6)
CC             write (*,*) 'outside limits: approximate solution'
CC             write (*,*) v
            else
CC             v = vl2-1.d-4
               V = VL2-1.D-6
               if (v.le.vl1) v = (vl1+vl2)/2.d0
CC             v = min(v,(sqrt(1.d2-1.d0)/1.d1))
               V = MIN(V,1.D0-1.D-6)
CC             write (*,*) 'outside limits: approximate solution'
CC             write (*,*) v
            endif
         endif
      else
         if ((vsr.ge.vl1).and.(vsr.le.vl2)) then
            v = vsr
CC          v = min(v,(sqrt(1.d2-1.d0)/1.d1))
            V = MIN(V,1.D0-1.D-6)
CC          write (*,*) 'complex: approximate solution'
CC          write (*,*) v
         else
```

```
            v = (vl1+vl2)/2.d0
CC          v = min(v,(sqrt(1.d2-1.d0)/1.d1))
            V = MIN(V,1.D0-1.D-6)
CC          write (*,*) 'complex: approximate solution'
CC          write (*,*) v
         endif
      endif
C
 900  CONTINUE
      W3(3,I) = V
C
 101  CONTINUE
C
      DO 103 I=1,NW
            W4(1,I) = W0(1,I)*SQRT(1.D0-W3(3,I)**2)
         IF (SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2).GE.1.D-30) THEN
            W4(2,I) = W0(2,I)*W3(3,I)
     +                /SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2)
         ELSE
            W4(2,I) = 0.D0
         ENDIF
         IF (SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2).GE.1.D-30) THEN
            W4(3,I) = W0(3,I)*W3(3,I)
     +                /SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2)
         ELSE
            W4(3,I) = 0.D0
         ENDIF
         IF (SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2).GE.1.D-30) THEN
            W4(4,I) = W0(4,I)*W3(3,I)
     +                /SQRT(W0(2,I)**2+W0(3,I)**2+W0(4,I)**2)
         ELSE
            W4(4,I) = 0.D0
         ENDIF
            W4(5,I) = W0(5,I)-W0(2,I)*W4(2,I)
     +                       -W0(3,I)*W4(3,I)
     +                       -W0(4,I)*W4(4,I)
 103  CONTINUE
c
CC    stop
      RETURN
      end
```

```
      subroutine prot
c-----------------------------------------------------------------------
c  protection against density and pressure
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz
      real*8 den, pre, denmin, premin
c
      denmin = 1.d-4
      premin = 1.d-8
c-----------------------------------------------------------------------
c  minimum density and pressure
c-----------------------------------------------------------------------
      do 101 iz=1,nz
      do 101 iy=1,ny
      do 101 ix=1,nxp
         den = u1(ix,iy,iz)
         u1(ix,iy,iz) = max(den,denmin)
         pre = (gam-1.d0)*(u5(ix,iy,iz)-u1(ix,iy,iz))
         pre = max(pre,premin)
         u5(ix,iy,iz) = pre/(gam-1.d0)+u1(ix,iy,iz)
 101  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```

```
      subroutine dump
c-----------------------------------------------------------------------
c  data dump
c-----------------------------------------------------------------------
      implicit none
c
      include "mpif.h"
      include "common"
c
      integer ix, iy, iz
      real*8 den(nxp,ny,nz), pre(nxp,ny,nz), vel(nxp,ny,nz)
      real*8 fb(nxp,ny,nz), fc(nxp,ny,nz)
      real*8 sum0, sum1, sum2, sum3, sum0p, sum1p, sum2p, sum3p
      character*9 fname
c-----------------------------------------------------------------------
c  dump file
c-----------------------------------------------------------------------
      write (fname,900) 'jt', nunit, pi
 900  format (a2,i4.4,i2.2)
c
      open (unit=90,file=fname,status='unknown',form='unformatted')
c-----------------------------------------------------------------------
c  data dump
c-----------------------------------------------------------------------
      do 101 iz=1,nz
      do 101 iy=1,ny
      do 101 ix=1,nxp
         den(ix,iy,iz) = u1(ix,iy,iz)
         pre(ix,iy,iz) = (gam-1.d0)*(u5(ix,iy,iz)-u1(ix,iy,iz))
         vel(ix,iy,iz) = sqrt(u2(ix,iy,iz)**2
     +                        +u3(ix,iy,iz)**2+u4(ix,iy,iz)**2)
c
         fb(ix,iy,iz) = f1(ix,iy,iz)/q1(ix,iy,iz)
         fc(ix,iy,iz) = f2(ix,iy,iz)/q1(ix,iy,iz)
 101  continue
c
      write (90) den
      write (90) pre
      write (90) vel
c
      write (90) fb
      write (90) fc
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
```

```
c  end
c-----------------------------------------------------------------------
c     return
c     end
c-----------------------------------------------------------------------
c  time, effective radius, mean velocity, thermal energy
c-----------------------------------------------------------------------
      sum0p = 0.d0
c
      do 105 iz=1,nz
      do 105 iy=1,ny
      do 105 ix=1,nxp
         sum0p = sum0p + den(ix,iy,iz)*fc(ix,iy,iz)*dx**3
 105  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call MPI_REDUCE (sum0p,sum0,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_BCAST (sum0,1,MPI_DOUBLE_PRECISION,master,
     +                MPI_COMM_WORLD,err)
c
c     if (pi.eq.master) then
c        write (50,*) t, sum0
c     endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      sum1p = 0.d0
      sum2p = 0.d0
      sum3p = 0.d0
c
      do 115 iz=1,nz
      do 115 iy=1,ny
      do 115 ix=1,nxp
         x = (dble(ix+nxp*pi)-0.5d0)*dx
         y = (dble(iy)-0.5d0)*dy
         z = (dble(iz)-0.5d0)*dz
c
         sum1p = sum1p + den(ix,iy,iz)*fc(ix,iy,iz)*(x-4.d0)**2
     +                   *dx**3/sum0
         sum2p = sum2p + den(ix,iy,iz)*fc(ix,iy,iz)*(y-3.5d0)**2
     +                   *dy**3/sum0
         sum3p = sum3p + den(ix,iy,iz)*fc(ix,iy,iz)*(z-4.d0)**2
     +                   *dz**3/sum0
```

```
 115  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call MPI_REDUCE (sum1p,sum1,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum2p,sum2,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum3p,sum3,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
c
      if (pi.eq.master) then
         write (51,*) t, sqrt(sum1), sqrt(sum2), sqrt(sum3)
      endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      sum1p = 0.d0
      sum2p = 0.d0
      sum3p = 0.d0
c
      do 125 iz=1,nz
      do 125 iy=1,ny
      do 125 ix=1,nxp
         sum1p = sum1p + den(ix,iy,iz)*fc(ix,iy,iz)*u2(ix,iy,iz)
     +                   *dx**3/sum0
         sum2p = sum2p + den(ix,iy,iz)*fc(ix,iy,iz)*u3(ix,iy,iz)
     +                   *dy**3/sum0
         sum3p = sum3p + den(ix,iy,iz)*fc(ix,iy,iz)*u4(ix,iy,iz)
     +                   *dz**3/sum0
 125  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call MPI_REDUCE (sum1p,sum1,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum2p,sum2,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum3p,sum3,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
c
      if (pi.eq.master) then
         write (52,*) t, sum1, sum2, sum3
      endif
c
```

```
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      sum1p = 0.d0
c
      do 135 iz=1,nz
      do 135 iy=1,ny
      do 135 ix=1,nxp
         sum1p = sum1p + den(ix,iy,iz)*fc(ix,iy,iz)*pre(ix,iy,iz)*3.d0
     +                   *dx**3/sum0
 135  continue
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call MPI_REDUCE (sum1p,sum1,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
c
      if (pi.eq.master) then
         write (53,*) t, sum1
      endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  time, intensity at 90 deg, 45 deg, 0 deg
c-----------------------------------------------------------------------
      sum1p = 0.d0
      sum2p = 0.d0
      sum3p = 0.d0
c
      do 107 iz=1,nz
      do 107 iy=1,ny
      do 107 ix=1,nxp
         if (fb(ix,iy,iz).ge.0.001) then
            sum1p = sum1p + pre(ix,iy,iz)**(3.75d0/2.d0)*dx**3
     +                      *(1.d0-vel(ix,iy,iz)**2)
c
            sum2p = sum2p + pre(ix,iy,iz)**(3.75d0/2.d0)*dx**3
     +                      *(1.d0-vel(ix,iy,iz)**2)
     +                      /(1.d0-vel(ix,iy,iz)/sqrt(2.d0))**2
c
            sum3p = sum3p + pre(ix,iy,iz)**(3.75d0/2.d0)*dx**3
     +                      *(1.d0-vel(ix,iy,iz)**2)
     +                      /(1.d0-vel(ix,iy,iz))**2
         endif
 107  continue
c
```

```
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c
      call MPI_REDUCE (sum1p,sum1,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum2p,sum2,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
      call MPI_REDUCE (sum3p,sum3,1,MPI_DOUBLE_PRECISION,MPI_SUM,master,
     +                 MPI_COMM_WORLD,err)
c
      if (pi.eq.master) then
         write (71,*) t, sum1, sum2, sum3
      endif
c
      call MPI_BARRIER (MPI_COMM_WORLD,err)
c-----------------------------------------------------------------------
c  end
c-----------------------------------------------------------------------
      return
      end
```