Contents

Chapter 1

Lecture

In this chapter, we learn how to solve hydrodynamical equations and magnetohydrodynamical equations. These equations are hyperbolic partial differential equations and describe the propagation of waves. The wave property is key in constructing a numerical scheme for solving hyperbolic partial differential equations. Thus, in the first half of this chapter, we learn how to solve a simple one-dimensional wave equation. The numerical method gives us a basis for the hydrodynamical and magnetohydrodynamical equations, which are multi-dimensional and nonlinear.

1.1 Finite Difference Method

Physical quantities such as density and velocity are described as a field, i.e., a function of time and space, in hydrodynamics and magnetohydrodynamics. These quantities are described by arrays having finite numbers of discrete elements in numerical computations. You may already be familiar with this concept, which is also used to solve ordinary differential equations. When we solve partial differential equations, the physical quantities are tensor, because time and space are independent variables.

We consider the simplest case, in which a physical quantity, f, is a function of time, t, and the distance, x. Suppose that we know the value of f at the points $(x,\ t) = (j\Delta x,\ n\Delta t)$ for any integers j and n, where Δt and Δx denote a very short time interval and a very short distance, respectively. Then the array,

$$f_{j,n} \equiv f\left(x = j\,\Delta x,\, t = n\,\Delta t\right), \tag{1.1}$$

can be regarded as a function of x and t in a practical sense. We call Δx and Δt the grid spacing and time step, respectively.

We can evaluate the partial derivative $\partial f/\partial x$ from the array $f_{j,n}$. However, the evaluation is not unique. Suppose that the function, f, is a solution of a partial differential equation. Several different expressions converge to $\partial f/\partial x$ in the limit of $\Delta x \to 0$. For simplicity, we assume that the function f is smooth and differentiable. Then, the following three expressions:

$$\frac{f_{j+1,n} - f_{j,n}}{\Delta x} = \frac{\partial f}{\partial x} + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Delta x + \mathcal{O}\left(\Delta x^2\right), \tag{1.2}$$

$$\frac{f_{j,n} - f_{j-1,n}}{\Delta x} = \frac{\partial f}{\partial x} - \frac{1}{2}\frac{\partial^2 f}{\partial x^2}\Delta x + \mathcal{O}\left(\Delta x^2\right), \tag{1.3}$$

$$\frac{f_{j+1,n} - f_{j-1,n}}{2\Delta x} = \frac{\partial f}{\partial x} + \frac{1}{6}\frac{\partial^3 f}{\partial x^3}\Delta x^2 + \mathcal{O}\left(\Delta x^3\right), \tag{1.4}$$

denote good approximations to $\partial f/\partial x$. Equations (1.2), (1.3), and (1.4), are referred to as the forward difference, backward difference, and central difference, respectively. The forward difference and backward difference are of first-order accuracy because the deviation from the differential is proportional to Δx. The central difference is of second-order accuracy because the deviation is proportional to Δx^2. One might expect that the central difference should always be superior to the forward difference and backward difference because the accuracy is of higher order. This is not true however, and, as shown later, we obtain a better numerical solution when a differential equation is approximated with either the forward difference or the backward difference. An example is presented in the next section.

1.2 Linear Wave Equation

We consider the simplest wave equation:

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial x} = 0\,, \tag{1.5}$$

where c is a constant and denotes the wave propagation speed. We approximate Equation (1.5) by replacing each term with a finite difference. It is a natural choice to approximate

$$\frac{\partial f}{\partial t} \rightarrow \frac{f_{j,n+1} - f_{j,n}}{\Delta t}\,, \tag{1.6}$$

and to express $\partial f/\partial x$ as a function of $f_{j+1,n}$, $f_{j,n}$, $f_{j-1,n}$. Then, we can express $f_{j,n+1}$ as an explicit function of $f_{j+1,n}$, $f_{j,n}$, $f_{j-1,n}$ and solve the initial value problem of Equation (1.5) easily. If we apply the forward difference to the spatial derivative, we have

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + c\frac{f_{j+1,n} - f_{j,n}}{\Delta x} = 0\,. \tag{1.7}$$

Equation (1.7) is of forward difference in space and in time. If we apply the backward difference and central difference to the spatial derivative, we have

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + c\frac{f_{j,n} - f_{j-1,n}}{\Delta x} = 0\,, \tag{1.8}$$

and

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + c\frac{f_{j+1,n} - f_{j-1,n}}{2\Delta x} = 0\,, \tag{1.9}$$

respectively. Although these three finite difference equations are quite similar in expression, their respective solutions are quite different from one another, as shown below.

1.3 Stability Analysis

We next compare the solutions of equations (1.7), (1.8), and (1.9), with the solution of equation (1.5). Suppose that the initial value at $t = 0$ is given by

$$f = \exp\left(\frac{ikx}{\Delta x}\right)\,, \tag{1.10}$$

where i denotes $\sqrt{-1}$. For later convenience, the function f is assumed to be complex. [1] Then, the initial value is expressed as

$$f_{j,0} \;=\; \exp\left(ikj\right), \tag{1.11}$$

in array form. The wave number is assumed to be in the range $0 \;<\; k \;\leq\; \pi$ without loss of generality. [2] The solution of the differential equation is

$$f \;=\; \exp\left[\frac{ik\left(x \;-\; ct\right)}{\Delta x}\right]. \tag{1.12}$$

Thus, ideally, the solution of the finite difference equation is expressed as

$$f_{j,n} \;=\; \exp\left[ik\left(j \;-\; \frac{cn\Delta t}{\Delta x}\right)\right]. \tag{1.13}$$

The solution of Equation (1.7) is

$$f_{j,n}^{(\mathrm{F})} \;=\; \left\{1 \;-\; \frac{c\Delta t}{\Delta x}\;\left[\exp(ik) \;-\; 1\right]\right\}^{n}\;\exp(ikj)\,. \tag{1.14}$$

Although some computations are required to derive the solution, it is easy to confirm the solution by substitution. Similarly, the solutions of (1.8) and (1.9) are

$$f_{j,n}^{(\mathrm{B})} \;=\; \left\{1 \;-\; \frac{c\Delta t}{\Delta x}\;\left[1 \;-\; \exp(-ik)\right]\right\}^{n}\;\exp(ikj)\,, \tag{1.15}$$

and

$$f_{j,n}^{(\mathrm{C})} \;=\; \left\{1 \;-\; i\,\frac{c\Delta t}{\Delta x}\;\sin k\right\}^{n}\;\exp(ikj)\,, \tag{1.16}$$

respectively. All of these solutions approach the exact solution given by Equation (1.13) in the limit of very small Δt, because the term in brackets is approximately $1 - ikc\Delta t/\Delta x$. The ratios between the approximate and ideal solutions are evaluated to be

$$f_{j,n}^{(\mathrm{F})} \;=\; \left[g^{(\mathrm{F})}\right]^{n}\;f_{j,n}\,, \tag{1.17}$$

$$f_{j,n}^{(\mathrm{B})} \;=\; \left[g^{(\mathrm{B})}\right]^{n}\;f_{j,n}\,, \tag{1.18}$$

$$f_{j,n}^{(\mathrm{C})} \;=\; \left[g^{(\mathrm{C})}\right]^{n}\;f_{j,n}\,, \tag{1.19}$$

where

$$g^{(\mathrm{F})} \;=\; \left\{1 \;-\; \frac{c\Delta t}{\Delta x}\;\left[\exp(ik) \;-\; 1\right]\right\}\;\exp\left(\frac{ick\Delta t}{\Delta x}\right), \tag{1.20}$$

$$g^{(\mathrm{B})} \;=\; \left\{1 \;-\; \frac{c\Delta t}{\Delta x}\;\left[1 \;-\; \exp(-ik)\right]\right\}\;\exp\left(\frac{ick\Delta t}{\Delta x}\right), \tag{1.21}$$

$$g^{(\mathrm{C})} \;=\; \left[1 \;-\; i\,\frac{c\Delta t}{\Delta x}\;\sin k\right]\;\exp\left(\frac{ick\Delta t}{\Delta x}\right). \tag{1.22}$$

[1] We assume that the reader is familiar with complex analysis. The same mathematical techniques are used in the analysis of electric circuits.

[2] Since j is an integer, the value of $f_{j,0}$ remains unchanged or changes only its sign when we replace k with $k' \;=\; k \;+\; n\pi$. The restriction, $0 \;<\; k \;\leq\; \pi$, does not cause any loss of generality.

Thus, the error due to the finite difference is estimated from the deviations of $g^{(\mathrm{F})}$, $g^{(\mathrm{B})}$, and $g^{(\mathrm{C})}$ from unity.

The error depends on c, k, Δt, and Δx. However, we can restrict ourselves to the case of $c > 0$ without any loss of generality. Note the relation, $g^{(F)}(-c) = g^{(B)}(c)^*$, where the asterisk denotes the complex conjugate. The error of the forward difference for negative c is evaluated from that of the backward difference. This is based on the fact that simultaneous inversion of the propagation speed and coordinate causes no practical change. The error does not depend on the sign of the propagation speed in the central difference because $g^{(C)}(-c) = g^{(C)}(c)$. Furthermore, the speed of sound appears only in the form of $c\Delta t/\Delta x$ in Equations (1.21), (1.22), and (1.22). The term, $c\Delta t/\Delta x$, is called the CFL number, where CFL stands for three mathematicians, Courant, Friedrich, and Levy. Thus, the error depends on two nondimensional numbers, $c\Delta t/\Delta x$ and k. When the wavenumber is very small ($0 < k \ll 1$), any g is very close to unity. Thus, we are interested in the case in which the wave number k is relatively large. Here, we study the dependence of the error on the CFL number for $k/\pi = 0.25$, 0.5, and 0.75.

As shown in Figure 1.1, the absolute value $|g^{(\mathrm{F})}|$ is larger than unity and increases monotonically with $c\Delta t/\Delta x$. This causes difficulty in that the wave amplitude increases with time in the forward difference. Even when $|g^{(\mathrm{F})}|$ is only slightly larger than unity, the wave amplitude increases exponentially with time because we repeat the manipulation.

Next, we examine the backward difference. Figure 1.2 shows the error of the numerical solution obtained by the backward difference. The absolute value of the ratio, $|g^{(\mathrm{B})}|$, does not exceed unity in the range $0 \leq c\Delta t/\Delta x \leq 1$. Accordingly, the amplitude does not grow unnaturally as long as the CFL number is smaller than or equal to unity. The backward difference gives the best solution among the three differences compared in this section, as shown later. The reader might wonder why this is the case. Why does the wave amplitude diminish in the backward difference solution, whereas it is constant in the exact solution? We shall examine the central difference before answering this question.

Figure 1.3 shows the error of the numerical solution obtained by the central difference. The ratio, $|g^{(\mathrm{C})}|$, is the closest to unity than $|g^{(\mathrm{F})}|$ and $|g^{(\mathrm{C})}|$ when the CFL number is small. However, the ratio is slightly larger than unity and the wave amplitude grows exponentially with time. Thus, the central difference does not provide a satisfactory solution.

The above analysis, which is referred to as von Neumann stability analysis, is applied to a sinusoidal wave. However, the result is valid for any initial value because any solution is expressed by superpositions of sinusoidal waves for a linear differential equation. [3] If the von Neumann stability analysis does not allow for the possibility of divergence for any sinusoidal wave, the amplitude of the wave never diverges, irrespective of the wave form. We shall confirm this for a rectangular wave.

Figure 1.4 shows the numerical solution of Equation (1.5) for $c = 1$. The solution is obtained by the forward difference. The initial value is given by

$$f = \begin{cases} 1 & (x < 0) \\ 0 & (x \geq 0) \end{cases}, \tag{1.23}$$

at $t = 0$. The time step and grid spacing are taken to be $\Delta t = 0.08$ and $\Delta x = 0.1$, respectively. The solid curve denotes the numerical solution at $t = 0.24$, while the dotted curve denotes the exact solution at $t = 0.24$. As expected from the von Neumann stability analysis, the numerical solution shows a large oscillation and is very different from the exact solution. Note that only three time steps spoil the solution completely.

[3] This is proven by Fourier analysis.

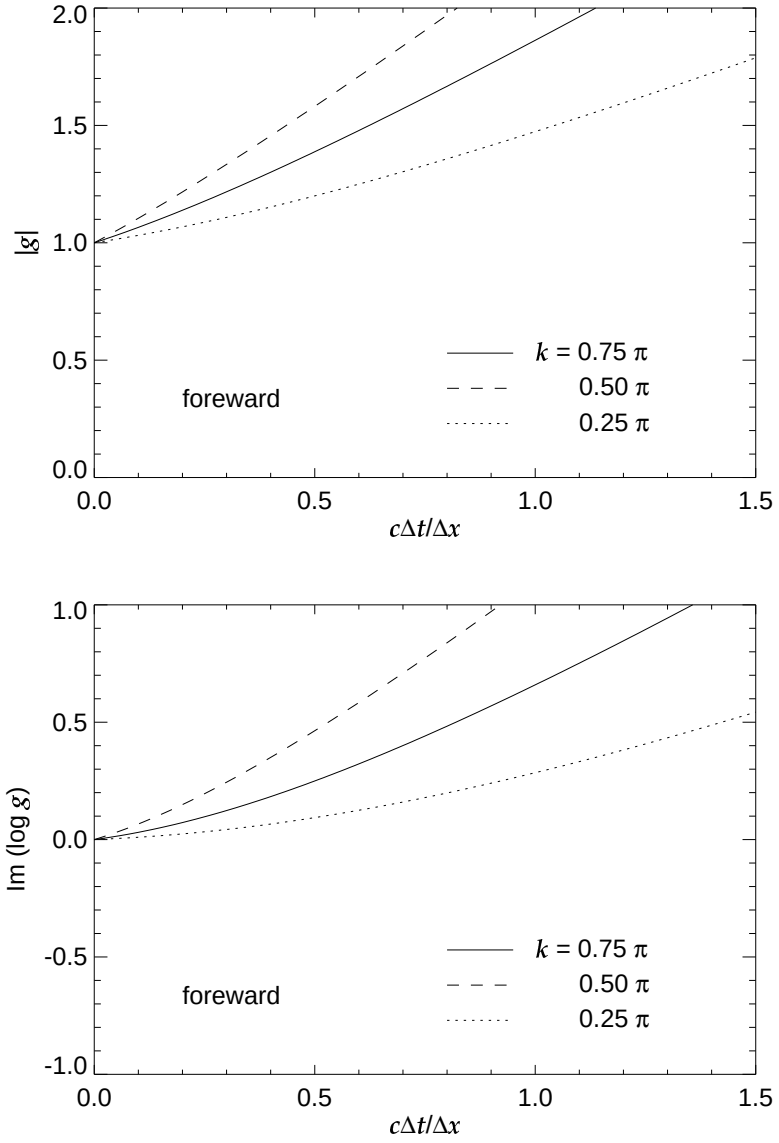

Figure 1.1: The ratio of the numerical solution obtained by the forward difference to the exact solution is shown as a function of the CFL number. The upper panel denotes the absolute value, $|g^{(\mathrm{F})}|$, while the lower denotes the phase difference, $\Im \log g^{(\mathrm{F})}$.

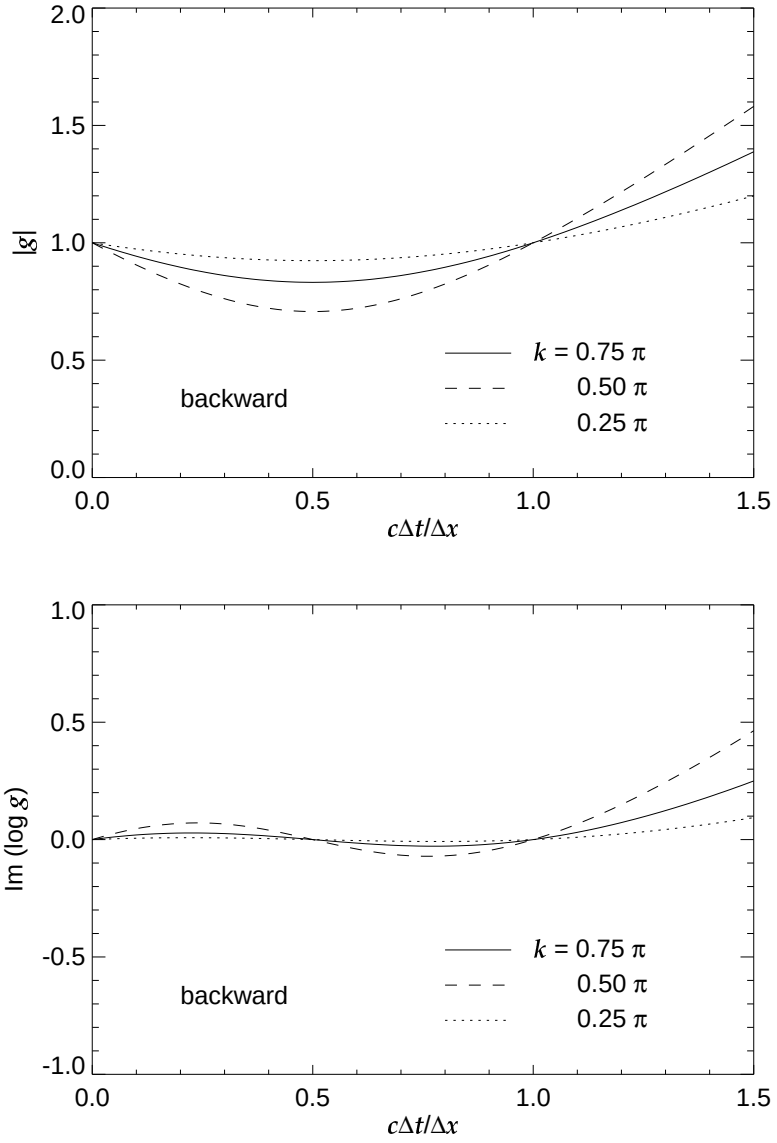

Figure 1.2: The ratio of the numerical solution obtained by the backward difference to the exact solution is shown as a function of the CFL number. The upper panel denotes the absolute value, $|g^{(\mathrm{F})}|$, while the lower denotes the phase difference, $\Im \log g^{(\mathrm{F})}$.

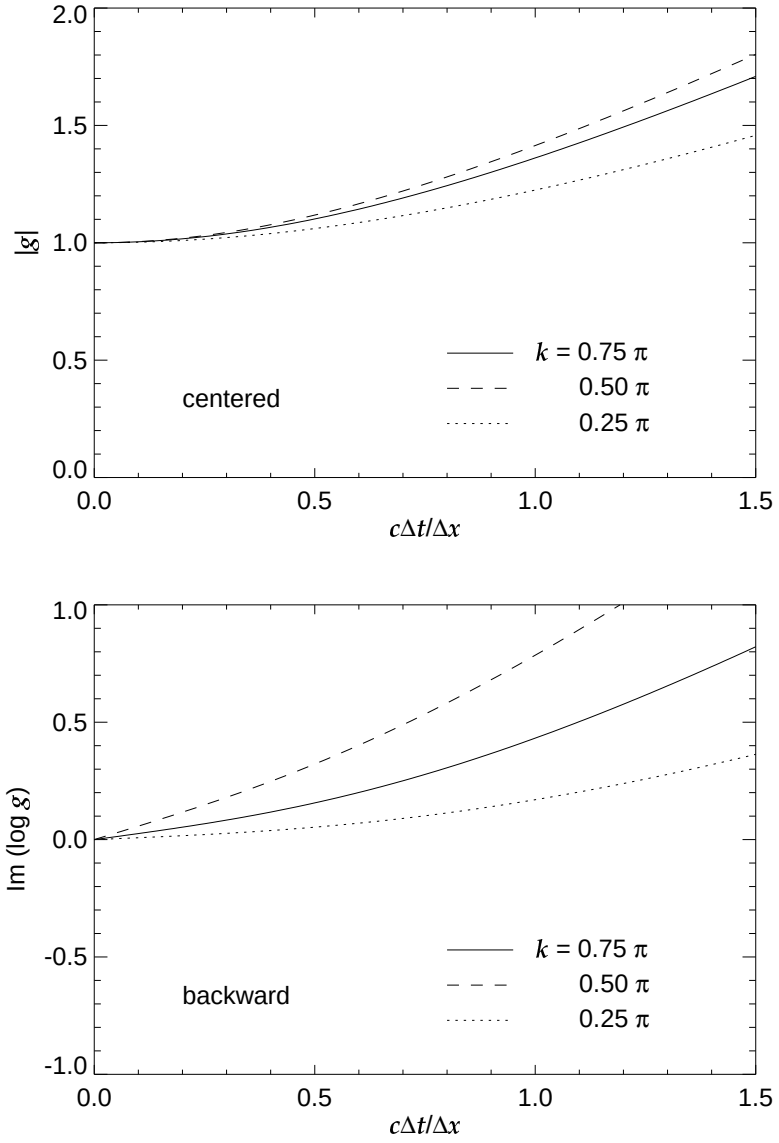

Figure 1.3: The ratio of the numerical solution obtained by the central difference to the exact solution is shown as a function of the CFL number. The upper panel denotes the absolute value, $|g^{(\mathrm{F})}|$, while the lower denotes the phase difference, $\Im \log g^{(\mathrm{F})}$.

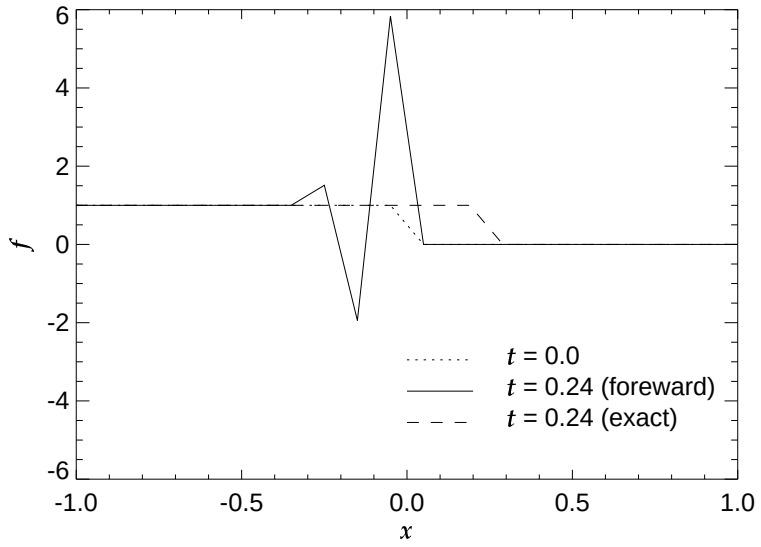

Figure 1.4: Numerical solution obtained by the forward difference. The dotted line denotes the initial condition, while the solid curve denotes the numerical solution for the stage forwarded three time steps. It is very different from the exact solution for the corresponding stage (dashed curve). The CFL number is taken to be 0.8.

The numerical solution shown in Figure 1.4 has another problem in that the wave does not seem to propagate to the right. This problem is also explained by the von Neumann stability analysis. The lower panel of Figure 1.1 shows that the phase error, $\Im \log g^{(F)}$, is large for the forward difference. The wave propagation can not be followed properly in the forward difference due to the large phase error.

Figure 1.5 shows the numerical solution of Equation (1.5) for $c = 1$. The solution is obtained by the backward difference. The initial condition, wave propagation speed, time step and grid spacing are the same as those for Figure 1.4. The backward difference gives a good approximation of the exact solution, although the numerical solution is appreciably round at the wave front.

Figure 1.6 shows the numerical solution of Equation (1.5) for $c = 1$. The solution is obtained by the central difference. The numerical solution is closer to that obtained by the forward difference than that obtained by the backward difference. Equations (1.7)-(1.9) imply that the central difference is something similar to the average of the forward difference and backward difference. The average of the good solution and the bad solution is similar to the bad solution, because oscillation grows exponentially in the bad solution.

Comparison of Figures 1.4, 1.5 and 1.6 show that only the backward difference provides an acceptable solution. Next, we examine whether the backward difference gives a good solution for a later stage. Figure 1.7 shows the solution obtained by the backward difference for $t = 0.4$, 0.8, 1.2, 1.6, and 2.0. The time step and grid spacing are again $\Delta t = 0.08$ and $\Delta x = 0.1$, respectively. No artificial oscillation appears in the solution obtained by the backward difference. However, the approximate solution has an appreciably dull wave front at $t = 20$, whereas the wave front in the exact solution is sharp. The width of the transition region widens with time due to the damping of short waves, as indicated by the von Neumann stability analysis.

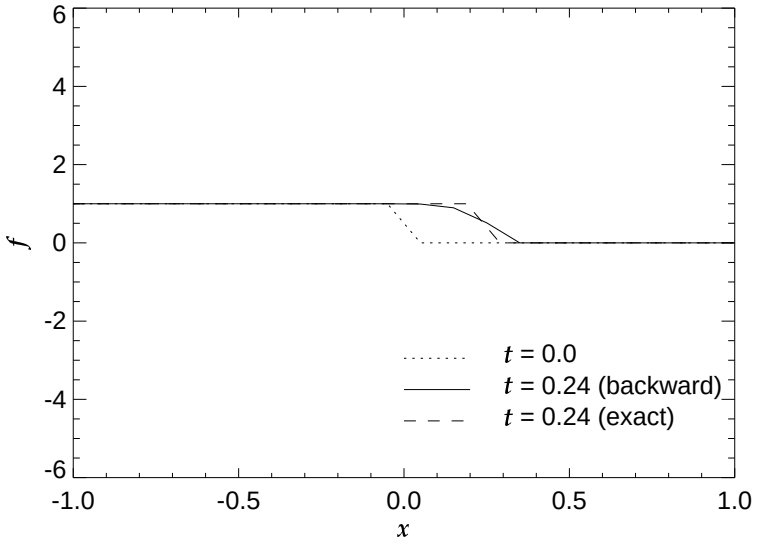

Figure 1.5: Numerical solution obtained by the backward difference. The dotted line denotes the initial condition, while the solid curve denotes the numerical solution for the stage forwarded three time steps. It is very different from the exact solution for the corresponding stage (dashed curve). The CFL number is taken to be 0.8.

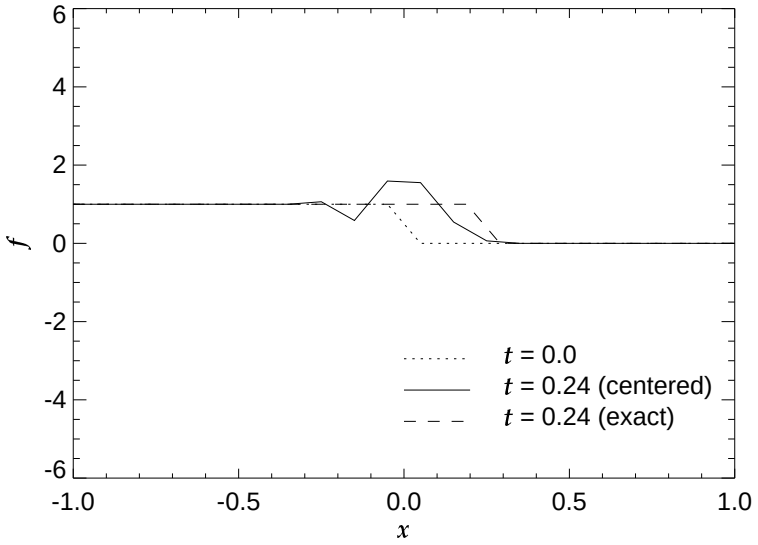

Figure 1.6: Numerical solution obtained by the central difference. The dotted line denotes the initial condition, while the solid curve denotes the numerical solution for the stage forwarded three time steps. It is very different from the exact solution for the corresponding stage (dashed curve). The CFL number is taken to be 0.8.

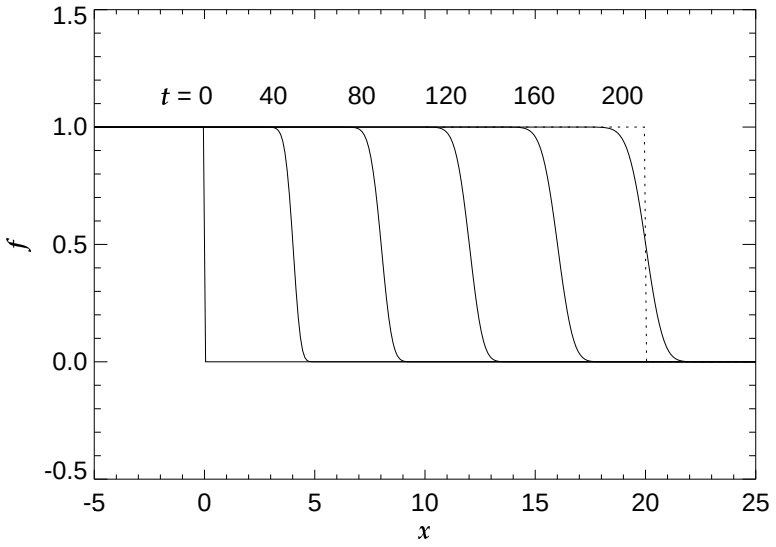

Figure 1.7: Numerical solution obtained by the backward difference. The dotted line denotes the initial condition, while the solid curve denotes the numerical solution for a later stage. It is very different from the exact solution for the corresponding stage (dashed curve). The CFL number is taken to be 0.8.

The numerical solution also depends on the CFL number. Figure 1.8 compares the solutions obtained with different values of Δt. The dashed curve denotes the solution obtained with $\Delta t = 0.04$. The wave front is rounder and the transition region is wider. This is due to two factors: the wave amplitude decreases more in each time step (smaller $|g^{(\mathrm{B})}|$) and more time steps are required to reach a given t. The solution obtained with a smaller Δt is no better, and the time step should be smaller than $\Delta t \leq 0.1$ in this problem. The solution obtained with $\Delta t = 0.101$ (dash-dotted curve) shows an unnaturally large oscillation. The von Neumann stability analysis indicates that the wave amplitude grows when the CFL number exceeds unity. When $\Delta t = 0.101$, the CFL number is 1.01 and the value of $|g^{(\mathrm{B})}|$ is only slightly larger than unity. However, the amplification is repeated 99 times at $t = 10$. Thus, the solution is not acceptable.

We can now be certain that the backward difference gives a good numerical solution as long as the CFL number is smaller than unity. Next, we consider why the backward difference succeeds while the forward time step fails.

The general solution of the wave equation [Equation (1.5)] is expressed as

$$f(x, t) = F(x - ct), \tag{1.24}$$

where F denotes an arbitrary function. Thus, we can derive the relation,

$$f(x, t + \Delta t) = f(x - c\Delta t, t). \tag{1.25}$$

On the other hand, Equation (1.8) is equivalent to

$$f(x, t + \Delta t) = \left(1 - \frac{c\Delta t}{\Delta x}\right) f(x, t) + \frac{c\Delta t}{\Delta x} f(x - \Delta x, t). \tag{1.26}$$

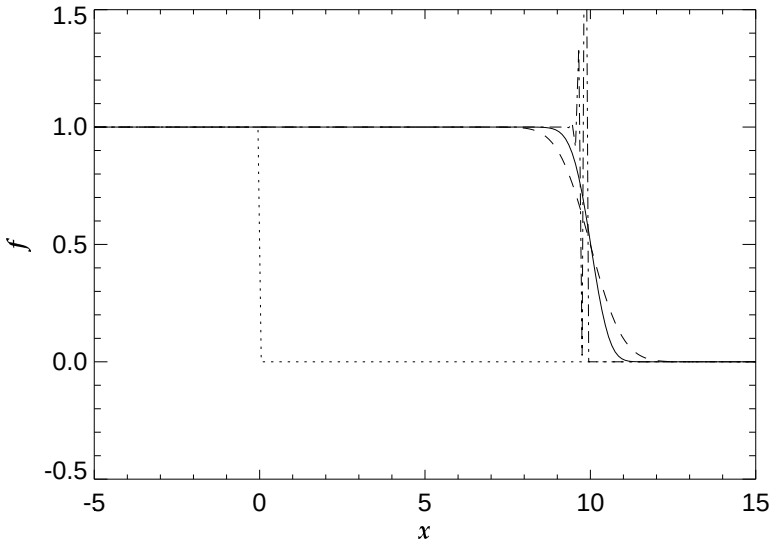

Figure 1.8: Comparison of numerical solutions obtained by the backward time step. The dotted curve denotes the initial condition at $t = 0$. The dashed curve denotes the solution at $t = 10$ obtained with $\Delta t = 0.08$. The dashed curve and dash-dotted curves denote the solutions obtained with $\Delta t = 0.04$ and 0.101, respectively.

Equation (1.26) can be derived from Equation (1.25) by interpolating f from the values at two adjacent points. When the CFL number is larger than unity, Equation (1.26) is derived not by interpolation but by extrapolation. It is not a coincidence that the condition for stability coincides with the condition for Equation (1.26) to be derived by interpolation.

Similarly, Equation (1.7) is equivalent to

$$f\,(x,\,t\,+\,\Delta t)\;=\;\left(1\,+\,\frac{c\Delta t}{\Delta x}\right)f\,(x,\,t)\;-\;\frac{c\Delta t}{\Delta x}\,f\,(x\,+\,\Delta x,\,t)\,. \tag{1.27}$$

Equation (1.27) can be derived from Equation (1.25) by extrapolation. Interpolation is modest and gives a stable approximate solution, whereas extrapolation occasionally gives an unstable solution.

We have learned that inadequate difference schemes give solutions that show unnatural oscillations. The absence of such an oscillation of numerical origin is often measured by the time evolution of the total variation in numerical hydrodynamics. When the total variation diminishes with time for any initial condition, the solution is ensured to contain no oscillation of the numerical origin. This condition is expressed as

$$\sum_j |f_{j+1,n+1}\,-\,f_{j,n+1}|\;\leq\;\sum_j |f_{j+1,n}\,-\,f_{j,n}|\,, \tag{1.28}$$

and is referred to as the TVD condition, or total variation diminishing condition. Although it may appear that the total variation should diminish, Equation (1.28) means that

$$\frac{d}{dt}\left[\int\left|\frac{\partial f}{\partial x}\right|\,dx\right]\;\leq\;0\,, \tag{1.29}$$

because

$$\sum_j |f_{j+1,n} - f_{j,n}| \simeq \int \left| \frac{\partial f}{\partial x} \right| dx \,. \tag{1.30}$$

The left-hand side of Equation (1.29) vanishes in the exact solution.[4] The TVD condition does not require reduction of the total variation. In the same way that the von Neumann stability analysis requires the wave amplitude not to grow, the TVD condition requires the total variation not to grow. The error is smaller when the decrease in the total variation is smaller.

Here, we consider the case in which the propagation speed, c, is negative. When the propagation speed is reversed, the forward difference and backward difference exchange stabilities, as mentioned in §1.2. When $c < 0$, the backward difference is unstable and the forward difference is stable in the range $-1 \leq c\Delta t/\Delta x \leq 0$. The stability criterion is then generalized as follows.

> The numerical solution is stable when the spatial derivative is replaced by the backward difference with respect to the wave propagation and the CFL number is in the range $|c|\Delta t/\Delta x$.

This criterion is valid irrespective of the sign of the propagation speed. The direction opposite to wave propagation corresponds to the upwind direction. Thus, the difference scheme given by Equation (1.8) is called the first-order upwind scheme, rather than the backward difference.

1.4 Nonlinear Wave Equation

In this section, as an example of a nonlinear wave equation, we study the Burgers equation,

$$\frac{\partial f}{\partial t} + f \frac{\partial f}{\partial x} = 0 \,. \tag{1.31}$$

The Burgers equation is the same as Equation (1.5) except that the propagation speed c is replaced with f. Before discussing the numerical method used to solve the equation, we examine the mathematical properties.

First, we examine the case in which f is nearly constant and is expressed as follows:

$$f\,(x,\,t) = f_0 + f_1(x,\,t)\,, \tag{1.32}$$

where

$$|f_1| \ll |f_0| \,. \tag{1.33}$$

Substituting Equation (1.32) into Equation (1.31) and neglecting a small second-order quantity, we obtain the following linearized equation:

$$\frac{\partial f_1}{\partial t} + f_0 \frac{\partial f_1}{\partial x} = 0 \,. \tag{1.34}$$

If we substitute f_0 for c, we obtain Equation (1.5). The wave amplitude coincides with the propagation speed in the Burgers equation.

Equation (1.31) has a special solution,

$$f = \frac{x - a}{t - b} \,, \tag{1.35}$$

[4] This statement is valid even when c is not constant.

where a and b are arbitrary constants. The value of f is always zero at $x = a$ in this special solution, which is not surprising since the propagation speed is zero at this location. Note the change in the spatial derivative, $\partial f/\partial x$. In the range $t > b$, it is positive and decreases with time, and in the range $t < b$, it is negative and increases with time.

Equation (1.35) may not be valid at $t = b$, because the value f diverges. In order to consider the validity, we consider the initial condition

$$f = \begin{cases} -1 & (x \geq 1) \\ -x & (-1 < x < 1) \\ 1 & (x \leq -1)\,, \end{cases} \tag{1.36}$$

at $t = 0$. The solution is expressed as

$$f = \begin{cases} -1 & (x \geq 1 - t) \\ \dfrac{x}{t-1} & (-1 + t < x < 1 - t)\ , \\ 1 & (x \leq -1 + t) \end{cases} \tag{1.37}$$

for the period $t < 1$. At $t = 1$, the gradient, $\partial f/\partial x$, diverges at $x = 0$ and the value of f becomes indefinite. Thus, the differential equation has no rigorous solution after this because the differential is already infinite. Still we can obtain a unique solution to the equation, as will be shown later. Equation (1.31) is equivalent to

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}\left(\frac{f^2}{2}\right) = 0\,. \tag{1.38}$$

Integrating Equation (1.38), we obtain

$$\frac{\partial}{\partial t}\int_a^b f\,dx + \left[\frac{f^2}{2}\right]_a^b = 0\,. \tag{1.39}$$

When the function f satisfies Equation (1.39) for an arbitrary interval $a \leq x \leq b$, it is a weak solution. While any solution of Equation (1.31) satisfies Equation (1.39), the reverse is not true. A function f may satisfy Equation (1.39), even when the gradient is indefinite at some points and hence it is not a solution of Equation (1.31). In fact, the function,

$$f = \begin{cases} -1 & (x > 0) \\ 1 & (x < 0) \end{cases}\ , \tag{1.40}$$

satisfies Equation (1.39) in the period $t \geq 1$ and continues to Equation (1.32) at $t = 1$. Thus, Equation (1.40) is a weak solution of Equation (1.31).

The value of f jumps from 1 to -1 at $x = 0$ in Equation (1.40). This jump is similar to the shock wave that appears in hydrodynamics. The density gradient and velocity gradient are very steep at the shock front, and sophisticated mathematics are required to handle an infinitely large quantity. Remember that the divergence apparently disappears in Equation (1.38). A similar feature is seen in Gauss's law of electromagnetism. The divergence seems to disappear in the integral form. In other words, the amount of divergence is evaluated correctly. Thus, the integral form is also used when we derive the numerical solution of Equation (1.31). [5]

[5]The integral form is also used in the finite element method.

Equation 1.38 is often referred to as the conservation form of the Burgers equation because it denotes conservation of f. Similarly, we can rewrite Equation (1.5) in conservation form as follows:

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(cf) = 0\,. \tag{1.41}$$

Then, we have

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + \frac{\frac{{f_{j,n}}^2}{2} - \frac{{f_{j-1,n}}^2}{2}}{\Delta x} = 0\,, \tag{1.42}$$

which gives us an upwind scheme for $f > 0$, and

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + \frac{\frac{{f_{j+1,n}}^2}{2} - \frac{{f_{j,n}}^2}{2}}{\Delta x} = 0\,, \tag{1.43}$$

which gives us an upwind scheme for $f < 0$. Since the wave propagation velocity is constant in Equation 1.5, either the forward difference is upwind everywhere or the backward difference is upwind everywhere. However, the propagation speed depends on f and accordingly changes its sign with time and place in the Burgers equation. Thus, we need to switch the forward difference and backward difference so that the scheme is upwind. Although the **if** clause can manage the switch both in Fortran and in C, we want to avoid the **if** clause for the following two reasons. First, the **if** clause makes a computer program more complex. Second, the **if** clause slows the computation speed appreciably. To avoid this, we rewrite Equations (1.42) and (1.43) as follows:

$$\frac{f_{j,n+1} - f_{j,n}}{\Delta t} + \frac{F^*_{j+1/2,n} - F^*_{j-1/2,n}}{\Delta x} = 0\,, \tag{1.44}$$

where

$$\begin{aligned} F^*_{j+1/2,n} &= \frac{1}{2}\left(\frac{{f_{j+1,n}}^2}{2} + \frac{{f_{j,n}}^2}{2}\right) \\ &\quad + \frac{1}{2}\frac{|f_{j+1/2,n} + f_{j-1/2,n}|}{2}\left(f_{j+1,n} - f_{j,n}\right)\,. \end{aligned} \tag{1.45}$$

The value of $F^*_{j+1/2,n}$ coincides with either ${f_{j+1,n}}^2/2$ or ${f_{j,n}}^2/2$ depending on the sign of $f_{j+1,n} + f_{j,n}$. When $f_{j+1,n} + f_{j,n} > 0$, Equations (1.44) and (1.45) are equivalent to Equation (1.42). When $f_{j+1,n} + f_{j,n} < 0$, they are equivalent to Equation (1.43). The variable, $F^*_{j+1/2,n}$, is called the numerical flux and denotes the flux at the midpoint, $x = (x_{j+1} + x_j)/2$.

Comparison of Equations (1.44) and (1.39) indicates that the variable, $f_{j,n}$, should be defined as

$$f_{j,n} \equiv \frac{1}{\Delta x}\int_{x_j - \Delta x/2}^{x_j + \Delta x/2} f(x,\ t_n)\,dx\,. \tag{1.46}$$

This definition states that the variable $f_{j,n}$ denotes the average of $f(x, t_n)$ in the cell $x_j - \Delta x/2 \le x \le x_j + \Delta x/2$. In other words, the variable $f_{j,n}$ denotes the cell average, and the spatial distance, Δx, denotes the cell width.

Figure 1.9 shows a numerical solution of the Burgers equation obtained by the upwind scheme. This solution has no unnatural oscillation and is an acceptable solution. The error is appreciable near $x = t - 1$ and $1 - t$. The gradient, $\partial f/\partial x$, changes continuously, although it should change discontinuously at $x = t - 1$ and $1 - t$. This numerical error is similar to that

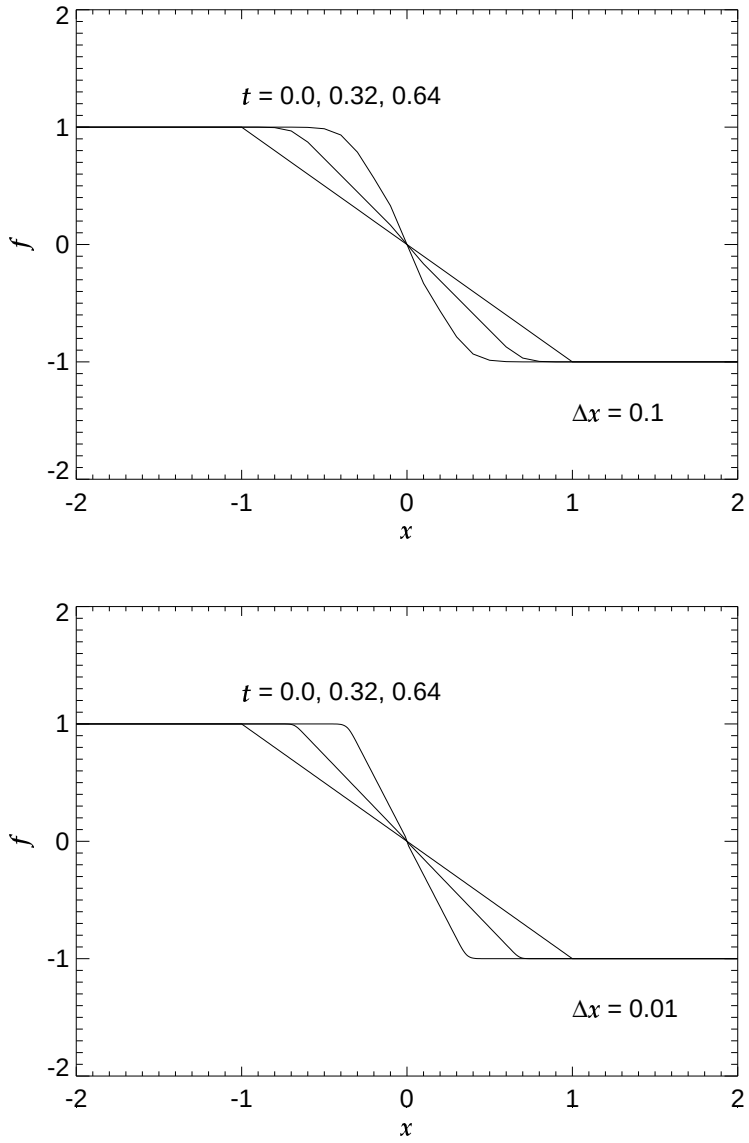

Figure 1.9: A numerical solution of the Burgers equation obtained by the upwind scheme. The initial value is given by Equation (1.40) at $t = 0$. The solid curves denote the solutions at $t = 0$, 0.32 , and 0.64. The grid spacing and time step are taken to be $\Delta x = 0.1$ and $\Delta t = 0.08$ in the upper panel and are taken to be $\Delta x = 0.01$ and $\Delta t = 0.008$ in the lower panel.

seen in the solution of the linear wave equation. The grid spacing, Δx, is 10 times smaller in the lower panel than in the upper panel. Thus, the resolution is higher and the solution is more accurate.

The next example shows the propagation of a shock wave in the Burgers equation. The initial condition is expressed as

$$f = \begin{cases} 1 & (x \le -1) \\ -x & (-1 < x < 0) \\ 0 & (x \ge 0) \end{cases} . \tag{1.47}$$

The solution is expressed as

$$f = \begin{cases} 1 & (x \le -1+t) \\ -\dfrac{x}{1-t} & (-1+t < x < 0) \\ 0 & (x \ge 0) \end{cases} , \tag{1.48}$$

for $0 < t < 1$, and

$$f = \begin{cases} 1 & (x \le t/2 - 1/2) \\ 0 & (x > t/2 - 1/2) \end{cases} , \tag{1.49}$$

for $t \ge 1$. The latter solution is a weak solution, i.e., it satisfies Equation (1.39) rather than Equation (1.31). Figure 1.10 shows the numerical solution of this problem solved by the upwind scheme. The grid spacing is taken to be rather large ($\Delta x = 0.1$) so that the error is easy to read. Thus, the shock front is not very sharp but keeps its form sharp. Remember that the wave front becomes less sharp in the numerical solution of the linear wave equation, as shown in Figure 1.7. On the other hand, the gradient becomes steeper in the Burgers equation when it is negative. The shock front maintains its sharpness due to the physical steepening. The asterisks denote positions at which the value, f, is evaluated in Figure 1.10. There is only one point in the transition region of $0.1 < f < 0.9$. Thus, the numerical solution is regarded as capturing the shock front with one point. Since the solution has no unnatural oscillation near the shock front, we call this scheme a shock capturing scheme.

Next, we examine the case in which the gradient $\partial f/\partial x$ is positive. Figure 1.11 shows the solution for which the initial condition is given by

$$f = \begin{cases} -1 & (x \le -1) \\ x & (-1 < x < 1) \\ 1 & (x \ge 1) \end{cases} . \tag{1.50}$$

Although the grid spacing is $\Delta x = 0.1$ both in the upper and lower panels, the value of f is evaluated at $x = j\,\Delta x$ in the upper panel and at $x = (j + 1/2)\,\Delta x$ in the lower panel. Note the appreciable difference between the solutions around $x = 0$. The exact solution is

$$f = \begin{cases} -1 & (x \le -1-t) \\ \dfrac{x}{t+1} & (-1-t < x < 1+t) \\ 1 & (x \ge 1+t) \end{cases} , \tag{1.51}$$

for $t > 0$. The gradient $\partial f/\partial x$ decreases with time in the exact solution and in the upper panel but maintains its initial value in the lower panel. This odd feature is called expansion shock and

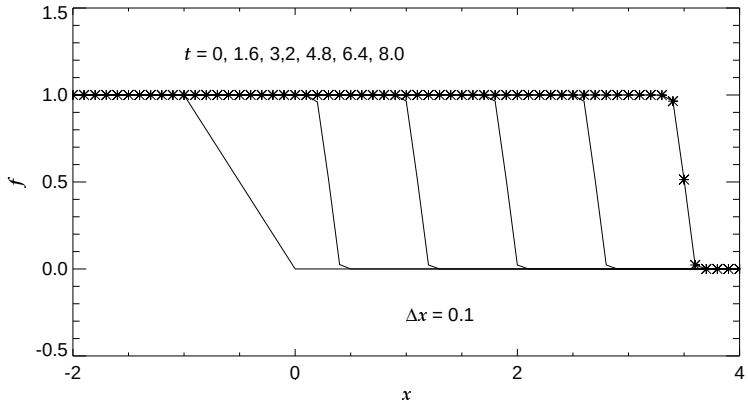

Figure 1.10: A numerical solution of the Burgers equation obtained by the upwind scheme. The grid spacing is taken to be $\Delta x = 0.1$.

occurs due to the erroneous evaluation of the numerical flux at $x = 0$ in the lower panel. The flux is $F = f^2/2 = 0$ at $x = 0$ in the exact solution, while the numerical flux is not ($F^* \neq 0$) in the lower panel.

Expansion shock is avoided in the upper panel because the numerical flux is not evaluated at $x = 0$ but at $x = \pm\Delta x/2$. The expansion shock can be avoided by another means. If we replace Equation (1.45) with

$$F^*_{j+1/2,n} = \frac{1}{2}\left(\frac{{f_{j+1,n}}^2}{2} + \frac{{f_{j,n}}^2}{2}\right) - \frac{|\lambda|}{2}\left(f_{j+1,n} - f_{j,n}\right), \tag{1.52}$$

$$|\lambda| = \begin{cases} \dfrac{|f_{j,n} + f_{j,n}|}{2} & \left(\dfrac{|f_{j+1,n} + f_{j,n}|}{2} \geq \varepsilon\right) \\ \varepsilon & (\text{otherwise}) \end{cases}, \tag{1.53}$$

$$\varepsilon = \max\left(0, \frac{f_{j+1,n} - f_{j,n}}{2}\right), \tag{1.54}$$

then the solution has no expansion shock. This modification changes the numerical flux only when $f_{j+1,n} > f_{j,n}$. Thus, the solutions shown in Figures 1.9 and 1.10 are not affected by this modification. We often refer to this procedure as the addition of the entropy condition.

1.5 Upwind Scheme for the Hydrodynamical Equations

This section deals with the hydrodynamical equations. First, we consider the simplest problem, a one-dimensional flow, and show that the hydrodynamical equations are a system of wave equations.

The mass conservation is described as

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial x}(\rho v) = 0, \tag{1.55}$$

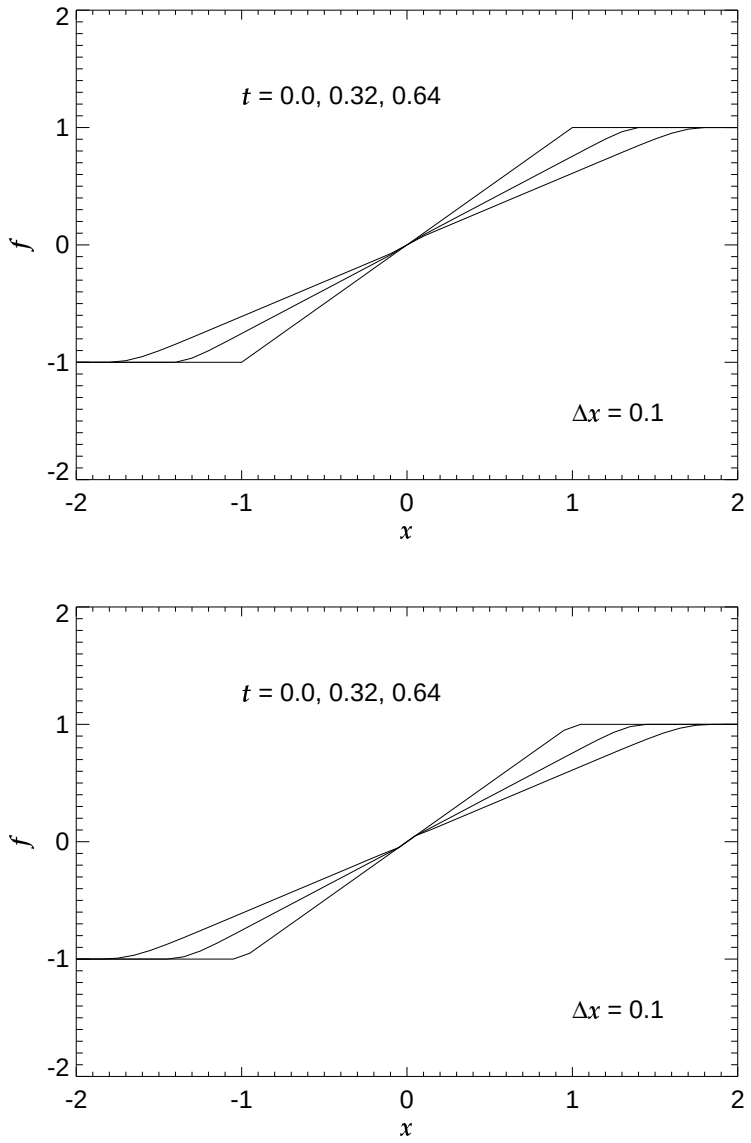

Figure 1.11: Numerical solutions of the Burgers equation. The upwind scheme is used both in the upper and lower panels. The initial condition and the grid spacing are the same. Nevertheless, the solutions are appreciably different. The value of f is evaluated at $x_j = j\Delta x$ in the upper panel and is evaluated at $x_j = (j + 1/2)\,\Delta x$ in the lower panel.

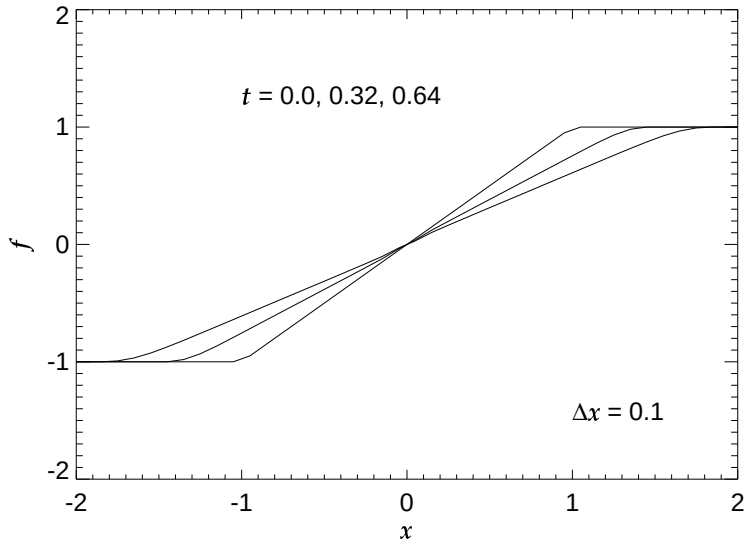

Figure 1.12: Numerical solution obtained with the addition of the entropy condition. The grid spacing and initial condition are the same as those in the lower panel of Figure 1.11. The expansion shock is removed by the addition of the entropy condition.

where ρ and v denote the density and velocity in the x-direction, respectively. The equation of motion is expressed as

$$\frac{Dv}{Dt} + \frac{\partial P}{\partial x} = 0\,, \tag{1.56}$$

where P denotes the pressure. Here, the symbol D/Dt denotes the time Lagrangian time derivative and is given by $\partial/\partial t + v\,\partial/\partial x$. Thus, Equation (1.56) is rewritten as

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + \frac{\partial P}{\partial x} = 0\,. \tag{1.57}$$

Although the pressure (P) is generally a function of density and temperature (T), for simplicity, we herein assume that the pressure depends solely on the density. [6] For later convenience, we define the speed of sound as [7]

$$a \equiv \sqrt{\frac{dP}{d\rho}}\,. \tag{1.58}$$

Then, Equation (1.57) is rewritten as

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} + a^2\frac{\partial \rho}{\partial x} = 0\,. \tag{1.59}$$

[6] When the pressure is a function of the density, the fluid is defined to be barotropic. This is a good approximation when heating and cooling are negligible and, accordingly, the entropy is constant with respect to time and space. This is also a good approximation when the temperature is maintained constant.

[7] When a gas is thermodynamically stable, the quantity under the square root is positive, $dP/d\rho$. If the pressure decreases with increasing density, the the gas is further compressed and the density increases further. Thus, the gas is thermodynamically unstable. When $dP/d\rho < 0$, the gas separates into two phases, i.e., high-density liquid and low-density gas.

The sum of Equation (1.59) and Equation (1.55) multiplied by a/ρ yields

$$\frac{a}{\rho}\frac{\partial \rho}{\partial t} + \frac{\partial v}{\partial t} + (v + a)\left(\frac{a}{\rho}\frac{\partial \rho}{\partial x} + \frac{\partial v}{\partial x}\right) = 0\,. \tag{1.60}$$

This equation is similar to the linear wave equation and the Burgers equation, and is equivalent to

$$\frac{\partial J_+}{\partial t} + (v + a)\frac{\partial J_+}{\partial x} = 0\,, \tag{1.61}$$

where

$$J_+ \equiv \int \frac{a}{\rho}\, d\rho + v\,. \tag{1.62}$$

Equation 1.61 is quite similar to the Burgers equation. Similarly, we obtain

$$\frac{\partial J_-}{\partial t} + (v - a)\frac{\partial J_-}{\partial x} = 0\,, \tag{1.63}$$

where

$$J_- \equiv \int \frac{a}{\rho}\, d\rho - v \tag{1.64}$$

from Equations (1.55) and (1.59). The hydrodynamical equations are equivalent to Equations (1.61) and (1.63) when the flow is one-dimensional and barotropic. [8]

Next, we examine Equations (1.61) and (1.63). Equation (1.61) demonstrates that a hydrodynamical wave propagates with $v + a$. Since the phase velocity is the sum of the flow velocity (v) and speed of sound (a), the wave propagates faster than the flow. Thus, the symbol J_+ denotes the amplitude of the wave that propagates toward the right, i.e., in the direction of increasing x. On the other hand, the symbol J_- denotes the amplitude of the wave that propagates toward the left. In mathematical terminology, the characteristic speed is identical to the phase velocity. The wave amplitudes, J_+ and J_-, are referred to as Riemann invariants in mathematics.

The velocity and density are expressed as a function of the two Riemann invariants. The velocity is proportional to the difference of the Riemann invariants, $v = (J_+ - J_-)/2$. Thus, the velocity change contains two components that propagate at speeds of $v + a$ and $v - a$. Similarly, the density distribution has two components.

In the previous sections, we learned that we need to use either the backward difference or the forward difference, depending on the sign of the propagation speed, when we solve a wave equation. When the speed of sound (a) is larger than the absolute value of the flow velocity ($|v|$), the two characteristic speeds have different signs. Thus, neither the forward difference nor the backward difference can treat one of the two components properly. This means that neither the forward difference nor the backward difference succeeds in solving Equations (1.55) and (1.56). On the other hand, Equations (1.61) and (1.63) describe only one wave component and so can be solved numerically either by the forward difference or by the backward difference. Thus, we have rewritten Equations (1.55) and (1.56) to obtain Equations (1.61) and (1.63).

Roe (1981) proposed a numerical scheme in which hydrodynamical equations are decomposed into components, and in which each component equation is solved with the upwind difference. This scheme is widely used because it is relatively easy to implement and gives a good numerical

[8] This argument is presented by Sakashita and Ikeuchi (1996). Their textbook entitled "Uchuu Ryuutai Rikigaku" discusses the mathematical properties of the hydrodynamical equations in greater detail.

solution. This scheme is now applied to the magnetohydrodynamical equations. Although a number of good schemes exist, we herein consider only this scheme.

Roe's original scheme is for an ideal gas. Here, the energy conservation is expressed as

$$\frac{\partial}{\partial t}(\rho E) + v\frac{\partial}{\partial x}(\rho H v) = 0, \tag{1.65}$$

where

$$E = \frac{v^2}{2} + \varepsilon, \tag{1.66}$$

$$H = E + \frac{P}{\rho}. \tag{1.67}$$

where E and ε denote the specific energy of the fluid and the specific energy of the internal energy, respectively. Thus, H denotes the sum of the specific enthalpy and specific kinetic energy. The specific energy of the fluid and the sum of the specific enthalpy and specific kinetic energy are evaluated to be

$$E = \frac{v^2}{2} + \frac{1}{\gamma - 1}\frac{P}{\rho}, \tag{1.68}$$

and

$$H = \frac{v^2}{2} + \frac{\gamma}{\gamma - 1}\frac{P}{\rho}, \tag{1.69}$$

respectively, for an ideal gas, where γ denotes the ratio of specific heat at constant pressure and constant volume, respectively. The dependent variables are expressed as a function of ρ, v, and E. The pressure is evaluated to be

$$P = (\gamma - 1)\rho\left(E - \frac{v^2}{2}\right). \tag{1.70}$$

Similarly, we obtain

$$H = \gamma\left(E - \frac{v^2}{2}\right) + \frac{v^2}{2}. \tag{1.71}$$

We have Equations (1.55), (1.57), and (1.65) and three unknowns, ρ, v, and E. Therefore, we can compute the change in the density and the change in the velocity by solving these equations simultaneously with the appropriate initial and boundary conditions.

For later convenience, we rewrite Equation (1.57) as follows:

$$\frac{\partial}{\partial t}(\rho v) + \frac{\partial}{\partial x}(\rho v^2 + P) = 0, \tag{1.72}$$

using Equation (1.55). Equation (1.72), like Equations (1.55) and (1.65), is written in conservation form. These equations can be expressed in vector form as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} = 0, \tag{1.73}$$

$$\boldsymbol{U} = \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} \rho \\ \rho v \\ \rho E \end{pmatrix}, \tag{1.74}$$

$$\boldsymbol{F} \;=\; \begin{pmatrix} F_1 \\ F_2 \\ F_3 \end{pmatrix} \;=\; \begin{pmatrix} \rho v \\ \rho v^2 \,+\, P \\ \rho H v \end{pmatrix} \,. \tag{1.75}$$

In the following, we refer to $\boldsymbol{U}$ and $\boldsymbol{F}$ as the state vector and the flux vector, respectively. This expression is useful in deriving the phase velocities of hydrodynamical waves. The flux vector is expressed as a function of the state vector, i.e.,

$$\boldsymbol{F} \;=\; \begin{pmatrix} U_2 \\ \dfrac{3\,-\,\gamma}{2}\,\dfrac{(U_2)^2}{U_1} \,+\, (\gamma\,-\,1)\,U_3 \\ \dfrac{U_2}{U_1}\left[\gamma\,U_3 \,-\, (\gamma\,-\,1)\dfrac{(U_2)^2}{2U_1}\right] \end{pmatrix} \,. \tag{1.76}$$

We can then rewrite Equation (1.73) as

$$\frac{\partial \boldsymbol{U}}{\partial t} \;+\; \boldsymbol{A}\,\frac{\partial \boldsymbol{U}}{\partial x} \;=\; 0\,, \tag{1.77}$$

where

$$\boldsymbol{A} \;\equiv\; \frac{\partial \boldsymbol{F}}{\partial \boldsymbol{U}} \tag{1.78}$$

$$\;=\; \begin{bmatrix} 0 & 1 & 0 \\ \dfrac{1}{2}\,(\gamma\,-\,3)\left(\dfrac{U_2}{U_1}\right)^2 & -\,(\gamma\,-\,3)\,\dfrac{U_2}{U_1} & \gamma\,-\,1 \\ -\,\dfrac{\gamma U_2 U_3}{(U_1)^2} \,+\, (\gamma\,-\,1)\left(\dfrac{U_2}{U_1}\right)^3 & \dfrac{\gamma U_3}{U_1} \,-\, \dfrac{3}{2}\,(\gamma\,-\,1)\left(\dfrac{U_2}{U_1}\right)^2 & \dfrac{\gamma U_2}{U_1} \end{bmatrix}$$

$$\;=\; \begin{bmatrix} 0 & 1 & 0 \\ -\,\dfrac{3\,-\,\gamma}{2}\,v^2 & (3\,-\,\gamma)\,v & \gamma\,-\,1 \\ \left(\dfrac{\gamma\,-\,1}{2}v^2 \,-\, H\right) v & H \,-\, (\gamma\,-\,1)\,u^2 & \gamma v \end{bmatrix} \,. \tag{1.79}$$

Comparison of Equations (1.77) and (1.5) reveals that the matrix $\boldsymbol{A}$ denotes the velocity of the wave.

From the previous sections, we have

$$\frac{\boldsymbol{U}_{j,n+1} \,-\, \boldsymbol{U}_{j,n}}{\Delta t} \;+\; \frac{\boldsymbol{F}^*_{j,n+1/2} \,-\, \boldsymbol{F}^*_{j,n-1/2}}{\Delta x} \;=\; 0\,, \tag{1.80}$$

$$\boldsymbol{F}^*_{j+1/2,n} \;=\; \frac{1}{2}\left[\boldsymbol{F}_{j+1,n} \,+\, \boldsymbol{F}_{j,n} \,-\, |\boldsymbol{A}|\,\left(\boldsymbol{U}_{j+1,n} \,-\, \boldsymbol{U}_{j,n}\right)\right] \,. \tag{1.81}$$

One remaining problem is the evaluation of $|\boldsymbol{A}|$, which is evaluated as follows. The velocity matrix has eigenvalues λ_i and right eigenvectors $\boldsymbol{r}_i$.

$$\boldsymbol{A}\,\boldsymbol{r}_i \;=\; \lambda_i\,\boldsymbol{r}_i\,, \tag{1.82}$$

$$\lambda_1 \;=\; v \,-\, a\,, \tag{1.83}$$

$$\lambda_2 \;=\; v\,, \tag{1.84}$$

$$\lambda_3 \;=\; v \,+\, a\,, \tag{1.85}$$

$$\boldsymbol{r}_1 = \begin{pmatrix} 1 \\ v - a \\ H - va \end{pmatrix}, \tag{1.86}$$

$$\boldsymbol{r}_2 = \begin{pmatrix} 1 \\ v \\ \frac{v^2}{2} \end{pmatrix}, \tag{1.87}$$

$$\boldsymbol{r}_3 = \begin{pmatrix} 1 \\ v + a \\ H + va \end{pmatrix}, \tag{1.88}$$

where

$$a = \sqrt{\frac{\gamma P}{\rho}}, \tag{1.89}$$

$$= (\gamma - 1)\left(H - \frac{v^2}{2}\right). \tag{1.90}$$

Using the matrix,

$$\boldsymbol{R} = (\boldsymbol{r}_1, \boldsymbol{r}_2, \boldsymbol{r}_3) \tag{1.91}$$

$$= \begin{pmatrix} 1 & 1 & 1 \\ v - a & v & v + a \\ H - va & \frac{v^2}{2} & H + va \end{pmatrix}, \tag{1.92}$$

we can diagonalize the velocity matrix as follows

$$\boldsymbol{A} = \boldsymbol{R} \begin{pmatrix} v - a & 0 & 0 \\ 0 & v & 0 \\ 0 & 0 & v + a \end{pmatrix} \boldsymbol{R}^{-1}. \tag{1.93}$$

The matrix, $\boldsymbol{R}^{-1}$, is expressed as

$$\boldsymbol{R}^{-1} = \frac{\gamma - 1}{2a^2} \begin{pmatrix} \frac{v^2}{2} + \frac{va}{\gamma - 1} & -v + \frac{a}{\gamma - 1} & 1 \\ -v^2 + \frac{2a^2}{\gamma - 1} & 2v & -2 \\ -\frac{v^2}{2} - \frac{va}{\gamma - 1} & v - \frac{a}{\gamma - 1} & 1 \end{pmatrix} \tag{1.94}$$

$$= \boldsymbol{L} \tag{1.95}$$

$$= \begin{pmatrix} \boldsymbol{\ell}_1 \\ \boldsymbol{\ell}_2 \\ \boldsymbol{\ell}_3 \end{pmatrix}. \tag{1.96}$$

The vectors $\boldsymbol{\ell}_1$, $\boldsymbol{\ell}_2$, and $\boldsymbol{\ell}_3$ are left eigenvectors because

$$\boldsymbol{\ell}_i \boldsymbol{A} = \lambda_i \boldsymbol{\ell}_i. \tag{1.97}$$

The eigenvalues, the diagonal elements of the central matrix in Equation (1.93), indicate the propagation speed of the hydrodynamical waves. Thus, we can evaluate $|\boldsymbol{A}|$ to be

$$|\boldsymbol{A}| \;=\; \boldsymbol{R} \begin{pmatrix} |v \,-\, a| & 0 & 0 \\ 0 & |v| & 0 \\ 0 & 0 & |v \,+\, a| \end{pmatrix} \boldsymbol{L}\,. \tag{1.98}$$

The eigenvalues $v + a$ and $v - a$ denote the phase velocities of sound waves propagating toward the right and toward the left, respectively. These waves also appear in a barotropic fluid. The other wave, the phase velocity of which coincides with the flow velocity, disappears in a barotropic fluid.

Equation (1.98) should be evaluated from $\boldsymbol{U}_{j+1,n}$ and $\boldsymbol{U}_{j,n}$, i.e., from an average state vector, as in the case of the Burgers equation. Roe proposed evaluation of $\boldsymbol{R}$, $\boldsymbol{L}$, $\lambda_i\,(i \,=\, 1,\; 2,\; 3)$ by the following averages:

$$\bar{v} \;=\; \frac{\sqrt{\rho_j}v_j \,+\, \sqrt{\rho_{j+1}}v_{j+1}}{\sqrt{\rho_j} \,+\, \sqrt{\rho_{j+1}}}\,, \tag{1.99}$$

$$\bar{H} \;=\; \frac{\sqrt{\rho_j}H_j \,+\, \sqrt{\rho_{j+1}}H_{j+1}}{\sqrt{\rho_j} \,+\, \sqrt{\rho_{j+1}}}\,, \tag{1.100}$$

$$\bar{a}^2 \;=\; (\gamma \,-\, 1)\left(\bar{H} \,-\, \frac{\bar{v}^2}{2}\right). \tag{1.101}$$

Although computing $|\boldsymbol{A}|$ may appear difficult, the right-hand side of Equation (1.81) can be rewritten as

$$\boldsymbol{F}^*_{j+1/2,n} \;=\; \frac{1}{2}\left[\boldsymbol{F}_{j+1,n} \,+\, \boldsymbol{F}_{j,n} \,-\, \sum_{k=1}^{3} |\lambda_k|\, w_k\, \boldsymbol{r}_k\right], \tag{1.102}$$

where

$$w_1 \;=\; \frac{1}{2\bar{a}}\left[\frac{P_{j+1} \,-\, P_j}{\bar{a}} \,-\, \bar{\rho}\,(v_{j+1} \,-\, v_j)\right], \tag{1.103}$$

$$w_2 \;=\; \rho_{j+1} \,-\, \rho_j \,-\, \frac{P_{j+1} \,-\, P_j}{\bar{a}^2}\,, \tag{1.104}$$

$$w_3 \;=\; \frac{1}{2\bar{a}}\left[\frac{P_{j+1} \,-\, P_j}{\bar{a}} \,+\, \bar{\rho}\,(v_{j+1} \,-\, v_j)\right], \tag{1.105}$$

because

$$\boldsymbol{U}_{j+1} \,-\, \boldsymbol{U}_j \;=\; \sum_{k=1}^{3} w_j\, \boldsymbol{r}_j\,, \tag{1.106}$$

for any $\boldsymbol{U}_j$ and $\boldsymbol{U}_{j+1}$.

Here, we consider a wave having a phase velocity of v. As shown in Equation (1.104), the amplitude is w_2 and is evaluated to be

$$w_2 \;\simeq\; \left(\frac{\partial P}{\partial s}\right)_{\rho} (s_{j+1} \,-\, s_j)\,, \tag{1.107}$$

because

$$\delta P \;=\; \left(\frac{\partial P}{\partial \rho}\right)_{s} \,+\, \left(\frac{\partial P}{\partial s}\right)_{\rho} \delta s \tag{1.108}$$

$$\;=\; a^2\,\delta\rho \,+\, \left(\frac{\partial P}{\partial s}\right)_{\rho} ds\,, \tag{1.109}$$

where s denotes the specific entropy. This means that the wave amplitude is proportional to the entropy difference. Thus, the wave is referred to as the entropy wave. When only the entropy wave exists, the pressure and velocity are uniform and the temperature changes. A cold front, in which the temperature drops appreciably, is an example of an entropy wave. [9]

As in the case of the Burgers equation, the characteristic speeds should be modified as

$$|\lambda_1| = \max(|\bar{v} - \bar{a}|, \varepsilon_1), \tag{1.110}$$

$$\varepsilon_1 = \max(\lambda_{j+1,1} - \lambda_{j,1}, 0), \tag{1.111}$$

$$\lambda_{j,1} = v_j - \sqrt{\frac{\gamma P_j}{\rho_j}}, \tag{1.112}$$

$$|\lambda_3| = \max(|\bar{v} + \bar{a}|, \varepsilon_3), \tag{1.113}$$

$$\varepsilon_3 = \max(\lambda_{j+1,3} - \lambda_{j,3}, 0), \tag{1.114}$$

$$\lambda_{j,3} = v_j - \sqrt{\frac{\gamma P_j}{\rho_j}}, \tag{1.115}$$

in order to avoid the expansion shock. Entropy correction is not required for the entropy wave (λ_2). Entropy waves never evolve into an expansion shock.

Figures 1.13 and 1.14 show numerical examples in which a shock tube problem is solved by the Roe scheme. Figures 1.13 and 1.14 confirm that the Roe scheme can solve the hydrodynamical equations without difficulty, even when the initial density and pressure distributions are discontinuous. The figures show no unnatural oscillation around the shock wave.

Next, we evaluate the numerical error by comparing the solutions for different values of Δx. Figure 1.15 compares solutions obtained with different values of Δx. The initial condition is the same as that of the solution shown in Figures 1.13 and 1.14. The dash-dotted curve denotes the density distribution obtained with $\Delta x = 0.1$, while the solid and dashed curves denote those obtained with $\Delta x = 0.001$ and 0.01, respectively. All of the curves denote the density distribution at $t = 0.8$. As Δx decreases, the density decreases gradually in the range $-1.04 \leq x \leq, 0.03$ and approaches the distribution having discontinuities at $x = 0.78$ and 1.43. These features indicate that the numerical solution obtained by the Roe scheme converges to the exact solution in the limit of infinitesimal Δx. Convergence is an important issue, and a stable scheme is of limited value if the error remains too high.

The speed of convergence depends on the position. The density and temperature change in the rarefaction wave, at the contact discontinuity ($x = 0.78$), and at the shock front ($x = 1.43$). As demonstrated in Figure 1.15, the convergence is fast at the shock front but is very slow at the contact discontinuity. The dashed curve ($\Delta x = 0.01$) is a fairly good solution around the shock front, while the contact discontinuity is not sharp enough, even in the solution with $\Delta x = 0.001$. The speed of convergence depends little on the initial condition. The contact discontinuity is always less sharp than the shock front. The change in temperature becomes vague, although it should change sharply at the contact discontinuity.

Figure 1.16 shows the speed of convergence quantitatively. The abscissa denotes the grid spacing Δx, while the ordinate denotes the numerical errors in the density (solid curve) and velocity (dashed curve) at $(x, t) = (-0.75, 0.8)$. The time step is taken to be $\Delta t = 0.5\,\Delta x$ so that the CFL number is the same. The errors are proportional to the grid spacing, i.e., they are of the first order. This is because the truncation error is proportional to Δx^2 in the first-order

[9] The entropy is given by $s = \log P - \gamma \log \rho$ for an ideal fluid with specific heat ratio γ. The pressure is then expressed as $P = e^s \rho^\gamma$.

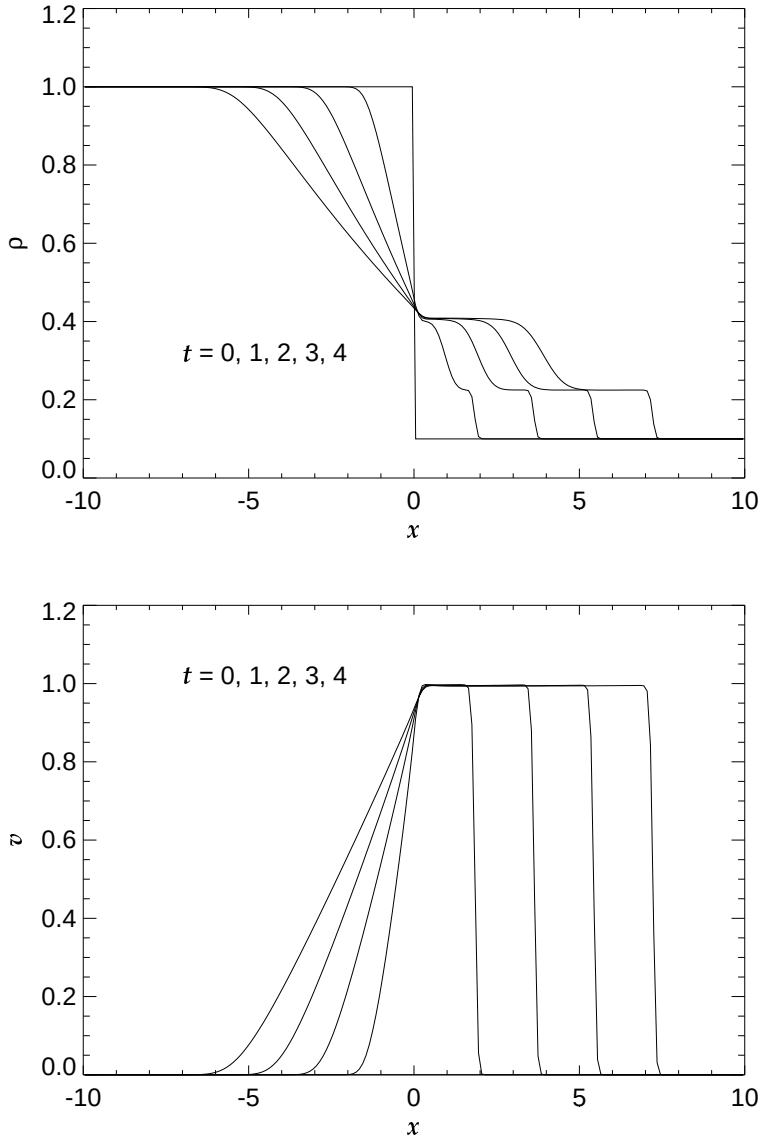

Figure 1.13: A numerical solution obtained with the Roe scheme. The initial density and pressure are $\rho = 0.1$ and $P = 0.05$ in the region $x > 0$ and are $\rho = 1.0$ and $P = 1.0$ in the region $x < 0$. The initial velocity is $v = 0$ over the entire region. The specific heat ratio is set to be $\gamma = 5/3$. The grid spacing is $\Delta x = 0.1$ and the time step is $\Delta t = 0.04$. The upper panel shows the density, and the lower panel shows the velocity.

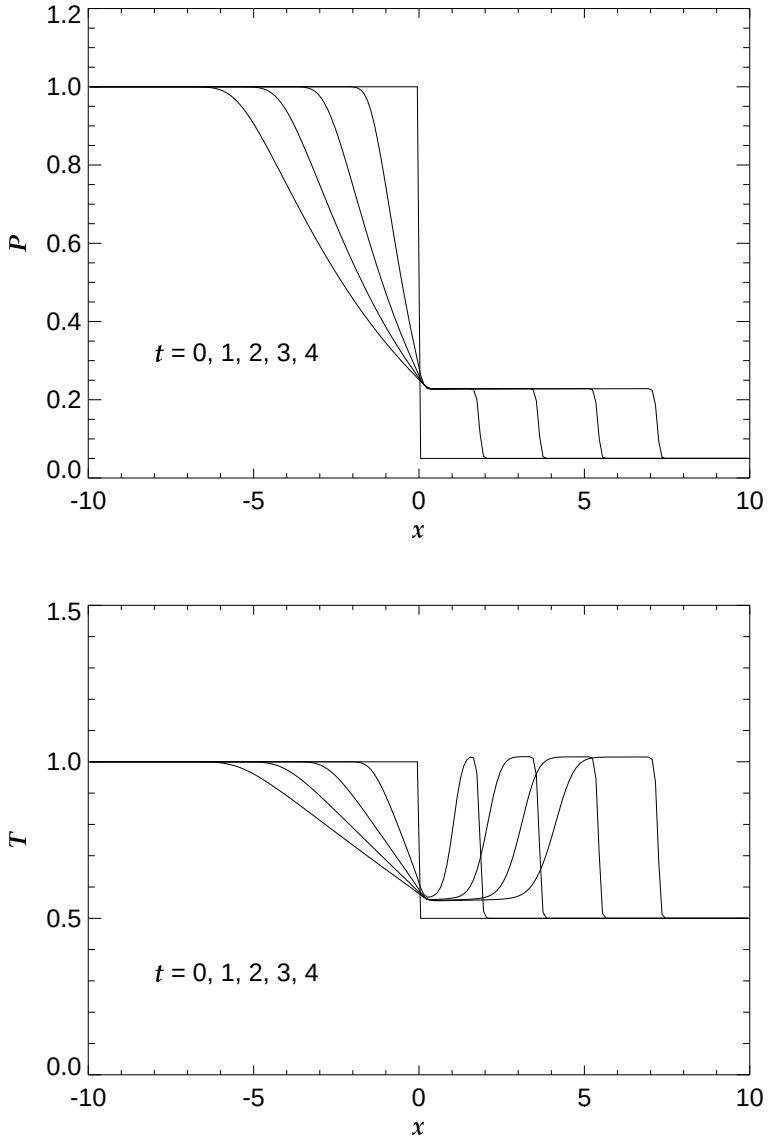

Figure 1.14: A numerical solution obtained with the Roe scheme. The initial density and pressure are $\rho = 0.1$ and $P = 0.05$ in the region $x > 0$ and are $\rho = 1.0$ and $P = 1.0$ in the region $x < 0$. The initial velocity is $v = 0$ over the entire region. The grid spacing is $\Delta x = 0.1$ and the time step is $\Delta t = 0.04$. The upper panel shows the pressure, and the lower panel shows the temperature.

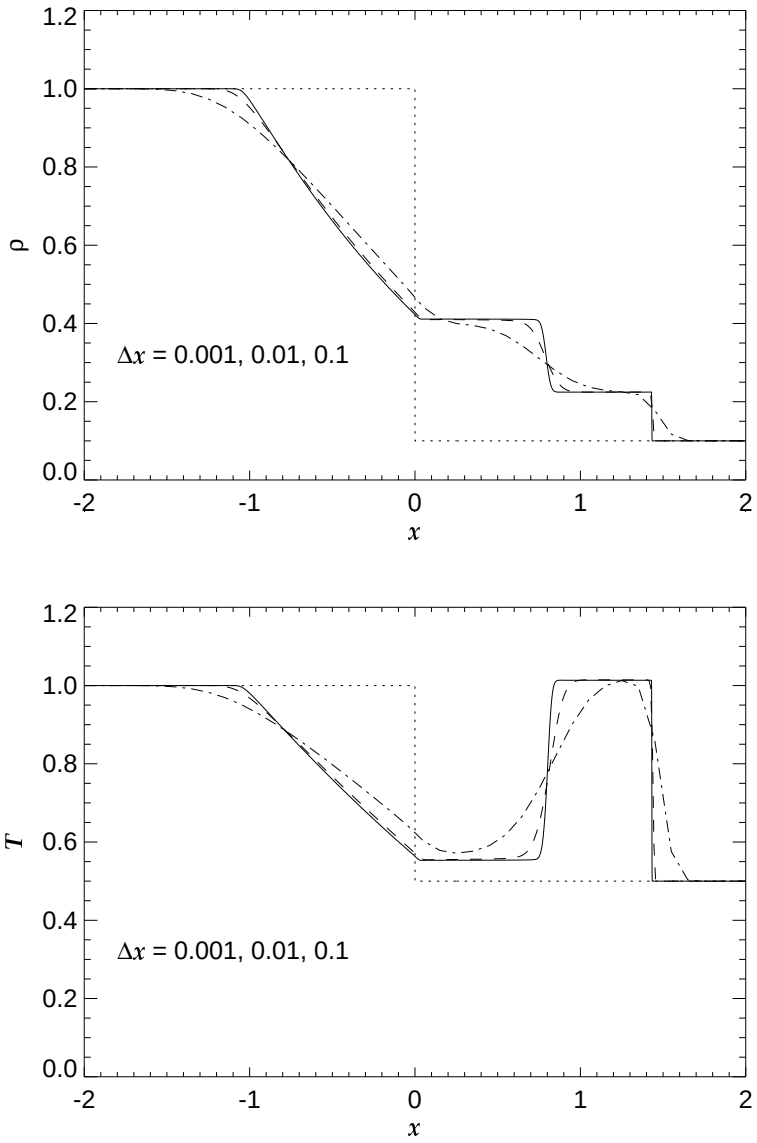

Figure 1.15: Comparison of the numerical solutions with different values of Δx. The initial condition is the same as that of the solution shown in Figures 1.13 and 1.14. The upper and lower panels show the density and temperature, respectively. The solid curve denotes the solution obtained with $\Delta x = 0.001$, and the dashed curve shows the solution obtained with $\Delta x = 0.01$. The dash-dotted curve denotes the solution obtained with $\Delta x = 0.1$. The time step is taken to be $\Delta t = 0.4\,\Delta x$ in all of the solutions.

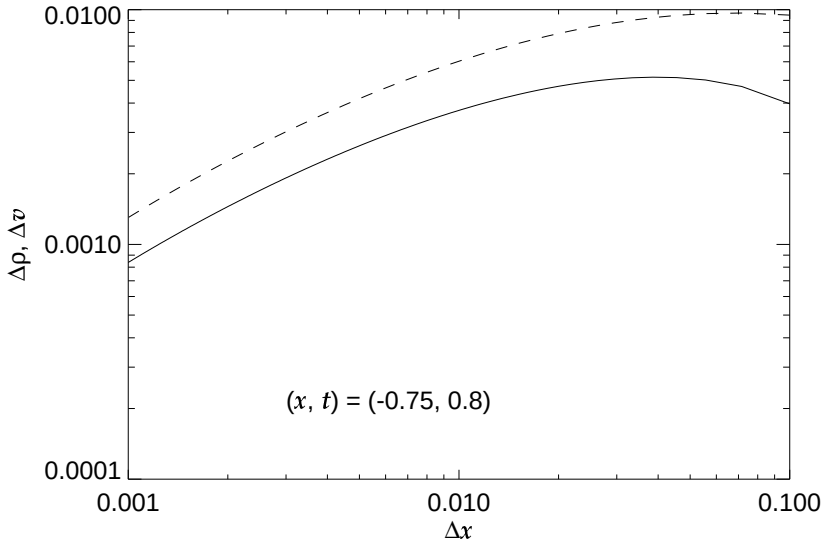

Figure 1.16: The errors in the density and velocity are shown as a function of the grid spacing for the solutions given in Figures 1.13 and 1.14. The errors are evaluated at $(x,\ t) = (-0.75,\ 0.8)$.

upwind scheme. The number of time steps required to reach a given t is inversely proportional to Δx because the time step is proportional to Δx. A scheme is of first-order accuracy when the error is proportional to the grid spacing. A higher-order scheme is explained in the next section.

The next example gives the solution for the following initial condition:

$$(\rho,\ P) = \begin{cases} (0.5,\ 0.2) & (x \le -1) \\ (1.0,\ 1.0) & (-1 < x \le -1) \\ (0.1,\ 0.05) & (x > 1) \end{cases} . \tag{1.116}$$

The initial velocity vanishes everywhere ($v = 0$). The upper and lower panels denote the density and pressure distributions, respectively, by gray scale and contour. The The density and pressure are higher in the black regions. The contours radiate from $(x,\ t) = (0,\ \pm 1)$ in the early phase in the diagrams. The contours are straight because the waves have constant phase velocities. The two rarefaction waves cross each other at $t \simeq 1$. After crossing, the phase velocities change and the slopes of the contours change. Similar results are obtained when the initial velocity is not spatially constant, and also when the changes in initial density, velocity and pressure are more complex. These three types of waves radiate from any point at which the initial density, velocity, or pressure changes. The waves propagate and cross each other. Thus, a scheme can solve the hydrodynamical equations for a given initial condition if it can solve a shock tube problem. Thus, shock tube problems are often used as test problem for numerical simulation code.

Before proceeding further, we consider why the Roe scheme succeeds in solving the shock wave. The primary reason is that we have diagonalized the velocity matrix. Equation (1.77)

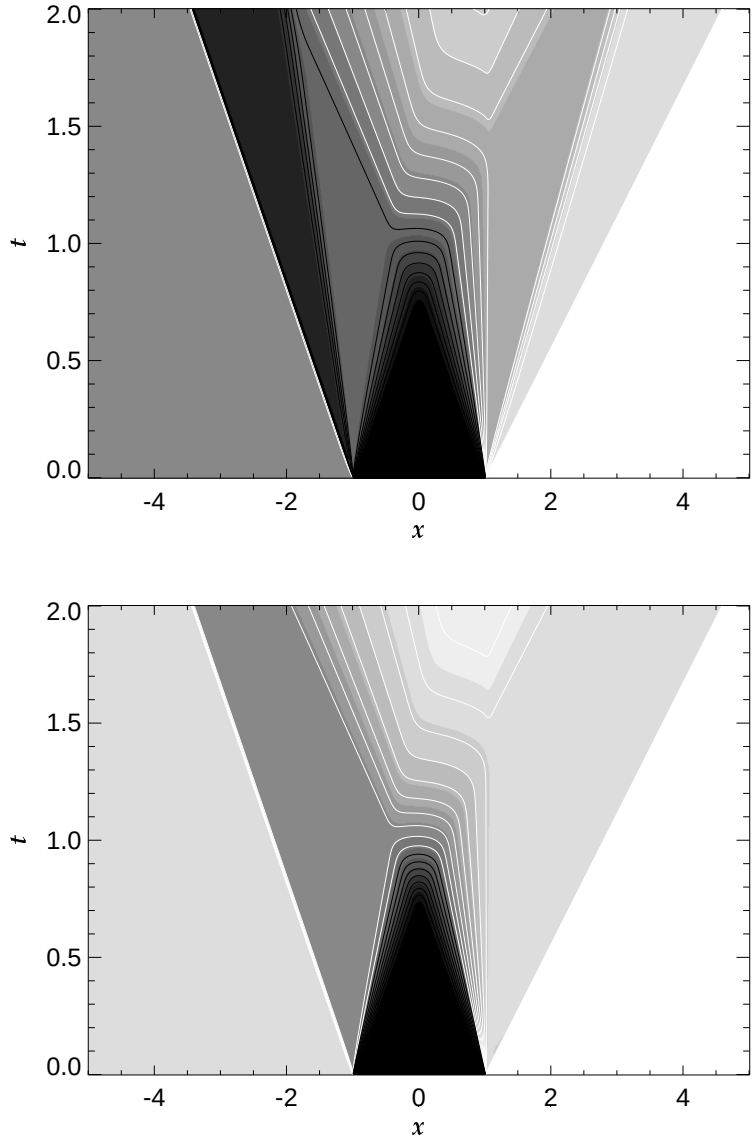

Figure 1.17: A test problem in which the initial density and pressure change at two points. The upper panel denotes the evolution of the density distribution by gray scale and contour, and the lower panel denotes the evolution of the pressure distribution.

can be rewritten as

$$\frac{\partial \boldsymbol{J}}{\partial t} + \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix} \frac{\partial \boldsymbol{J}}{\partial x} = 0 , \tag{1.117}$$

where

$$d\boldsymbol{J} = \boldsymbol{L}\, d\boldsymbol{U} , \tag{1.118}$$

by multiplying the matrix $\boldsymbol{L}$. The newly defined variables, J_1, J_2, and J_3, are Riemann invariants. The multiplications of $\boldsymbol{R}$ and $\boldsymbol{L}$ transform the hydrodynamical equations into a set of simple wave equations. However, Equations (1.99) – (1.101) also contribute to the success of the Roe scheme. Thanks to these equations, the relation,

$$\boldsymbol{F}(\boldsymbol{U}_{j+1}) - \boldsymbol{F}(\boldsymbol{U}_j) = \boldsymbol{A}\,(\boldsymbol{U}_{j+1} - \boldsymbol{U}_j) , \tag{1.119}$$

holds exactly for any given $\boldsymbol{U}_j$ and $\boldsymbol{U}_{j+1}$. This is the virtue of Roe's averaging. When $\boldsymbol{U}_j = \boldsymbol{U}_{j+1}$, the velocity matrix coincides with $\partial \boldsymbol{F}/\partial \boldsymbol{U}$. Furthermore, Equation (1.101) guarantees that a^2 is always positive, and hence all of the phase velocities are real (not complex). These three conditions are referred to as the U property. If the velocity matrix $\boldsymbol{A}$ fulfills the U property in a numerical scheme, then the hyperbolic differential equations can be solved by the upwind scheme and the solution is stable and free from numerical oscillation. Such velocity matrixes have been found for two-dimensional flows, three-dimensional flows and non-ideal-gas flows. When a numerical scheme is based on the velocity matrix satisfying the U property, it is generally referred to as a Roe type scheme, even when it is not derived by Roe. A Roe type scheme is also obtained for the magnetohydrodynamical equations. Some numerical schemes are classified as Roe type schemes by the developer even when they do not satisfy the U property. These schemes provide relatively good solutions, although they are often unstable with respect to strong shock waves.

The velocity matrix that satisfies the U property for an arbitrary equation of state was obtained by Nobuta and Hanawa (1999, ApJ, vol. 510, p. 614). They applied this scheme to a gas in which black body radiation was taken into account, and the scheme was shown to be applicable to degenerate electron gas. Eulderink and Mellema (1993, A&AS, 110, p. 587) obtained the velocity matrix for special relativistic hydrodynamical equations. Their velocity matrix satisfies the U property. A Roe type scheme was derived for the magnetohydrodynamical equations for the first time by Brio and Wu (1988, J. Comput. Phys., 75, p. 400). Their velocity matrix does not satisfy the U property, except for $\gamma = 2$. The velocity matrix satisfying the U property for any γ was presented by Cargo and Gallice (1997, J. Comput. Phys., 136, 446).

1.6 Higher-order Accuracy

In the previous section, a numerical method by which to solve one-dimensional hydrodynamical equations was described. In this section, we introduce a method by which to obtain a solution of higher-order accuracy. Numerical methods used to solve two-dimensional and three-dimensional hydrodynamical equations are shown in the next section.

The upwind scheme, which was shown in the previous section, is of the first-order in space and time. As the grid spacing and time step are taken to be smaller, the truncation error is smaller. However, an enormous computation time is required to obtain an accurate solution with the first-order scheme. Most numerical simulations employ a second-order or even higher-order scheme to reduce the computation time. In this section, we introduce the Monotone

Upstream-centered Scheme for Conservation Laws (MUSCL) to achieve second-order accuracy. Since MUSCL is rather difficult, easier methods may be desired. Thus, we first understand necessity of a rather complicated method.

First, we consider the one-dimensional wave equation to simplify the problem. Equation (1.9), the central difference, is of second-order accuracy because the error is a small third-order quantity. However, the solution obtained by Equation (1.9) is unstable and shows unnatural oscillations around a discontinuity. Is it possible to realize a higher-order accuracy while maintaining stability and avoiding unnatural oscillations? We shall examine the case in which the solution of Equation (1.5) at step $n+1$ should be a linear combination of the solution at step n, i.e.,

$$f_{j,n+1} \;=\; \sum_k B_k \, f_{j+k,n} \,. \tag{1.120}$$

This expression includes the forward difference, the backward difference, and the central difference, which are all shown in Section 1.2. If Equation (1.120) denotes the backward difference, then the coefficients, B_k, are given by

$$B_k \;=\; \begin{cases} 1 - \dfrac{c\Delta t}{\Delta x} & (k \;=\; 0) \\ \dfrac{c\Delta t}{\Delta x} & (k \;=\; -1) \\ 0 & (\text{otherwise}) \end{cases} \,. \tag{1.121}$$

Similarly, for the central difference scheme, they are given by

$$B_k \;=\; \begin{cases} 1 & (k \;=\; 0) \\ -\dfrac{c\Delta t}{2\Delta x} & (k \;=\; 1) \\ \dfrac{c\Delta t}{2\Delta x} & (k \;=\; -1) \\ 0 & (\text{otherwise}) \end{cases} \,. \tag{1.122}$$

By arranging B_k, we can create an infinite number of schemes. The Godunov theorem states that no scheme expressed by Equation (1.120) can achieve second- or higher-order accuracy without unnatural numerical oscillations.

> **Godunov theorem**: Any second- or higher-order scheme expressed by Equation (1.120) cannot achieve monotonicity of the solution.

The gradient $\partial f / \partial x$ changes its sign in the region in which an unnatural numerical oscillation arises. However, the gradient should remain positive if the initial gradient is positive because the wave form should keep the original sign. In other words, the wave amplitude should always increase monotonically with increasing x if the initial amplitude increases monotonically with increasing x. It is desired that the monotonicity be preserved in a numerical solution. The monotonicity is preserved if and only if

$$B_k \;\geq\; 0 \,. \tag{1.123}$$

The proof of this equation is not easy. First, we prove that the monotonicity is preserved if Equation (1.123) holds. Equation (1.120) yields

$$f_{j+1,n+1} \;-\; f_{j,n} \;=\; \sum_k B_k \, (f_{j+1+k,n} \;-\; f_{j+k,n}) \,. \tag{1.124}$$

If $B_k \geq 0$ and $f_{j+1,n} - f_{j,n} \geq 0$ for any given k at $t = n\Delta t$, then $f_{j+1,n+1} - f_{j,n+1} \geq 0$. The same is true for $f_{j+1,n} - f_{j,n} \leq 0$. Thus, the monotonicity of function f is preserved if $B_k \geq 0$.

The necessity of $B_k \geq 0$ for maintaining monotonicity is proven by reduction to absurdity. Suppose $B_m < 0$ for a given m. If the initial condition is given by

$$f_{j,n} = \begin{cases} 0 & (j > 0) \\ 1 & (j \leq 0) \end{cases}, \tag{1.125}$$

we obtain

$$\begin{aligned} f_{-m+1,n+1} - f_{-m,n+1} &= \sum_k B_k \left(f_{k-m+1,n} - f_{k-m,n} \right) && (1.126) \\ &= B_m \left(f_{1,n} - f_{0,n} \right) && (1.127) \\ &> 0 \,. && (1.128) \end{aligned}$$

This solution violates monotonicity, because the function $f_{j,n}$ increases monotonically with j while $f_{j,n+1}$ does not. Thus, we have proven that the monotonicity is preserved if and only if $B_k \geq 0$.

Next, we examine the truncation error using the Taylor series. Taylor series expansion gives

$$f_{j,n+1} = f_{j,n} + \sum_{m=1}^{\infty} \frac{\partial^m f}{\partial t^m} \frac{(\Delta t)^m}{m!} \,, \tag{1.129}$$

which is rewritten as

$$f_{j,n+1} = f_{j,n} + \sum_{m=1}^{\infty} \frac{\partial^m f}{\partial x^m} \frac{(c\Delta t)^m}{m!} \,, \tag{1.130}$$

because

$$\frac{\partial^m f}{\partial t^m} = (-c)^m \frac{\partial^m f}{\partial x^m} \tag{1.131}$$

is derived from Equation (1.5). Another Taylor series expansion gives

$$f_{j+k,n} = f_{j,n} + \sum_{k=1}^{\infty} \frac{\partial^k f}{\partial x^k} (k\Delta x)^m \,. \tag{1.132}$$

Comparison of Equations (1.132) and (1.130) gives

$$\begin{aligned} \sum_k B_k &= 1 && (1.133) \\ \sum_k k^m B_k &= \left(-\frac{c\Delta t}{\Delta x} \right)^m \,. && (1.134) \end{aligned}$$

from the condition that Equation (1.134) gives the condition that the scheme is of m-th order accuracy. We can easily confirm that the backward difference satisfies Equations (1.133) and (1.134) for $m = 1$ but not for $m = 2$. The central difference satisfies Equation (1.134) for $m \leq 2$, although it does not guarantee the monotonicity of the solution because $B_1 < 0$.

We obtain

$$\sum_k \left[k^2 - 2k \frac{c\Delta t}{\Delta x} + \left(\frac{c\Delta t}{\Delta x} \right)^2 \right] B_k = 0 \,, \tag{1.135}$$

from the sum of Equation 1.133 multiplied by $(c\Delta t/\Delta x)^2$, Equation (1.134) for $m = 1$ multiplied by $2\,(c\Delta t/\Delta x)$, and Equation (1.134) for $m = 2$. Equation (1.135) is rewritten as

$$\sum_k \left(k - \frac{c\Delta t}{\Delta x}\right)^2 B_k = 0\,. \tag{1.136}$$

If $B_k \geq 0$ for any k, Equation (1.136) does not hold except when $c\Delta t/\Delta x$ is an integer. Thus, Equations (1.133) and (1.134) for $m = 1$ and 2 do not hold simultaneously as long as $B_k \geq 0$ for any k except Δt.

The Godunov theorem tells us that conservation of monotonicity is a tough constraint. However, monotonicity is violated and unnatural oscillations arise only in a limited region. As shown in Section 1.2, the central difference causes numerical oscillation only in the region in which the value of f changes drastically. Thus, to avoid unnatural oscillations, we need care about monotonicity only in this region. When the value of f changes drastically, the Taylor series does not provide a good approximation and second-order accuracy is not important in this case. Second-order accuracy has no value in particular at the shock front because the density and velocity are discontinuous there. This allows for a balance between monotonicity and second-order accuracy to be struck. Second-order accuracy is realized almost everywhere, except in small regions in which the physical quantities change drastically.

Next, we reexamine second-order accuracy, paying attention to the numerical flux. Substituting the numerical flux defined as

$$F_{j+1/2,n} = c\left(\frac{f_{j+1,n} + f_{j,n}}{2}\right), \tag{1.137}$$

into Equation (1.44), we obtain the central difference. The numerical flux denotes the flux at $x = (x_{j+1} + x_j)/2$, as is indicated by the subscripts. The Taylor series expansion of the right-hand side of Equation (1.137) is of first-order accuracy when expanded in the Taylor series. On the other hand, the numerical flux is evaluated either at $x = x_j$ or x_{j+1} in the backward difference and the forward difference. Thus, the truncation error is larger.

We next compare the accuracy of the first-order upwind scheme and that of the central difference using Figure 1.18. Suppose that the initial condition is given by f_j. The function is implicitly assumed to be a step function (dashed line) in the upwind scheme because the numerical flux at $x = x_{j+1/2}$ is evaluated by either f_j or f_{j+1}. On the other hand, the function is implicitly assumed to be a stepwise linear function (solid curve) in the central difference, because the numerical flux is evaluated by the average of f_j and f_{j+1}. If we can approximate the initial condition not by the simple linear interpolation, but by another stepwise linear function, we will be able to achieve first-order accuracy in the numerical flux and second-order accuracy in the solution.

Fortunately, in addition to for the simple linear interpolation, we have several ways to approximate a function by a stepwise function. For example, extrapolating from the left-hand side,

$$f^{(\mathrm{L})}_{j+1/2} = f_j + \frac{1}{2}\,(f_j - f_{j-1}) \tag{1.138}$$

$$= \frac{3\,f_j - f_{j-1}}{2}\,, \tag{1.139}$$

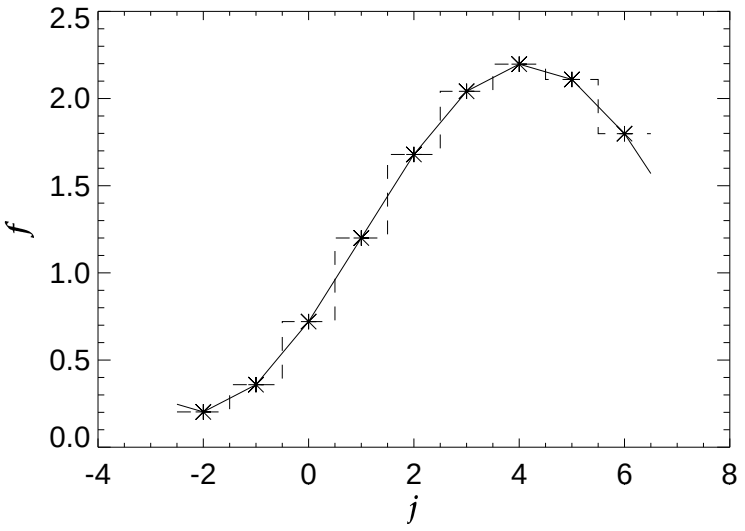

Figure 1.18: Example of simple linear interpolation.

is of first-order accuracy. Similarly, extrapolating from the right-hand side,

$$f^{(\mathrm{R})}_{j+1/2} = f_{j+1} - \frac{1}{2}(f_{j+2} - f_{j+1}) \tag{1.140}$$

$$= \frac{3\,f_{j+1} - f_{j+2}}{2}, \tag{1.141}$$

is of first-order accuracy. Here, $f^{(\mathrm{L})}_{j+1/2}$ and $f^{(\mathrm{R})}_{j+1/2}$ provide a solution of second- order accuracy if used in the evaluation of the numerical flux.

The use of $f^{(\mathrm{L})}_{j+1/2}$ and $f^{(\mathrm{R})}_{j+1/2}$ requires careful consideration. Figure 1.19 shows an example of extrapolation by Equations (1.139) and (1.141). The initial data are the same as those shown in Figure 1.18. Unnatural irregularities appear around the local minima and local maxima in the extrapolation.

It is a natural consequence of the Godunov theorem that the extrapolation violates monotonicity. The Godunov theorem states that the monotonicity is preserved only by the upwind scheme, i.e., by the step function.

Next, we modify Equation (1.139) to guarantee monotonicity. If we replace Equation (1.139) with

$$f^{*(\mathrm{L})}_{j+1/2} = \begin{cases} f_j & \Delta_{j+1/2}\,\Delta_{j-1/2} \le 0 \\ f_j + \dfrac{\Delta_{j-1/2}}{2} \min\left(1, \left|\dfrac{\Delta_{j+1/2}}{\Delta_{j-1/2}}\right|\right) & (\text{otherwise}) \end{cases}, \tag{1.142}$$

$$\Delta_{j+1/2} = f_{j+1} - f_j, \tag{1.143}$$

$$\Delta_{j-1/2} = f_j - f_{j-1}, \tag{1.144}$$

then the monotonicity problem is resolved. Equation (1.144) is of first-order accuracy only when the sign of $\Delta_{j+1/2}$ is the same as that of $\Delta_{j-1/2}$, i.e., only when the function increases

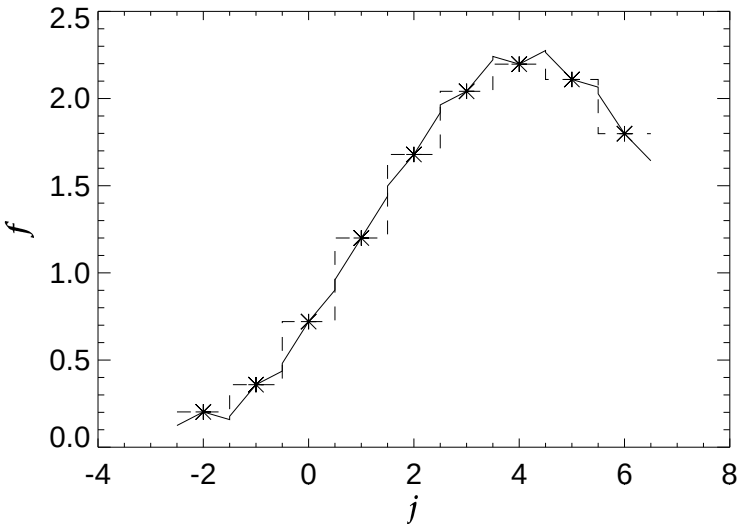

Figure 1.19: An example of extrapolation by Equations (1.139) and (1.141). The data are the same as those of Figure 1.18.

or decreases monotonically in the interval $x_{j-1} \le x \le x_{j+1}$. Otherwise, Equation (1.144) is approximated by f_j because the gradient changes and the function has either a local minimum or a local maximum in the interval. When the sign of $\Delta_{j+1/2}$ is the same as that of $\Delta_{j-1/2}$, the minimum absolute gradient is used for the extrapolation. In most textbooks, Equation (1.142) is expressed as

$$f_{j+1/2}^{(L)} = f_j + \frac{\Delta_{j-1/2}}{2} \Psi \left(\frac{\Delta_{j+1/2}}{\Delta_{j-1/2}} \right) , \tag{1.145}$$

$$\Psi(r) = \begin{cases} 0 & (r \le 0) \\ r & (0 < r < 1) \\ 1 & (r \ge 1) \end{cases} , \tag{1.146}$$

and Ψ is referred to as the minmod limiter. In short, Equation (1.144) provides a modest extrapolation. [10]

Similarly, Equation (1.141) is replaced by

$$f_{j+1/2}^{*(\mathrm{R})} = \begin{cases} f_{j+1} & \Delta_{j+3/2}\,\Delta_{j+1/2} \le 0 \\ f_{j+1} - \dfrac{\Delta_{j+1/2}}{2} \min \left(1, \left| \dfrac{\Delta_{j+3/2}}{\Delta_{j+1/2}} \right| \right) & (\text{otherwise}) \end{cases} . \tag{1.147}$$

Equation (1.147 also gives a modest extrapolation from the right-hand side. When $f_j \le f_{j+1}$, we obtain the following inequality:

$$f_j \le f_{j+1/2}^{*(\mathrm{L})} \le f_{j+1/2}^{*\mathrm{R}} \le f_{j+1} , \tag{1.148}$$

[10] In addition to the minmod limiter, the superbee and other limiters are described by Hirsch (1981).

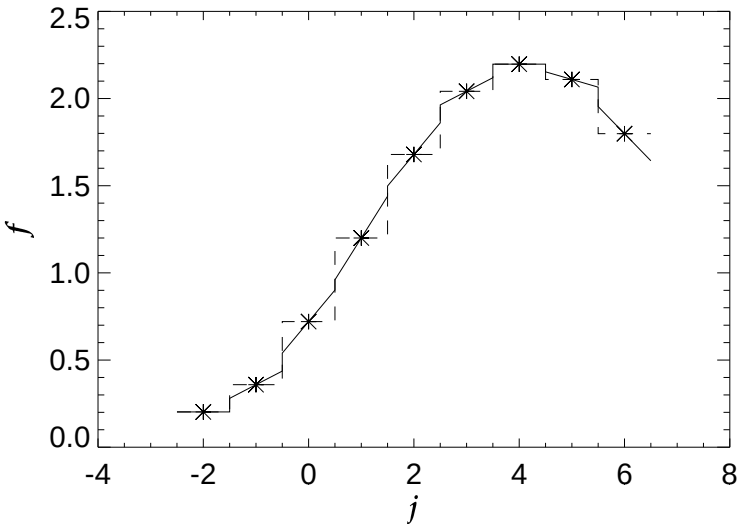

Figure 1.20: An example of extrapolation by Equations (1.144) and (1.147). The data are the same as those shown in Figure 1.18.

On the other hand, we obtain the following inequality:

$$f_j \geq f^{*(\mathrm{L})}_{j+1/2} \geq f^{*\mathrm{R}}_{j+1/2} \geq f_{j+1}\,, \tag{1.149}$$

when $f_j \geq f_{j+1}$. The monotonicity is preserved in both cases. Figure 1.20 gives an example of the stepwise linear function obtained by Equations (1.144) and (1.147), which demonstrates that this stepwise linear extrapolation preserves the monotonicity.

Using $f^{*(\mathrm{L})}_{j+1/2}$ and $f^{*(\mathrm{R})}_{j+1/2}$, we can construct the numerical flux as follows:

$$F^*_{j+1/2} = \frac{c}{2}\left(f^{*(\mathrm{L})}_{j+1/2} + f^{*(\mathrm{R})}_{j+1/2}\right) - \frac{|c|}{2}\left(\frac{f^{*(\mathrm{L})}_{j+1/2} - f^{*(\mathrm{R})}_{j+1/2}}{2}\right). \tag{1.150}$$

This numerical flux is constructed from the data of the upwind side and is of first-order accuracy. This numerical flux provides second-order accuracy in space.

Second-order accuracy in time is achieved by a two-stage method such as the predictor-corrector method for the ordinary differential equation. Suppose that we want to obtain the function f at $t = t_0 + \Delta t$ from that at $t = t_0$. In this case, we obtain the function f at $t = t_0 + \Delta t/2$ at the first stage. At the second stage, we obtain the function f at $t = t_0 + \Delta t$ using the numerical flux at $t = t_0 + \Delta/2$.

The following is the procedure used to solve the Burgers equation with second-order accuracy in space and time.

(1) Extrapolate the function while preserving the monotonicity.

$$f^{*(\mathrm{L})}_{j+1/2,n} = \begin{cases} f_{j,n} & (\Delta_{j+1/2,n}\,\Delta_{j-1/2,n} \leq 0) \\ f_{j,n} + \dfrac{\Delta_{j-1/2,n}}{2}\min\left(1, \left|\dfrac{\Delta_{j+1/2,n}}{\Delta_{j-1/2,n}}\right|\right) & (\text{otherwise}) \end{cases}, \tag{1.151}$$

$$f^{*(\mathrm{R})}_{j+1/2,n} = \begin{cases} f_{j+1,n} & (\Delta_{j+1/2,n}\,\Delta_{j+3/2,n} \le 0) \\ f_{j+1,n} - \dfrac{\Delta_{j+3/2,n}}{2} \min\left(1, \left|\dfrac{\Delta_{j+1/2,n}}{\Delta_{j+3/2,n}}\right|\right) & (\text{otherwise}) \end{cases}, \tag{1.152}$$

$$\Delta_{j-1/2,n} = f_{j,n} - f_{j-1,n}\,, \tag{1.153}$$
$$\Delta_{j+1/2,n} = f_{j+1,n} - f_{j,n}\,, \tag{1.154}$$
$$\Delta_{j+3/2,n} = f_{j+2,n} - f_{j+1,n}\,. \tag{1.155}$$

(2) Compute the numerical flux from $f^{(\mathrm{L})}_{j+1/2,n}$ and $f^{(\mathrm{R})}_{j+1/2,n}$.

$$F^{*}_{j+1/2,n} = \frac{1}{2}\left[\frac{\left(f^{*(\mathrm{L})}_{j+1/2,n}\right)^2}{2} + \frac{\left(f^{*(\mathrm{R})}_{j+1/2,n}\right)^2}{2}\right] - \frac{|\lambda|}{2}\left(f^{*(\mathrm{L})}_{j+1/2,n} - f^{*(\mathrm{R})}_{j+1/2,n}\right), \tag{1.156}$$

$$|\lambda| = \begin{cases} \dfrac{|f^{*(\mathrm{L})}_{j+1/2,n} + f^{*(\mathrm{R})}_{j+1/2,n}|}{2} & \left(\dfrac{|f^{*(\mathrm{L})}_{j+1/2,n} + f^{*(\mathrm{R})}_{j+1/2,n}}{2} \ge \varepsilon\right) \\ \varepsilon & (\text{otherwise}) \end{cases}, \tag{1.157}$$

$$\varepsilon = \max\left(0, \frac{f^{*(\mathrm{L})}_{j+1/2,n} - f^{*(\mathrm{R})}_{j+1/2,n}}{2}\right), \tag{1.158}$$

(3) Obtain $f_{j,n+1/2}$ using the numerical flux.

$$f_{j,n+1/2} = f_{j,n} + \frac{\Delta t}{2\Delta x}\left(F^{*}_{j+1/2,n} - F^{*}_{j-1/2,n}\right). \tag{1.159}$$

(4) Extrapolate $f_{j,n+1/2}$ as in step (1).

$$f^{*(\mathrm{L})}_{j+1/2,n+1/2} = \begin{cases} f_{j,n+1/2} & (\Delta_{j+1/2,n+1/2}\,\Delta_{j-1/2,n+1/2} \le 0) \\ f_{j,n+1/2} + \dfrac{\Delta_{j-1/2,n+1/2}}{2} \min\left(1, \left|\dfrac{\Delta_{j+1/2,n+1/2}}{\Delta_{j-1/2,n+1/2}}\right|\right) & (\text{otherwise}) \end{cases}, \tag{1.160}$$

$$f^{*(\mathrm{R})}_{j+1/2,n+1/2} = \begin{cases} f_{j+1,n+1/2} & (\Delta_{j+1/2,n+1/2}\,\Delta_{j+3/2,n+1/2} \le 0) \\ f_{j+1,n+1/2} - \dfrac{\Delta_{j+3/2,n+1/2}}{2} \min\left(1, \left|\dfrac{\Delta_{j+1/2,n+1/2}}{\Delta_{j+3/2,n+1/2}}\right|\right) & (\text{otherwise}) \end{cases}, \tag{1.161}$$

$$\Delta_{j-1/2,n+1/2} = f_{j,n+1/2} - f_{j-1,n+1/2}\,, \tag{1.162}$$
$$\Delta_{j+1/2,n+1/2} = f_{j+1,n+1/2} - f_{j,n+1/2}\,, \tag{1.163}$$
$$\Delta_{j+3/2,n+1/2} = f_{j+2,n+1/2} - f_{j+1,n+1/2}\,. \tag{1.164}$$

(5) Obtain the numerical flux from $f^{(\mathrm{L})}_{j+1/2,n+1}$ and $f^{(\mathrm{R})}_{j+1/2,n+1/2}$. [11]

$$F^*_{j+1/2,n+1/2} = \frac{1}{2}\left[\frac{\left(f^{*(\mathrm{L})}_{j+1/2,n+1/2}\right)^2}{2} + \frac{\left(f^{*(\mathrm{R})}_{j+1/2,n+1/2}\right)^2}{2}\right] - \frac{|\lambda|}{2}\left(f^{*(\mathrm{L})}_{j+1/2,n+1/2} - f^{*(\mathrm{R})}_{j+1/2,n+1/2}\right), \tag{1.165}$$

$$|\lambda| = \begin{cases} \frac{|f^{*(\mathrm{L})}_{j+1/2,n+1/2} + f^{*(\mathrm{R})}_{j+1/2,n+1/2}|}{2} & \left(\frac{|f^{*(\mathrm{L})}_{j+1/2,n+1/2} + f^{*(\mathrm{R})}_{j+1/2,n+1/2}}{2} \geq \varepsilon\right) \\ \varepsilon & (otherwise) \end{cases}, \tag{1.166}$$

$$\varepsilon = \max\left(0, \frac{f^{*(\mathrm{L})}_{j+1/2,n+1/2} - f^{*(\mathrm{R})}_{j+1/2,n+1/2}}{2}\right), \tag{1.167}$$

(6) Obtain $f_{j,n}$ using the numerical flux obtained in **(5)**.

$$f_{j,n} = f_{j,n} + \frac{\Delta t}{\Delta x}\left(F^*_{j+1/2,n+1/2} - F^*_{j-1/2,n+1/2}\right). \tag{1.168}$$

The above procedure is called MUSCL and can be applied to the hydrodynamical equations. When MUSCL is applied to the hydrodynamical equations, the question arises as to which variables should be extrapolated. For beginners, we recommend extrapolation of the density, velocity, and temperature ($T \equiv P/\rho$). Extrapolation of the pressure and extrapolation of the state variable $\boldsymbol{U}$ may cause difficulty near the shock front. Extrapolation of the Riemann invariants succeed in solving the magnetohydrodynamical equations (see, Fukuda and Hanawa 1999, ApJ, 517, p. 226). Extrapolation of the Riemann invariants also enables solution of the hydrodynamical equations. However, the extrapolation of the Riemann solver increases the computation cost and the complexity of the computation code. Thus, we do not recommend the extrapolation of the Riemann invariants for beginners.

1.7 Extension to Multi-dimensional Problems

In the previous sections, we restricted ourselves to the consideration of one-dimensional problems. In this section, we introduce numerical methods for solving the two-dimensional and three-dimensional hydrodynamical equations. We use Cartesian coordinates in this section. The use of cylindrical coordinates and spherical coordinates is discussed in Section 1.9.

The hydrodynamical equations for a multi-dimensional problem are expressed as follows:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla}\cdot(\rho \boldsymbol{v}) = 0, \tag{1.169}$$

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\boldsymbol{\nabla})\boldsymbol{v} + \frac{1}{\rho}\boldsymbol{\nabla}P = 0, \tag{1.170}$$

$$\frac{\partial}{\partial t}(\rho E) + \boldsymbol{\nabla}(\rho \boldsymbol{v} H) = 0. \tag{1.171}$$

[11] Although it is omitted here for simplicity, we should take the entropy condition into account, .

As in the case for the one-dimensional problem, they can be rewritten in conservation form as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}_x}{\partial x} + \frac{\partial \boldsymbol{G}_y}{\partial y} + \frac{\partial \boldsymbol{H}_z}{\partial z} = 0\,, \tag{1.172}$$

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \tag{1.173}$$

$$\boldsymbol{F} = \begin{pmatrix} \rho u \\ \rho u^2 \\ \rho uv \\ \rho uw \\ \rho H u \end{pmatrix}, \tag{1.174}$$

$$\boldsymbol{G} = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 \\ \rho vw \\ \rho H v \end{pmatrix}, \tag{1.175}$$

$$\boldsymbol{H} = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 \\ \rho H w \end{pmatrix}. \tag{1.176}$$

Here, u, v, and w denote the x-, y-, and z components of the velocity, respectively. The density is evaluated as

$$\rho_{i,j,k,n} \equiv \rho\,(x = i\,\Delta x,\, y = j\,\Delta y,\, z = k\,\Delta z,\, t = n\,\Delta t) \tag{1.177}$$

at the center of the small rectangular box having a volume of $\Delta x \times \Delta y \times \Delta z$. Similarly, each component of the velocity, E and H, is evaluated at the center of the rectangular box. The last subscript indicates the time, as in the case of the one-dimensional flow. For later convenience, we use the following notation:

$$\boldsymbol{U}_{i,j,k,n} \equiv \boldsymbol{U}\,(x = i\,\Delta x,\, y = j\,\Delta y,\, z = k\,\Delta x,\, t = n\,\Delta t)\,. \tag{1.178}$$

Then the difference equation for a multi-dimensional flow is expressed as

$$\begin{aligned} &\frac{\boldsymbol{U}_{i,j,k,n+1} - \boldsymbol{U}_{i,j,k,n}}{\Delta t} + \frac{\boldsymbol{F}^*_{i+1/2,j,k,n} - \boldsymbol{F}^*_{i-1/2,j,k,n+1}}{\Delta x} \\ &\qquad + \frac{\boldsymbol{G}^*_{i,j+1/2,k,n} - \boldsymbol{G}^*_{i,j-1/2,k,n}}{\Delta x} \\ , \qquad &\qquad + \frac{\boldsymbol{H}^*_{i,j,k+1/2,n} - \boldsymbol{H}^*_{i,j,k-1/2,n}}{\Delta x} = 0\,, \end{aligned} \tag{1.179}$$

where $\boldsymbol{F}^*_{i+1/2,j,k,n}$, $\boldsymbol{G}^*_{i,j+1/2,k,n}$, and $\boldsymbol{H}^*_{i,j,k+1/2,n}$ denote the numerical fluxes in the x-, y-, and z-directions, respectively. The numerical fluxes should of course be upwind fluxes.

First, we obtain the numerical flux in the x-direction, $\boldsymbol{F}_{i+1/2,j,k,n}$. Since the state vector $\boldsymbol{U}$ and the flux vector $\boldsymbol{F}$ have three components in the one-dimensional problem, the velocity matrix $\boldsymbol{A}$ has three rows and three columns. Consequently, there exist three eigenvalues, three right eigenvectors, and three left eigenvectors. The state and flux vectors ($\boldsymbol{U}$, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$) have five components in the three-dimensional hydrodynamical equations. Thus, we have five characteristic speeds, five right eigenvectors, and five left eigenvectors, which are derived as follows. First, we express each component of the flux vector as a function of the state:

$$\boldsymbol{F} = \left\{ \begin{array}{c} U_2 \\ (\gamma - 1)\left[U_5 - \dfrac{(U_2)^2 + (U_3)^2 + (U_4)^2}{U_1}\right] + \dfrac{(U_2)^2}{U_1} \\ \dfrac{U_2 U_3}{U_1} \\ \dfrac{U_2 U_4}{U_1} \\ \dfrac{U_2 U_5}{U_1} + (\gamma - 1)\left[U_5 - \dfrac{(U_2)^2 + (U_3)^2 + (U_4)^2}{U_1}\right]\dfrac{U_2}{U_1} \end{array} \right\}. \tag{1.180}$$

The velocity matrix ($\boldsymbol{A} \equiv \partial \boldsymbol{F}/\partial \boldsymbol{U}$) is then expressed as

$$\boldsymbol{A} = \left[\begin{array}{ccccc} 0 & 1 & 0 & 0 & 0 \\ \dfrac{(\gamma - 1)}{2} q - u^2 & (3 - \gamma)\, u & (1 - \gamma)\, v & (1 - \gamma)\, w & (\gamma - 1) \\ -uv & v & u & 0 & 0 \\ -uw & w & 0 & u & 0 \\ -\gamma u E + (\gamma - 1)\, uq & \gamma E - \dfrac{(\gamma - 1)}{2}(2u^2 + q) & (1 - \gamma)\, uv & (1 - \gamma)\, uw & \gamma u \end{array} \right], \tag{1.181}$$

$$q = u^2 + v^2 + w^2 . \tag{1.182}$$

This velocity matrix has five eigenvalues,

$$\lambda_1 = u - a , \tag{1.183}$$
$$\lambda_2 = u , \tag{1.184}$$
$$\lambda_3 = u , \tag{1.185}$$
$$\lambda_4 = u , \tag{1.186}$$
$$\lambda_5 = u + a , \tag{1.187}$$

where a denotes the speed of sound as in the case of the case of a one-dimensional flow. The

corresponding right eigenvectors are expressed as follows:

$$\boldsymbol{r}_1 = \begin{pmatrix} 1 \\ u - a \\ v \\ w \\ H - au \end{pmatrix}, \tag{1.188}$$

$$\boldsymbol{r}_2 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ -a \\ -wa \end{pmatrix}, \tag{1.189}$$

$$\boldsymbol{r}_3 = \begin{pmatrix} 0 \\ 0 \\ a \\ 0 \\ va \end{pmatrix}, \tag{1.190}$$

$$\boldsymbol{r}_4 = \begin{pmatrix} 1 \\ u \\ v \\ w \\ \frac{q}{2} \end{pmatrix}, \tag{1.191}$$

$$\boldsymbol{r}_5 = \begin{pmatrix} 1 \\ u + a \\ v \\ w \\ H + au \end{pmatrix}. \tag{1.192}$$

Similarly, the left eigenvectors are expressed as follows:

$$\boldsymbol{\ell}_1 = \left[\frac{1}{2}\left(b_1 + \frac{u}{a}\right), -\frac{1}{2}\left(\frac{1}{a} + b_2 u\right), -\frac{b_2 v}{2}, -\frac{b_2 w}{2}, \frac{b_2}{2}\right], \tag{1.193}$$

$$\boldsymbol{\ell}_2 = \left(\frac{w}{a}, 0, 0, -\frac{1}{c}, 0\right), \tag{1.194}$$

$$\boldsymbol{\ell}_3 = \left(-\frac{v}{a}, 0, \frac{1}{c}, 0, 0\right), \tag{1.195}$$

$$\boldsymbol{\ell}_2 = \left(\frac{w}{a}, 0, 0, -\frac{1}{c}, 0\right), \tag{1.196}$$

$$\boldsymbol{\ell}_4 = (1 - b_1, b_2 u, b_2 v, b_2 w, -b_2), \tag{1.197}$$

$$\boldsymbol{\ell}_5 = \left[\frac{1}{2}\left(b_1 - \frac{u}{a}\right), \frac{1}{2}\left(\frac{1}{a} - b_2 u\right), -\frac{b_2 v}{2}, -\frac{b_2 w}{2}, \frac{b_2}{2}\right], \tag{1.198}$$

where

$$b_1 = \frac{q}{2}\frac{\gamma - 1}{a^2}, \tag{1.199}$$

$$b_2 = \frac{\gamma - 1}{a^2}. \tag{1.200}$$

The right and left eigenvectors are normalized so that

$$\boldsymbol{\ell}_m \cdot \boldsymbol{r}_n \;=\; \delta_{m,n}\,. \tag{1.201}$$

These eigenvalues and eigenvectors are, of course, similar to those for the one-dimensional hydrodynamical equations. The two eigenvectors, the eigenvalues of which are $v \pm a$, are modified slightly to include the y- and z-components of the velocity. The eigenvalue u is degenerated three times to obtain two additional eigenvectors. The eigenvector $\boldsymbol{r}_4$ denotes the entropy flow, as in the one-dimensional hydrodynamical equations. The eigenvectors $\boldsymbol{r}_2$ and $\boldsymbol{r}_3$ denote the shear flows in which the z- and y-components of the velocity, respectively, change in the x-direction.

If we evaluate the eigenvalues and eigenvectors by the Roe average defined as

$$\bar{u} \;=\; \frac{\sqrt{\rho_{i+1,j,k}}u_{i+1,j,k} \,+\, \sqrt{\rho_{i,j,k}}u_i}{\sqrt{\rho_{i+1,j,k}} \,+\, \sqrt{\rho_{i,j,k}}}\,, \tag{1.202}$$

$$\bar{v} \;=\; \frac{\sqrt{\rho_{i+1,j,k}}v_{i+1,j,k} \,+\, \sqrt{\rho_{i,j,k}}v_{i,j,k}}{\sqrt{\rho_{i+1,j,k}} \,+\, \sqrt{\rho_{i,j,k}}}\,, \tag{1.203}$$

$$\bar{w} \;=\; \frac{\sqrt{\rho_{i+1,j,k}}w_{i+1,j,k} \,+\, \sqrt{\rho_{i,j,k}}w_{i,j,k}}{\sqrt{\rho_{i+1,j,k}} \,+\, \sqrt{\rho_{i,j,k}}}\,, \tag{1.204}$$

$$\bar{q} \;=\; \bar{u}^2 \,+\, \bar{v}^2 \,+\, \bar{w}^2\,, \tag{1.205}$$

$$\bar{H} \;=\; \frac{\sqrt{\rho_{i+1,j,k}}H_{i+1,j,k} \,+\, \sqrt{\rho_{i,j,k}}H_{i,j,k}}{\sqrt{\rho_{i+1,j,k}} \,+\, \sqrt{\rho_{i,j,k}}}\,, \tag{1.206}$$

$$\bar{a} \;=\; (\gamma \,-\, 1)\left(\bar{H} \,-\, \frac{\bar{q}}{2}\right), \tag{1.207}$$

then the velocity matrix satisfies the U property. Thus, the numerical flux is given by

$$\boldsymbol{F}^*_{i+1/2,j,k} \;=\; \frac{1}{2}\left(\boldsymbol{F}_{i+1,j,k} \,+\, \boldsymbol{F}_{i,j,k}\right) \,-\, \frac{1}{2}\,\boldsymbol{R}\,|\boldsymbol{\Lambda}|\,\boldsymbol{L}\,\left(\boldsymbol{U}_{i+1,j,k} \,-\, \boldsymbol{U}_{i,j,k}\right). \tag{1.208}$$

Equation (1.208) is rewritten as

$$\boldsymbol{F}^*_{j+1/2,n} \;=\; \frac{1}{2}\left[\boldsymbol{F}_{j+1,n} \,+\, \boldsymbol{F}_{j,n} \,-\, \sum_{k=1}^{5} |\lambda_k|\, w_k\, \boldsymbol{r}_k\right], \tag{1.209}$$

where

$$w_1 \;=\; \frac{1}{2\bar{a}}\left[\frac{P_{j+1} \,-\, P_j}{\bar{a}} \,-\, \bar{\rho}\,(v_{j+1} \,-\, v_j)\right], \tag{1.210}$$

$$w_2 \;=\; -\,\frac{\bar{\rho}}{\bar{a}}\,(w_{j+1} \,-\, w_j)\,, \tag{1.211}$$

$$w_3 \;=\; -\,\frac{\bar{\rho}}{\bar{a}}\,(v_{j+1} \,-\, v_j)\,, \tag{1.212}$$

$$w_4 \;=\; \rho_{j+1} \,-\, \rho_j \,-\, \frac{P_{j+1} \,-\, P_j}{\bar{a}^2}\,, \tag{1.213}$$

$$w_5 \;=\; \frac{1}{2\bar{a}}\left[\frac{P_{j+1} \,-\, P_j}{\bar{a}} \,+\, \bar{\rho}\,(v_{j+1} \,-\, v_j)\right], \tag{1.214}$$

because

$$\boldsymbol{U}_{j+1} \,-\, \boldsymbol{U}_{j+1} \;=\; \sum_{k=1}^{5} w_k \boldsymbol{r}_k\,, \tag{1.215}$$

for any $\boldsymbol{U}_j$ and $\boldsymbol{U}_{j+1}$.

The remaining numerical fluxes, $\boldsymbol{G}_{i,j+1/2,k,n}$ and $\boldsymbol{H}_{i,j,k+1/2,n}$, are obtained in a similar manner. However, we can obtain the corresponding formula by replacing u, v, and w in a cyclic manner. The details are omitted here in order to save space.

We strongly recommend using the same function (or subroutine) to compute $\boldsymbol{F}_{i+1/2,j,n}$, $\boldsymbol{G}_{i,j+1/2,k,n}$, and $\boldsymbol{H}_{i,j,k+1/2,n}$ when coding the program in C or in Fortran. If the arguments are changed appropriately, the function (or subroutine) can compute all of the numerical fluxes. Two or more functions that are essentially the same are not easy to handle.

By substituting the numerical fluxes into Equation (1.172), we obtain a new state vector, $\boldsymbol{U}_{i,j,k,n+1}$, which is of the first-order in space and time.

Next, we consider the Courant condition. The time step should be smaller in the three-dimensional flow than in the one-dimensional flow. If the time step is smaller than

$$\frac{1}{\Delta t} \geq \frac{\lambda_{x,max}}{\Delta x} + \frac{\lambda_{y,max}}{\Delta y} + \frac{\lambda_{z,max}}{\Delta z}\,, \tag{1.216}$$

$$\lambda_{x,max} = |v_x| + c_s\,, \tag{1.217}$$

$$\lambda_{y,max} = |v_y| + c_s\,, \tag{1.218}$$

$$\lambda_{z,max} = |v_z| + c_s\,, \tag{1.219}$$

we can obtain the stable solution safely. [12]

The solution of second-order accuracy in time is obtained by the following two-stage procedure:

1. Compute the numerical fluxes $\boldsymbol{F}^*_{i+1/2,j,k,n}$, $\boldsymbol{G}^*_{i,j+1/2,k,n}$, and $\boldsymbol{H}^*_{i,j,k+1/2,n}$ from the initial state $\boldsymbol{U}_{i,j,k,n}$.

2. Obtain the intermediate state $\boldsymbol{U}_{i,j,k,n+1/2}$, at $t = t_0 + \Delta t/2$ using the numerical flux.

3. Obtain the numerical fluxes $\boldsymbol{F}^*_{i+1/2,j,k,n+1/2}$, $\boldsymbol{G}^*_{i,j+1/2,k,n+1/2}$, and $\boldsymbol{H}^*_{i,j,k+1/2,n+1/2}$ from the intermediate state $\boldsymbol{U}_{i,j,k,n+1/2}$ at $t = t_0 + \Delta t/2$ so that the numerical fluxes are of first-order accuracy in space.

4. Obtain the new state $\boldsymbol{U}_{i,j,k,n+1}$ at $t = t_0 + \Delta t$ using the numerical flux obtained in the previous stage.

We have treated the three numerical fluxes separately, which is referred to as directional splitting. Directional splitting is widely used because it is easy to implement. However, directional splitting does have some disadvantages, as discussed below.

When the gradient of the density (or that of the velocity) is parallel to the coordinate, the flow is essentially one-dimensional and directional splitting is not problematic. However, when the gradient is inclined with respect to the coordinate, the directional splitting produces a spurious feature.

The weak points are relaxed if we take nearby cells into account when we interpolate and extrapolate the density and velocity. As shown in Section 1.6, the physical variables, such as the density and velocity, are extrapolated in order to achieve higher-order accuracy. The variables are extrapolated along the coordinate in directional splitting. If the surrounding cells are taken into account, the gradient will be improved.

The numerical fluxes for the two-dimensional hydrodynamical equations are derived easily from the numerical fluxes for the three- dimensional hydrodynamical equations.

[12] The time step can be taken to be slightly longer than that shown here. Here, a very safe criterion is given.

1.8 Inclusion of Gravity, Heating and Cooling

Thus far, we have ignored gravity in the hydrodynamical equations. However, gravity plays an important role in astrophysics. Gravity appears as a source term in the hydrodynamical equations.

If we take gravity, $\boldsymbol{g}$, into account, the equation of motion and the equation for energy conservation are rewritten as

$$\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v}\cdot\boldsymbol{\nabla})\,\boldsymbol{v} + \frac{1}{\rho}\boldsymbol{\nabla} P = \rho\boldsymbol{g}\,, \tag{1.220}$$

and

$$\frac{\partial}{\partial t}(\rho E) + \boldsymbol{\nabla}\,(\rho \boldsymbol{v} H) = \rho\boldsymbol{v}\cdot\boldsymbol{g}\,, \tag{1.221}$$

respectively. The right-hand side of Equations (1.220) and (1.221) are the source terms due to gravity. The gravitational acceleration, $\boldsymbol{g}$, is given by an external field (i.e., by an explicit function of the coordinates) or by the solution of the Poisson equation for a given density distribution. The Poisson equation is solved by iteration. In this textbook, we assume that the gravitational acceleration has already been obtained.

The right-hand sides of Equations (1.220) and (1.221) are added separately after solving the left-hand sides. In other words, we obtain the solution of the first-order in space and time as

$$\frac{\boldsymbol{U}_{j,n+1} - \boldsymbol{U}_{j,n}}{\Delta t} = \frac{\boldsymbol{F}^*_{j+1/2,n} - \boldsymbol{F}^*_{j-1/2,n}}{\Delta x} + \boldsymbol{S}_{j,n} \tag{1.222}$$

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ \rho\boldsymbol{g} \\ \rho\boldsymbol{g}\cdot\boldsymbol{v} \end{pmatrix}. \tag{1.223}$$

The solution of the second-order accuracy in time is obtained by adding the source term ($\boldsymbol{S}$) at each stage.

Inclusion of the gravity is easy, as shown above. However, a certain degree of care must be taken. First, the grid spacing, Δx, should be small, so that

$$|\boldsymbol{g}|\,\Delta x \ll a^2\,. \tag{1.224}$$

The left-hand side of Equation (1.224) denotes the potential energy difference over the grid spacing, which is equal to the potential energy change when the gas flows from a cell to its neighbor. The right-hand side of Eq. (1.224) denotes the square of the speed of sound, which is equal to the specific thermal energy. Thus, Equation (1.224) requires that the spatial change in the potential energy should be much smaller than the thermal energy. This is equivalent to the requirement that the grid spacing be much smaller than the pressure scale height. If Equation (1.224) is divided by $|\boldsymbol{g}|$, the right-hand denotes the pressure scale height. When the grid spacing is smaller than one-tenth the pressure scale height, gravity is taken into account fairly accurately. If the grid spacing is larger than half the pressure scale height, the accuracy of the solution is very limited. In addition, unnatural features may be observed due to the large gravity.

Inclusion of the gravity appears to place another constraint on the time step, Δt. Suppose that the initial velocity vanishes. The gas element moves by $g\,\Delta t^2/2$ in a time step. The movement should be smaller than the grid spacing, Δx. This gives the following condition:

$$\frac{g\,\Delta t^2}{2} \le \Delta x\,, \tag{1.225}$$

which is equivalent to

$$\Delta t \leq \sqrt{\frac{2}{g\,\Delta x}}\,\Delta x\,. \tag{1.226}$$

This constraint is automatically satisfied if the Courant condition and Equation (1.224) are satisfied simultaneously. In practice, for safety, we recommend lowering the CFL number slightly when gravity is included in the computation.

Heating and cooling by radiation and nuclear and thermal reactions appear as source terms in the hydrodynamical equations:

$$\frac{\partial}{\partial t}(\rho E) + \boldsymbol{\nabla}(\rho \boldsymbol{v} H) = \rho \boldsymbol{v}\cdot\boldsymbol{g} + \Gamma - \Lambda\,, \tag{1.227}$$

where Γ and Λ denote the heating and cooling rates per unit volume, respectively. When expressed as functions of density and temperature, they can be included easily. Heating by nuclear and chemical reactions can be expressed as a function of density and temperature. In addition, heating and cooling by radiation can be approximated as a function of density and temperature, when the system is optically thin. When the system is optically thick, the radiative transfer equations should be solved simultaneously. However, this is beyond the scope of this textbook.

1.9 Extension to the Cylindrical and Spherical Coordinates

Next, in this section, we introduce a numerical method by which to solve the hydrodynamical equations using cylindrical or spherical coordinates. When the hydrodynamical equations are expressed in cylindrical or spherical coordinates, we need to modify them to apply the upwind scheme.

Suppose that the density and velocity are independent of φ in cylindrical coordinates, $(r,\,\varphi,\,z)$, for simplicity. The mass conservation is then expressed as

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r) + \frac{\partial}{\partial z}(r\rho v_z) = 0\,. \tag{1.228}$$

The second term diverges from the axis ($r = 0$) and is difficult to treat in this form. To avoid the divergence, we rewrite the mass conservation as

$$\frac{\partial}{\partial t}(r\rho) + \frac{\partial}{\partial r}(r\rho v_r) + \frac{\partial}{\partial z}(r\rho v_z) = 0\,, \tag{1.229}$$

by multiplying by r. We can rewrite this differential equation in integral form as

$$\int_V \frac{\partial \rho}{\partial t}\,dV + \int_S \rho \boldsymbol{v}\cdot d\boldsymbol{S} = 0\,, \tag{1.230}$$

by integrating in the r- and z-directions. Here, dV denotes the volume integral and is equal to $r dr\, d\varphi\, dz$. The integral form (Gauss's law) denotes that the temporal change in mass within a volume is equal to the sum of the mass flux flowing into and out from the volume thorough the surface. Equation (1.229) is better than Equation (1.228) because the mass conservation is more clearly shown in this expression.

If we multiply the hydrodynamical equations by r, they are expressed as follows:

$$\frac{\partial}{\partial t}(r\boldsymbol{U}) + \frac{\partial}{\partial r}(r\boldsymbol{F}_r) + \frac{\partial}{\partial z}(r\boldsymbol{F}_z) = \boldsymbol{S}\,, \tag{1.231}$$

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho v_r \\ \rho v_\varphi \\ \rho v_z \\ \rho E \end{pmatrix}, \tag{1.232}$$

$$\boldsymbol{F}_r = \begin{pmatrix} \rho v_r \\ \rho {v_r}^2 + P \\ \rho v_r v_\varphi \\ \rho v_r v_z \\ \rho H v_r \end{pmatrix}, \tag{1.233}$$

$$\boldsymbol{F}_z = \begin{pmatrix} \rho v_z \\ \rho v_r v_z \\ \rho v_\varphi v_z \\ \rho {v_z}^2 + P \\ \rho H v_z \end{pmatrix}, \tag{1.234}$$

$$\boldsymbol{S} = \begin{pmatrix} 0 \\ \rho {v_\varphi}^2 + P \\ 0 \\ 0 \\ 0 \end{pmatrix}, \tag{1.235}$$

$$E = \frac{{v_r}^2 + {v_\varphi}^2 + {v_z}^2}{2} + \frac{1}{\gamma - 1}\frac{P}{\rho}, \tag{1.236}$$

$$H = \frac{{v_r}^2 + {v_\varphi}^2 + {v_z}^2}{2} + \frac{\gamma}{\gamma - 1}\frac{P}{\rho}. \tag{1.237}$$

Note that the extra source terms appear in this expression. The first source term denotes the centrifugal force, and the second source term is related to the pressure gradient because

$$\frac{\partial P}{\partial r} = \frac{1}{r}\frac{\partial}{\partial r}(rP) - \frac{P}{r}. \tag{1.238}$$

These source terms, as well as the gravity, should be taken into account in the cylindrical coordinates.

The state vector ($\boldsymbol{U}$) and the flux vectors ($\boldsymbol{F}_r$ and $\boldsymbol{F}_z$) are also multiplied by r. The radius, r, is evaluated at the cell surface when the numerical flux is computed. When we compute the numerical flux between the cells centered at $r = r_j$ and at $r = r_{j+1}$, the radius should be evaluated simply as $r = r_{j+1/2} \equiv (r_j + r_{j+1})/2$. Note that the value of r need not be evaluated at the upwind side. Thus, the numerical flux in the r-direction ($\boldsymbol{F}_r$) is evaluated by applying Roe's formula as given in Section 1.5 after replacing (v_x, v_y, v_z) with (v_r, v_φ, v_z).

Care must be taken in evaluating the rotation velocity v_φ near the axis, because it is small and proportional to the radius near the z-axis. However, the obtained value should be accurate because the centrifugal force is v_φ^2/r. Otherwise, the error is amplified by a factor of $1/r$. The best way to evaluate the rotation velocity is to assume that the angular velocity $\Omega \equiv v_\varphi/r$ is approximately constant and changes smoothly near the axis. When the flow is symmetric around the axis ($\partial/\partial\varphi = 0$), this is a fairly good approximation. Thus, the boundary condition

near the axis is given by

$$\frac{\partial \rho}{\partial r} = 0\,, \tag{1.239}$$
$$v_r = 0\,, \tag{1.240}$$
$$\frac{\partial \Omega}{\partial r} = 0\,, \tag{1.241}$$
$$\frac{\partial v_z}{\partial r} = 0\,. \tag{1.242}$$

This boundary condition is equivalent to assuming that ρ, Ω, and v_z are even with respect to the reversal of r, while v_r is odd.

Note that the third component of Equation (1.231) has no source term and hence the total angular momentum around the z-axis is conserved in this finite difference method. This is an advantage of the conservation form. The total angular momentum in the computation volume is conserved within the round-off error as long as the boundary conditions are appropriate.

It is possible to include the dependence on φ in the cylindrical coordinates. However, in practice, this is not possible near the z-axis because many cells are adjacent to each other on the axis of $r = 0$. Special care should be taken if the three-dimensional hydrodynamical equations are computed in cylindrical coordinates.

When spherical coordinates are applied, the mass conservation should be rewritten as

$$\frac{\partial}{\partial t}\left(r^2 \sin\theta\, \rho\right) + \frac{\partial}{\partial r}\left(r^2 \sin\theta\, \rho v_r\right) + \frac{\partial}{\partial \theta}\left(r^2 \sin\theta\, \rho v_\theta\right) + \frac{\partial}{\partial \varphi}\left(r^2 \sin\theta\, \rho v_\varphi\right) = 0\,. \tag{1.243}$$

Similarly, the other components of the hydrodynamical equations should be multiplied by a factor $r^2 \sin\theta$. Then, the source terms appear in the conservation form because of the centrifugal force and pressure. They can be treated similarly to the case of the cylindrical coordinates. Details are omitted to save space.

1.10 Boundary Conditions

The boundary condition, as well as the initial condition, is necessary for specifying a solution for a given partial differential equation. In this section, we introduce the discretization of the boundary condition.

The Dirichlet boundary condition and the Neumann boundary conditions are famous as mathematically well-defined boundary conditions. The former specifies the value of the variable on the boundary, while the latter specifies the gradient of the variable. In numerical simulations, we employ conventional boundary conditions other than well-defined boundary conditions in order to maintain the computation volume finite. First, we consider the Dirichlet and Neumann boundary conditions.

Equations (1.239) through (1.242), which are given on the boundary of $r = 0$, are equivalent to either the Dirichlet boundary condition (v_r) or the Neumann boundary condition (ρ, Ω, and v_z). For convenience, we assume that the radius can be negative. Then, Equations (1.239) through (1.242) are equivalent to

$$\rho(-r) = \rho(r)\,, \tag{1.244}$$
$$v_r(-r) = -\,v_r(r)\,, \tag{1.245}$$
$$\Omega(-r) = \Omega(r)\,, \tag{1.246}$$
$$P(-r) = P_r(r)\,. \tag{1.247}$$

Suppose that the grids are placed on

$$r_j = \left(j - \frac{1}{2}\right) \Delta r \,, \tag{1.248}$$

in the r-direction. Then, we have the following numerical boundary conditions:

$$\rho\left(r_0\right) = \rho\left(r_1\right) \,, \tag{1.249}$$
$$v_r\left(r_0\right) = -\, v_r\left(r_1\right) \,, \tag{1.250}$$
$$\Omega\left(r_0\right) = \Omega\left(r_1\right) \,, \tag{1.251}$$
$$P\left(r_0\right) = P_r\left(r_1\right) \,. \tag{1.252}$$

and

$$\rho\left(r_{-1}\right) = \rho\left(r_2\right) \,, \tag{1.253}$$
$$v_r\left(r_{-1}\right) = -\, v_r\left(r_2\right) \,, \tag{1.254}$$
$$\Omega\left(r_{-1}\right) = \Omega\left(r_2\right) \,, \tag{1.255}$$
$$P\left(r_{-1}\right) = P_r\left(r_2\right) \,. \tag{1.256}$$

The density, velocity and pressure in the region of negative r are obtained from those in the region of positive r by the symmetry with respect to $r = 0$. In other words, there is no boundary at $r = 0$ in a practical sense. The same procedure works for any symmetric boundary or reflection boundary.

As stated earlier, we want to limit the computation box to be a finite volume around stars or galaxies in numerical simulations. The surface of the computation box condition is artificial and is not limited by any physical laws. However, we want to reduce the artificial effects due to the boundaries as much as possible. For this purpose, several passive boundary conditions have been proposed.

The simplest (least expensive) condition is to place the reflection boundary at a very large distance from the center of the computation box. Although the waves reflected at the boundary are artificial and unnatural, the effects are rather limited if the density is extremely low near the boundary. If the grid spacing is larger near the boundary, then the artificial effects are further reduced because the boundary extends further. We can also add artificial damping in the narrow zone near the boundaries to reduce the reflected waves.

A more sophisticated boundary condition admits outgoing waves and inhibits incoming waves. This boundary condition is called the radiation boundary condition. Variations in density, velocity and pressure can be decomposed into waves, as described in previous sections. Thus, it is possible to exclude only incoming waves. However, it is not possible to perfectly cancel out a specific component of waves because the waves are nonlinear and the components are coupled with each other.

1.11 Extension to MHD Equations

The magnetohydrodynamical (MHD) equations can also be solved by the upwind scheme. The magnetohydrodynamical equations take the magnetic force and the induction of the magnetic

field into account and are expressed as follows:

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \boldsymbol{v}) = 0\,, \tag{1.257}$$

$$\rho \left\{ (\frac{\partial \boldsymbol{v}}{\partial t} + (\boldsymbol{v} \cdot \boldsymbol{\nabla})\, \boldsymbol{v} \right\} + \boldsymbol{\nabla} P + \frac{(\boldsymbol{\nabla} \times \boldsymbol{B}) \times \boldsymbol{B}}{4\pi} = 0\,, \tag{1.258}$$

$$\frac{\partial \boldsymbol{B}}{\partial t} - \boldsymbol{\nabla} \times (\boldsymbol{v} \times \boldsymbol{B}) = 0\,, \tag{1.259}$$

$$\frac{\partial}{\partial t} (\rho E) + \boldsymbol{\nabla} (\rho H) = 0\,, \tag{1.260}$$

$$E = \frac{|\boldsymbol{v}|^2}{2} + \frac{1}{\gamma - 1} \frac{P}{\rho} + \frac{|\boldsymbol{B}|^2}{8\pi\rho}\,, \tag{1.261}$$

$$H = \frac{|\boldsymbol{v}|^2}{2} + \frac{\gamma}{\gamma - 1} \frac{P}{\rho} + \frac{|\boldsymbol{B}|^2}{4\pi\rho}\,, \tag{1.262}$$

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} = 0\,. \tag{1.263}$$

The MHD equations are more complicated than the hydrodynamical equations. First of all, they have eight components, and we need to solve $\boldsymbol{B}$ as well as ρ, P, and $\boldsymbol{v}$. Second, they are associated with the constraint $\boldsymbol{\nabla} \cdot \boldsymbol{B} = 0$ and only seven of these components are independent. Moreover, the MHD waves can be degenerate, i.e., the phase velocity of an MHD wave may coincide with that of another. Even when the waves are degenerate, the eigenvectors should be independent of one other. Fortunately, all of these technical complexities have been resolved. We can solve the one-dimensional MHD equation as follows.

First, we rewrite the one-dimensional MHD equation into conservation form as follows:

$$\frac{\partial \boldsymbol{U}}{\partial t} + \frac{\partial \boldsymbol{F}}{\partial x} = 0\,, \tag{1.264}$$

where

$$\boldsymbol{U} = \begin{pmatrix} \rho \\ \rho v_x \\ \rho v_y \\ \rho v_z \\ B_y \\ B_z \\ \rho E \end{pmatrix}, \tag{1.265}$$

$$\boldsymbol{F} = \begin{pmatrix} \rho v_x \\ \rho v_x^2 + P + \dfrac{B_y^2 + B_z^2 - B_x^2}{8\pi} \\ \rho v_x v_y - \dfrac{B_x B_y}{4\pi} \\ \rho v_x v_z - \dfrac{B_x B_z}{4\pi} \\ v_x B_y - v_y B_x \\ v_x B_z - v_z B_x \\ \rho H v_x - \dfrac{B_x (B_x v_x + B_y v_y + B_z v_z)}{4\pi} \end{pmatrix}. \tag{1.266}$$

The numerical flux is expressed as

$$\boldsymbol{F}_{j+1/2}{}^{*} \;=\; \frac{1}{2}\left(\boldsymbol{F}_{j+1} \,+\, \boldsymbol{F}_{j}\right) \,-\, \sum_{k} \delta w_k \;|\lambda_k|\; \boldsymbol{r}_k \,, \tag{1.267}$$

where δw_k, λ_k, and $\boldsymbol{r}_k$ denote the amplitude, the phase velocity, and the eigenvector of the k-th eigenmode. For later convenience, we use the suffix 0 for the entropy wave, the suffix $\mathrm{A}_{\pm}$ for the Alfvén waves, the suffix $\mathrm{f}_{\pm}$ for the fast waves, and the suffix $\mathrm{s}_{\pm}$ for the slow waves.

The eigenvalues are denoted as follows:

$$\lambda_0 \;=\; \bar{v}_x \,, \tag{1.268}$$
$$\lambda_{\mathrm{A}+} \;=\; \bar{v}_x \,+\, b_x \,, \tag{1.269}$$
$$\lambda_{\mathrm{A}-} \;=\; \bar{v}_x \,-\, b_x \,, \tag{1.270}$$
$$\lambda_{f+} \;=\; \bar{v}_x \,+\, c_f \,, \tag{1.271}$$
$$\lambda_{f-} \;=\; \bar{v}_x \,-\, c_f \,, \tag{1.272}$$
$$\lambda_{s+} \;=\; \bar{v}_x \,+\, c_s \,, \tag{1.273}$$
$$\lambda_{s-} \;=\; \bar{v}_x \,-\, c_s \,, \tag{1.274}$$

where

$$\bar{\rho} \;=\; \sqrt{\rho_{j+1}\rho_j} \,, \tag{1.275}$$
$$\bar{v}_x \;=\; \frac{\sqrt{\rho_{j+1}}\, v_{x,j+1} \,+\, \sqrt{\rho_j}\, v_{x,j}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \,, \tag{1.276}$$
$$\bar{v}_y \;=\; \frac{\sqrt{\rho_{j+1}}\, v_{y,j+1} \,+\, \sqrt{\rho_j}\, v_{y,j}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \,, \tag{1.277}$$
$$\bar{v}_z \;=\; \frac{\sqrt{\rho_{j+1}}\, v_{z,j+1} \,+\, \sqrt{\rho_j}\, v_{z,j}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \,, \tag{1.278}$$
$$\bar{B}_y \;=\; \frac{\sqrt{\rho_{j+1}}\, B_{y,j} \,+\, \sqrt{\rho_j}\, B_{y,j+1}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \,, \tag{1.279}$$
$$\bar{B}_z \;=\; \frac{\sqrt{\rho_{j+1}}\, B_{z,j} \,+\, \sqrt{\rho_j}\, B_{z,j+1}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \,, \tag{1.280}$$
$$\bar{H} \;=\; \frac{\sqrt{\rho_{j+1}}\, H_{j+1} \,+\, \sqrt{\rho_j}\, H_j}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \tag{1.281}$$
$$\bar{P} \;=\; \frac{\sqrt{\rho_{j+1}}\, P_j \,+\, \sqrt{\rho_j}\, P_{j+1}}{\sqrt{\rho_{j+1}} \,+\, \sqrt{\rho_j}} \tag{1.282}$$

The eigenvectors are expressed as follows:

$$\delta b^2 = \frac{\gamma - 1}{\gamma - 2} \frac{(B_{y,j+1} - B_{y,j})^2 + (B_{z,j+1} - B_{z,j})^2}{8\pi (\sqrt{\rho_{j+1}} + \sqrt{\rho_j})^2}, \tag{1.283}$$

$$a^2 = (\gamma - 1) \left(\bar{H} - \frac{\bar{v}_x^2 + \bar{v}_y^2 + \bar{v}_z^2}{2} + \frac{B_x^2 + \bar{B}_y^2 + \bar{B}_z^2}{4\pi\rho} - \delta b^2 \right), \tag{1.284}$$

$$a_*^2 = (\gamma - 1) \left(\bar{H} - \frac{\bar{v}_x^2 + \bar{v}_y^2 + \bar{v}_z^2}{2} - \delta b^2 \right), \tag{1.285}$$

$$- (\gamma - 2) \left(\frac{B_x^2 + \bar{B}_y^2 + \bar{B}_z^2}{4\pi\rho} \right), \tag{1.286}$$

$$b_x = \frac{|B_x|}{\sqrt{4\pi\bar{\rho}}} \tag{1.287}$$

$$c_{f,s}^2 = \frac{a_*^2 \pm \sqrt{a_*^4 - 4a^2 b_x^2}}{2}, \tag{1.288}$$

$$\boldsymbol{r}_0 = \begin{pmatrix} 1 \\ \bar{v}_x \\ \bar{v}_y \\ \bar{v}_z \\ 0 \\ 0 \\ \frac{(\bar{\boldsymbol{v}})^2}{2} + \delta b^2 \end{pmatrix}, \tag{1.289}$$

$$\boldsymbol{r}_{\mathrm{A}+} = \begin{pmatrix} 0 \\ 0 \\ -\beta_z \operatorname{sgn}(B_x) \\ \beta_y \operatorname{sgn}(B_x) \\ \beta_z \sqrt{4\pi/\bar{\rho}} \\ -\beta_y \sqrt{4\pi/\bar{\rho}} \\ -(\beta_z \bar{v}_y - \beta_y \bar{v}_z) \operatorname{sgn}(B_x) \end{pmatrix}, \tag{1.290}$$

$$\boldsymbol{r}_{\mathrm{A}-} = \begin{pmatrix} 0 \\ 0 \\ \beta_z \operatorname{sgn}(B_x) \\ -\beta_y \operatorname{sgn}(B_x) \\ \beta_z \sqrt{4\pi/\bar{\rho}} \\ -\beta_y \sqrt{4\pi/\bar{\rho}} \\ (\beta_z \bar{v}_y - \beta_y \bar{v}_z) \operatorname{sgn}(B_x) \end{pmatrix}, \tag{1.291}$$

and

$$\boldsymbol{r}_{f+} = \begin{pmatrix} \alpha_f \\ \alpha_f\,(\bar{v}_x + c_f) \\ \alpha_f \bar{v}_y - \alpha_s\,\beta_y\,b_x\,\mathrm{sgn}(B_x) \\ \alpha_f \bar{v}_z - \alpha_s\,\beta_z\,b_x\,\mathrm{sgn}(B_x) \\ \alpha_s\,\beta_y\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_s\,\beta_z\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_f \left\{ \dfrac{(\bar{\boldsymbol{v}})^2}{2} + \delta b^2 + c_f \bar{v}_x + \dfrac{c_f^2}{\gamma - 1} \right. \\ \left. + \dfrac{\gamma - 2}{\gamma - 1}(c_f^2 - a^2) \right\} - \alpha_s b_x \mathrm{sgn}(B_x)(\beta_y \bar{v}_y + \beta_z \bar{v}_z) \end{pmatrix}, \tag{1.292}$$

$$\boldsymbol{r}_{f-} = \begin{pmatrix} \alpha_f \\ \alpha_f\,(\bar{v}_x - c_f) \\ \alpha_f \bar{v}_y + \alpha_s\,\beta_y\,b_x\,\mathrm{sgn}(B_x) \\ \alpha_f \bar{v}_z + \alpha_s\,\beta_z\,b_x\,\mathrm{sgn}(B_x) \\ \alpha_s\,\beta_y\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_s\,\beta_z\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_f \left\{ \dfrac{(\bar{\boldsymbol{v}})^2}{2} + \delta b^2 - c_f \bar{v}_x + \dfrac{c_f^2}{\gamma_t - 1} \right. \\ \left. + \dfrac{\gamma - 2}{\gamma - 1}(c_f^2 - a^2)) \right\} + \alpha_s b_x \mathrm{sgn}(B_x)(\beta_y \bar{v}_y + \beta_z \bar{v}_z) \end{pmatrix}, \tag{1.293}$$

$$\boldsymbol{r}_{s+} = \begin{pmatrix} \alpha_s \\ \alpha_s\,(\bar{v}_x + c_s) \\ \alpha_s \bar{v}_y + \alpha_f\,\beta_y\,a\,\mathrm{sgn}(B_x) \\ \alpha_s \bar{v}_z + \alpha_f\,\beta_z\,a\,\mathrm{sgn}(B_x) \\ -\dfrac{\alpha_f\,\beta_y\,a^2\,\sqrt{4\pi}}{c_f\sqrt{\bar{\rho}}} \\ -\dfrac{\alpha_f\,\beta_z\,a^2\,\sqrt{4\pi}}{c_f\sqrt{\bar{\rho}}} \\ \alpha_s\,\beta_z\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_s \left\{ \dfrac{(\bar{\boldsymbol{v}})^2}{2} + \delta b^2 + c_s \bar{v}_x + \dfrac{c_s^2}{\gamma - 1} \right. \\ \left. + \dfrac{\gamma - 2}{\gamma - 1}(c_s^2 - a^2) \right\} + \alpha_f a \mathrm{sgn}(B_x)(\beta_y \bar{v}_y + \beta_z \bar{v}_z) \end{pmatrix}, \tag{1.294}$$

$$\boldsymbol{r}_{s+} = \begin{pmatrix} \alpha_s \\ \alpha_s\,(\bar{v}_x - c_s) \\ \alpha_s \bar{v}_y - \alpha_f\,\beta_y\,a\,\mathrm{sgn}(B_x) \\ \alpha_s \bar{v}_z - \alpha_f\,\beta_z\,a\,\mathrm{sgn}(B_x) \\ -\dfrac{\alpha_f\,\beta_y\,a^2\,\sqrt{4\pi}}{c_f\sqrt{\bar{\rho}}} \\ -\dfrac{\alpha_f\,\beta_z\,a^2\,\sqrt{4\pi}}{c_f\sqrt{\bar{\rho}}} \\ \alpha_s\,\beta_z\,c_f\,\sqrt{4\pi/\bar{\rho}} \\ \alpha_s \left\{ \dfrac{(\bar{\boldsymbol{v}})^2}{2} + \delta b^2 - c_s \bar{v}_x + \dfrac{c_s^2}{\gamma - 1} \right. \\ \left. + \dfrac{\gamma - 2}{\gamma - 1}(c_s^2 - a^2) \right\} - \alpha_f a \mathrm{sgn}(B_x)(\beta_y \bar{v}_y + \beta_z \bar{v}_z) \end{pmatrix}, \tag{1.295}$$

where

$$\beta_y = \frac{\bar{B}_y}{\sqrt{\bar{B}_y^2 + \bar{B}_z^2}}, \tag{1.296}$$

$$\beta_z = \frac{\bar{B}_z}{\sqrt{\bar{B}_y^2 + \bar{B}_z^2}}, \tag{1.297}$$

$$\beta_y^2 + \beta_z^2 = 1, \tag{1.298}$$

$$\alpha_f^2 + \frac{b_x^2}{c_f^2}\alpha_s^2 = 1. \tag{1.299}$$

The wave amplitudes are given as follows:

$$\delta w_{\mathrm{A}+} = \frac{1}{2}\left[-\bar{\rho}(\beta_z \Delta v_y - \beta_y \Delta v_z)\mathrm{sgn}(B_x) + \sqrt{\frac{\bar{\rho}}{4\pi}}(\beta_z \Delta B_y - \beta_y \Delta B_z)\right], \tag{1.300}$$

$$\delta w_{\mathrm{A}-} = \frac{1}{2}\left[\bar{\rho}(\beta_z \Delta v_y - \beta_y \Delta v_z)\mathrm{sgn}(B_x) + \sqrt{\frac{\bar{\rho}}{4\pi}}(\beta_z \Delta B_y - \beta_y \Delta B_z)\right], \tag{1.301}$$

$$\delta w_{\mathrm{f}+} + \delta w_{\mathrm{f}-} = \frac{\alpha_f}{c_f^2}\left(\Delta P + \frac{\tilde{B}_y \Delta B_y + \tilde{B}_z \Delta B_z}{4\pi}\right) \tag{1.302}$$

$$+ \left\{\frac{\alpha_s}{a^2 c_f}\left[(\gamma - 1)c_s^2 - (\gamma - 2a^2\right]\sqrt{4\pi\bar{\rho}} \tag{1.303}$$

$$+ (\gamma - 2)\sqrt{\bar{B}_y^2 + \bar{B}_z^2}\frac{\alpha_f}{c_f^2}\right\}\frac{\beta_y \Delta B_y + \beta_z \Delta B_z}{4\pi}, \tag{1.304}$$

$$\delta w_1 - \delta w_7 = \frac{\alpha_f}{c_f}\bar{\rho}\Delta v_x - \frac{\alpha_s c_s}{c_f a}\mathrm{sgn}(B_x)\bar{\rho}(\beta_y \Delta v_y + \beta_z \Delta v_z), \tag{1.305}$$

$$\delta w_{\mathrm{s}+} + \delta w_{s-} = \frac{\alpha_s}{a^2}\left(\Delta P + \frac{\tilde{B}_y \Delta B_y + \tilde{B}_z \Delta B_z}{4\pi}\right) \tag{1.306}$$

$$+ \left\{\alpha_f\left[\frac{\gamma - 2}{c_f} - (\gamma - 1)\frac{c_f}{a^2}\right]\sqrt{4\pi\bar{\rho}}\right. \tag{1.307}$$

$$\left. + (\gamma - 2)\sqrt{\bar{B}_y^2 + \bar{B}_z^2}\frac{\alpha_s}{a^2}\right\}\frac{\beta_y \Delta B_y + \beta_z \Delta B_z}{4\pi}, \tag{1.308}$$

$$\delta w_{\mathrm{s}+} - \delta w_{\mathrm{s}-} = \frac{\alpha_s b_x}{c_f a}\bar{\rho}\Delta v_x + \frac{\alpha_f}{a}\mathrm{sgn}(B_x)\bar{\rho}(\beta_y \Delta v_y + \beta_z \Delta v_z), \tag{1.309}$$

$$\delta w_0 = \Delta\rho - \alpha_f(\delta w_1 + \delta w_7) - \alpha_s(\delta w_3 + \delta w_5) \tag{1.310}$$

These eigenvalues and eigenvectors satisfy the U property, as follows:

$$\boldsymbol{U}_{j+1} - \boldsymbol{U}_j = \sum_k \delta w_k \, \boldsymbol{r}_k, \tag{1.311}$$

$$\boldsymbol{F}_{j+1} - \boldsymbol{F}_j = \sum_k \delta w_k \, \lambda_k \, \boldsymbol{r}_k. \tag{1.312}$$

This type of numerical flux for the MHD equation was first given by Brio and Wu (1988). Parameters such as α_f were introduced by Ryu and Jones (1995) to handle the degeneracy. The numerical flux was modified to satisfy the U property given in Cargo and Gallice (1997).

1.12 Some Other Numerical Schemes

Although in this chapter we described only Roe's method in detail, there are a number of other popular schemes. This section briefly described these schemes, with emphasis on their advantages.

Before introducing specific methods, we shall discuss the qualities of a good numerical scheme. A perfect scheme can solve any problem stably with a high accuracy in a short time. Of course, perfect schemes do not exist. Perhaps perfect schemes have not yet been discovered because of the existence of contradicting requirements. For example, high-accuracy schemes tend to sacrifice short computation time. Robustness, which guarantees the ability to solve any problem, also requires greater computation cost in order to handle extreme conditions. We are often forced to compromise between two competing factors. Thus, a particular method recommended by a textbook or a paper should be regarded as being a good method for solving the particular problem in which the authors are interested and should not be considered to be good for all problems. As computer performance increases, the computation time for a specific problem is reduced. In the near future, some methods that are currently considered to be impractical may become practical. At present, however, a good method is only good for a particular type of problem.

Some good methods pursue higher-order accuracy, i.e., third-order or fourth-order accuracy. The Piecewise Parabolic Method (PPM) approximates the density, velocity, and temperature by piecewise parabolic functions in a cell and achieves third-order accuracy in space. Other methods employ the Runge-Kutta method, which is often used for solving ordinary differential equations, to achieve fourth-order accuracy in time. These schemes reduce truncation errors in the region where the density and velocity change smoothly.

Another method has been developed in order to capture the discontinuity more sharply. The Godunov method gives the exact solution for the initial density and velocity and regards the average over a cell as the solution. This method succeeds in treating a strong shock wave, in front of which the density and velocity change dramatically. The Cubic InterPolation (CIP) method proposed by Yabe and his collaborators is designed to capture the contact discontinuity more sharply. The CIP method succeeds in solving the problem of a solid body floating in liquid.

The reduction of computation time is another consideration when developing a new scheme. The HLL scheme reduces the time required to compute numerical flux by taking only the maximum and minimum phase speeds into account. This simplification reduces the computation time appreciably without reducing the quality of the solution greatly. If the computation time per cell is reduced, we can increase the number of cells and improve the resolution when the total computation time is fixed.

Generally, computation load is reduced when accuracy is reduced. Thus, it is not recommended to pursue excessive quality for unimportant regions. It is important to evaluate the specific requirements of the evaluation method before starting numerical simulations. Although robust code, which can manage any problem, provides good results, it consumes a great deal of computation time.

A number of computation codes for numerical simulations of astrophysical problems are available online. ZEUS, which was developed by J. Stone, is one such code. Another is CANS, which was developed by R. Matsumoto and his collaborators. The latter code is used for this winter school and can be downloaded from `http://www.astro.phys.s.chiba-u.ac.jp/netlab/pub/`. Both Japanese and English guide books are available for the CANS. The Japanese guide book was written by H. Hanayama, and the English guide book was translated from the Japanese by

S. Miyaji (Chapter 3).

For further study, we recommend "Riemann Solvers And Numerical Methods for Fluid Dynamics: A Practical Introduction" by E. Toro and "Numerical Computation of Internal and External Flows" by C. Hirsch.

Chapter 2

Exercises

2.1 Usage of the example package `scalar`

In this section, we will explain how to use the package for solving a scalar equation. The content of this package is as follows:

```
# ls scalar
Makefile    anime.pro   main.f      pldt.pro    pldtps.pro    rddt.pro
```

The program is written in Fortran that is one of the most popular programing language in the field of astronomical simulations. It is contained in the file 'main.f' in this example.

2.1.1 Compilation and execution of the program

Before executing a program, we need to 'compile' it – change a format of the program from a human readable one into a machine executable one. After moving to the directory "`scalar/`", execute the UNIX command '`make`'. Then, the program will be executed after a compilation. If succeed, you will find several new files, `main.o`, `a.out` and `out.dat` in this directory. The file `main.o` is an 'object' file corresponding to 'main.f', and the file 'a.out' is an 'executable' file. The result of the simulation is contained in the output data file 'out.dat'.

```
# cd scalar
# make
f77   -c -o main.o main.f
main.f:
 MAIN:
f77 -o a.out main.o
./a.out
  write    step=       0 time= 0.000E+00
  write    step=      50 time= 0.125E+02
  write    step=     100 time= 0.250E+02
   ### normal stop ###
# ls
Makefile    anime.pro   main.o      pldt.pro     rddt.pro
a.out*      main.f      out.dat     pldtps.pro
```

2.1.2 Output data file (out.dat)

You can read the content of the 'out.dat' file by using an editor or an appropriate UNIX command (e.g. `more`, `less`, `head` etc.) since it is written in a human readable format. The following is an example of the content. The first line indicates the spatial size (`jx`) of the data array and the number of outputs (`nx`) in the temporal sequence of the simulation. In this example, there are 3 sets of arrays with length of 100. The next line is for the 'time' information of the first data set. Here the item '0' and '0.00' correspond to the step (`ns`) and the time (`time`), respectively, of the first data set in the output. From the next to the 102nd line, the data is placed. The left and right column indicates the spatial coordinate (`x`) and the value (`u`) of the simulation result. The next data set starts from the 103rd line in the same order, and so on. This output format is defined at the 53, 55, and 59th lines in the Fortran program file '`main.f`'.

```
# head out.dat
  100,    3
    0,  0.00
  1.0, 1.0000000
  2.0, 1.0000000
  3.0, 1.0000000
  ......
```

2.1.3 Visualization of a result

We usually use a special software for the visualization of the simulation results. Here we introduce "IDL" that is one of such commercial (expensive!) softwares and is very popular in astronomical data analysis both for simulations and observations.

Startup of IDL (idl)

To startup IDL, type `idl`.

```
# idl
```

Then, it starts as follows:

```
IDL Version ....
Installation number: XXXXX.
Licensed for use by: XXXXX

IDL>
```

You can enter the IDL commands after its prompt "`IDL>`". You may also run an IDL program.

Loading the data into the IDL session (.r rddt)

To load the simulation data into the IDL session, use the IDL program `rddt.pro` as follows:

```
IDL> .r rddt
```

After this, you can refer to, process, and visualize the data in the IDL session. Here "`.r`" means

"run".

Plot of the data (.r pldt)

To plot the data, use the IDL program `pldt.pro` as follows:

```
IDL> .r pldt
```

The result will be like Fig 2.1

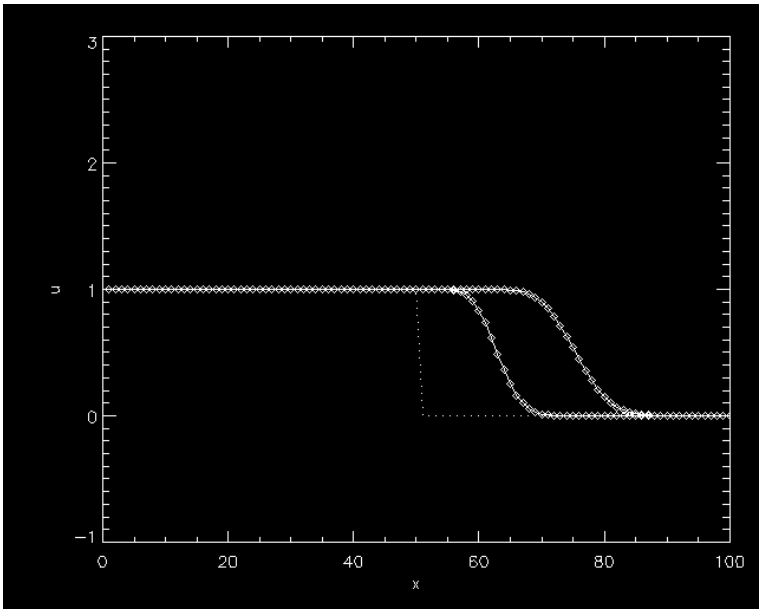

Figure 2.1: Plot of the simulation results in the 'scalar' package

Animation of the simulation results (.r anime)

To make an animation of the simulation results, use the IDL program `anime.pro` as follows:

```
IDL> .r anime
```

As a result of this IDL program, there appears a new window showing an animation. Note that an error occurs if you try to open another animation window simultaneously. Keep only one window.

Finish an IDL session (exit)

To finish an IDL session, type "`exit`" after the IDL prompt.

```
IDL> exit
```

2.1.4 Modification of the program

Change the hydrodynamic solver

The main solver in the example package is written with the upwind algorithm The Fortran statements of the algorithm is in the 72nd to 88th lines of 'main.f'. You can change it by modyfing these lines.

```
c-------------------------------------------------------------------------|
c     solve equation
c
c                                                       upwind - start
>>>
      do j=1,jx-1
         f(j)=0.5*(cs*(u(j+1)+u(j))-abs(cs)*(u(j+1)-u(j)))
      enddo

      f(jx)=f(jx-1)

      do j=2,jx-1
         u(j)=u(j)-dt/dx*(f(j)-f(j-1))
      enddo

      u(1)=u(2)
      u(jx)=u(jx-1)
c                                                       upwind - end   >>>
```

Change the number of mesh points (jx)

The number of mesh points is defined by the value of the variable `jx` at the 5th line of the program. The spatial resolution of the simulation can be controled by modyfing this part.

```
      parameter (jx=100)
```

Change the finishing step, the output settings (nstop, nskip)

The finishing step and interval step of the output are defined by the value of the variables `nstop` and `nskip`, respectively.

```
c   time control parameters

      nstop=100
      nskip = 50
```

Change the interval of the temporal stepping (safety)

The interval of the temporal stepping is determined by the CFL condition – a stability condition corresponding to each algorithm. This condition gives only an upper limit for the temporal interval. So we usually determine it by giving a 'safety number' (`safety`) less than unity. By changing this value, the stability, quality, and cost of simulations can be controlled.

```
c     obtain time spacing
      safety=0.25
```

2.1.5 Appendix

Sample Fortran program, main.f

```
c=======================================================================|
c     array definitions
c=======================================================================|
      implicit real*8 (a-h,o-z)
      parameter (jx=100)
      dimension x(1:jx),u(1:jx),f(1:jx)
c=======================================================================|
c     prologue
c=======================================================================|
c   time control parameters
      nstop=100
      nskip = 50
c-----------------------------------------------------------------------|
c  initialize counters
      time  = 0.0
      ns    = 0
      nx = nstop/nskip+1
c-----------------------------------------------------------------------|
c     Set initial condition
c-----------------------------------------------------------------------|
      pi=4.*atan(1.0)
c  grid
      dx=1.0
      x(1)=dx
      do j=1,jx-1
         x(j+1)=x(j)+dx
      enddo
c
c  variable
      do j=1,jx/2
         u(j)= 1.0
      enddo
      do j=jx/2+1,jx
         u(j)= 0.0
      enddo
c
c  velocity
      cs=1.0
c-----------------------------------------------------------------------|
c     Output initial condition
c
      write(6,103) ns,time
 103  format (1x,' write    ','step=',i8,' time=',e10.3)
      open(unit=10,file='out.dat',form='formatted')
      write(10,100) jx,nx
 100  format(i5,',',i5)
      write(10,101) ns,time
```

```
 101  format (i5,',',f6.2)
      do j=1,jx
         write(10,102) x(j),u(j)
      enddo
 102  format(f5.1,',',f10.7)
c=======================================================================|
c     time integration
c=======================================================================|
1000  continue
      ns = ns+1
c-----------------------------------------------------------------------|
c     obtain time spacing
      safety=0.25
      dt=safety*dx/cs
      time=time+dt
c-----------------------------------------------------------------------|
c     solve equation
c
c                                                     upwind - start >>>
      do j=1,jx-1
         f(j)=0.5*(cs*(u(j+1)+u(j))-abs(cs)*(u(j+1)-u(j)))
      enddo
      f(jx)=f(jx-1)

      do j=2,jx-1
         u(j)=u(j)-dt/dx*(f(j)-f(j-1))
      enddo
      u(1)=u(2)
      u(jx)=u(jx-1)
c                                                     upwind - end   >>>
c-----------------------------------------------------------------------|
c     data output
      if (mod(ns,nskip).eq.0) then
         write(6,103) ns,time
         write(10,101) ns,time
         do j=1,jx
            write(10,102) x(j),u(j)
         enddo
      endif

      if (ns .lt. nstop) goto 1000
      close(10)
*=======================================================================|
      write(6,*) '  ### normal stop ###'
      end
```

Sample IDL program, rddt.pro

```
; rddt.pro
openr,1,'out.dat'
readf,1,jx,nx

; define array
ns=intarr(nx)
t=fltarr(nx)

x=fltarr(jx)
u=fltarr(jx,nx)

; temporary variables for read data
ns_and_t=fltarr(2,1)
x_and_u=fltarr(2,jx)

for n=0,nx-1 do begin
 readf,1,ns_and_t
 readf,1,x_and_u
 ns(n)=fix(ns_and_t(0,0))
 t(n)=ns_and_t(1,0)
 u(*,n)=x_and_u(1,*)
endfor

close,1
free_lun,1

x(*)=x_and_u(0,*)

delvar,ns_and_t,x_and_u

help
end
```

Sample IDL program, pldt.pro

```
!x.style=1
!y.style=1
!p.charsize=1.4

plot,x,u(*,0),xtitle='x',ytitle='u',linest=1,yrange=[-1,3],xrange=[0,100]
for n=1,nx-1 do begin
oplot,x,u(*,n)
oplot,x,u(*,n),psym=4
endfor

end
```

Sample IDL program, anime.pro

```
!x.style=1
!y.style=1
!p.charsize=1.4

window,xsize=480,ysize=480
xinteranimate,set=[480,480,nx]

for n=0,nx-1 do begin

plot,x,u(*,n),xtitle='x',ytitle='u',yrange=[-1,3],xrange=[0,100]
oplot,x,u(*,n),psym=4

xinteranimate,frame=n,window=0

endfor

xinteranimate

end
```

2.1.6 Exercise

Linear wave equation

Run the example package of the 'scalar' by referring to Section 2.1.1 to 2.1.5 of this textbook. The package is for solving the linear wave equation by the upwind algorithm. The initial values are $u_j = 1$ for $j = 1, ...50$ and $u_j = 0$ for $j = 51, ...100$. The Courant number is $\nu = c\Delta t/\Delta x = 0.25$. Make following new programs by modifying the original one, namely,

1. a program solving by the FTCS algorithm, and

2. a program solving by the Lax-Wendroff algorithm,

3. a program solving with the minmod limiter (see Equation 1.146).

Plot and compare the results of these programs with each other.

Note: A finite difference form of the one-dimensional wave equation

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \tag{2.1}$$

can be written like

$$u_j^{n+1} = u_j^n - \frac{\Delta t}{\Delta x}(f_{j+1/2}^n - f_{j-1/2}^n). \tag{2.2}$$

By using the FTCS (Forward in Time and Centered in Space) algorithm, the numerical flux is given as

$$f_{j+1/2}^n = \frac{1}{2}(f_{j+1} + f_j) = \frac{1}{2}c\ (u_{j+1} + u_j). \tag{2.3}$$

Examples of the numerical flux of other algorithms for *the linear wave equation* as follows:
Lax-Friedrich algorithm:

$$f_{j+1/2}^n = \frac{1}{2}\left[(1 - \frac{1}{\nu})\ cu_{j+1} + (1 + \frac{1}{\nu})\ cu_j\right] \tag{2.4}$$

Upwind algorithm:

$$f_{j+1/2}^n = \frac{1}{2}[\ c\ (u_{j+1} + u_j) - |c|\ (u_{j+1} - u_j)\ ] \tag{2.5}$$

Lax-Wendroff algorithm:

$$f_{j+1/2}^n = \frac{1}{2}[\ (1 - \nu)\ cu_{j+1} + (1 + \nu)\ cu_j\ ] \tag{2.6}$$

Here, $\nu \equiv c\Delta t/\Delta x$.

Burgers equation

Make and run a program for solving the Burgers equation,

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0, \tag{2.7}$$

by the 1st-order upwind algorithm. Plot the results and compare them with those in Figures 1.9 – 1.12. The numerical flux for this program can be written as

$$f_{j+1/2}^n = \frac{1}{2}\left\{\left(\frac{{u_{j+1}}^2}{2} + \frac{{u_j}^2}{2}\right) - \frac{1}{2}|u_{j+1} + u_j|(u_{j+1} - u_j)\right\}. \tag{2.8}$$

Diffusion equation

Make and run a program for solving the diffusion equation,

$$\frac{\partial u}{\partial t} = \kappa \frac{\partial^2 u}{\partial x^2} \tag{2.9}$$

by the FTCS algorithm. Plot the results and compare them with those in Figure 2.2. Set up an appropriate initial distribution, e.g. a Gaussian distribution, and define the diffusion coefficient κ instead of the wave speed c as follows:

```
c  variable
      do j=1,jx
         u(j)= exp(-(((x(j)-x(jx/2))/5.)**2))
      enddo
c
c  kappa
      kappa=1.0
```

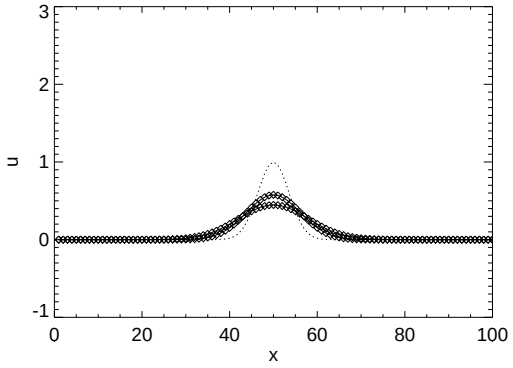

Figure 2.2: Result of a simulation for solving the diffusion equation.

2.2 Usage of the CANS package: shock tube problem

2.2.1 CANS1D

The CANS1D consists of many sets of *subroutines* and *model packages*. For example, the subroutines to solve the hydrodynamic / magnetohydrodynamic (MHD) equations are contained under the directory "`hdmlw`". The files are as follows:

```
# ls hdmlw
Makefile       mlw_ht.f       mlw_m3_g.f     mlw_m_g.f      mlwfull.f
README         mlw_ht_c.f     mlw_m3t.f      mlw_mt.f       mlwhalf.f
Readme.tex     mlw_ht_cg.f    mlw_m3t_c.f    mlw_mt_c.f     mlwsrcf.f
mlw_a.f        mlw_ht_g.f     mlw_m3t_cg.f   mlw_mt_cg.f    mlwsrch.f
mlw_h.f        mlw_m.f        mlw_m3t_g.f    mlw_mt_cgr.f
mlw_h_c.f      mlw_m3.f       mlw_m_c.f      mlw_mt_g.f
mlw_h_cg.f     mlw_m3_c.f     mlw_m_cg.f     mlw_rh.f
mlw_h_g.f      mlw_m3_cg.f    mlw_m_cgr.f    mlwartv.f
```

The model packages are collections of programs for solving the 'typical problems' that are considered to be basic for understanding the hydrodynamic / MHD simulations, e.g. the shock-tube problem, the Sedov point explosion problem and so on. Each package is contained in a separate directory whose name start with "`md_`". From here, we will explain the shock-tube problem as an example to use the CANS1D. The files in the shock-tube problem package are as follows:

```
# ls md_shktb
Makefile            bnd.f               pldt.pro
README              cipbnd.f            rddt.pro
Readme.pdf          main.f              shktb_analytic.pro
Readme.tex          main.pro
anime.pro           model.f
```

The solving program consists of several files with a file-name extension '.f' written in the Fortran language. The documents are in the files `README` and `Readme.pdf`.

2.2.2 Compilation of the subroutines in CANS1D

Before executing a program, we need to 'compile' subroutines. By this procedure, several 'library archive' files will be made with a file extension '.a' under the CANS top directory. After moving to the CANS top directory, execute the UNIX command '`make`'. (Warning! It will take much time if the CPU speed is low.) The products of this procedure are the library-archive files, `libcansnc.a`, `libcans1d.a`, `libcans2d.a`, and `libcans3d.a`. Each of these is an archive of object files of the subroutines.

```
# cd cans
# make
.......
# ls
Develop.txt   Models.tex  Readme.log  cans1d/  idl/         xmhdshktb.ps
Makefile      NonLTE/     Readme.pdf  cans2d/  libcans1d.a  xshktb.ps
Makefile.rel  README      Readme.ps   cans3d/  libcans2d.a
Models.pdf    Readme.aux  Readme.tex  cansnc/  libcans3d.a
Models.ps     Readme.dvi  avs/        htdocs/  libcansnc.a
```

2.2.3 Compilation and execution of the main program

For the compilation of the main program of the shock tube problem, move to the directory `cans1d/md_shktb`. Execute the UNIX command '`make`'. Then, the program will be executed after a compilation. If succeed, you will find several new files, `main.o`, `a.out`, `params.txt` and several files with extension of '`.dac`' in this directory. The file `main.o` is an 'object' file corresponding to 'main.f', and the file 'a.out' is an 'executable' file. The result of the simulation is contained in the output data file '*.dac'.

```
# cd cans1d/md_shktb
# make
f77   -c -o main.o main.f
f77   -c -o model.o model.f
f77   -c -o bnd.o bnd.f
f77   -c -o cipbnd.o cipbnd.f
f77 -o a.out main.o model.o bnd.o cipbnd.o \
        -L../.. -lcans1d -lcansnc
./a.out
  write    step=       0 time= 0.000E+00 nd =  1
  write    step=      51 time= 0.101E-01 nd =  2
  write    step=      93 time= 0.201E-01 nd =  3
  .....
  write    step=     585 time= 0.142E+00 nd = 16
  stop     step=     585 time= 0.142E+00
   ### normal stop ###
```

2.2.4 Visualization of a result

We usually use a special software for the visualization of the simulation results. Here we introduce "IDL" that is one of such commercial (expensive!) softwares and is very popular in astronomical data analysis both for simulations and observations.

Startup of IDL (idl)

To startup IDL, type `idl`.

```
# idl
```

Then, it starts as follows:

```
IDL Version ....
Installation number: XXXXX.
Licensed for use by: XXXXX

IDL>
```

You can enter the IDL commands after its prompt 'IDL>'. You may also run an IDL program.

Loading the data into the IDL session (.r rddt)

To load the simulation data into the IDL session, use the IDL program `rddt.pro` as follows:

```
IDL> .r rddt
```

After this, you can refer to, process, and visualize the data in the IDL session. Type '`help`' to obtain a list of available arrays and variables in the IDL session.

```
IDL> help
......
GM              FLOAT     =       1.40000
IX              LONG      =         1026
......
NX              LONG      =           16
PR              DOUBLE    = Array[1026, 16]
PR1             FLOAT     =      0.100000
RO              DOUBLE    = Array[1026, 16]
RO1             FLOAT     =      0.125000
......
T               DOUBLE    = Array[16]
TE              DOUBLE    = Array[1026, 16]
VX              DOUBLE    = Array[1026, 16]
......
```

Here `PR`, `RO`, `TE` and `VX` are arrays of the pressure, density, temperature and (x-component of) velocity, respectively. Note that in IDL sessions, the letter case of the variable names will be ignored, namely 'pr' and 'PR' correspond to the same variable.

Plot of the data (.r pldt)

To plot the data, use the IDL program `pldt.pro` as follows:

```
IDL> .r pldt
```

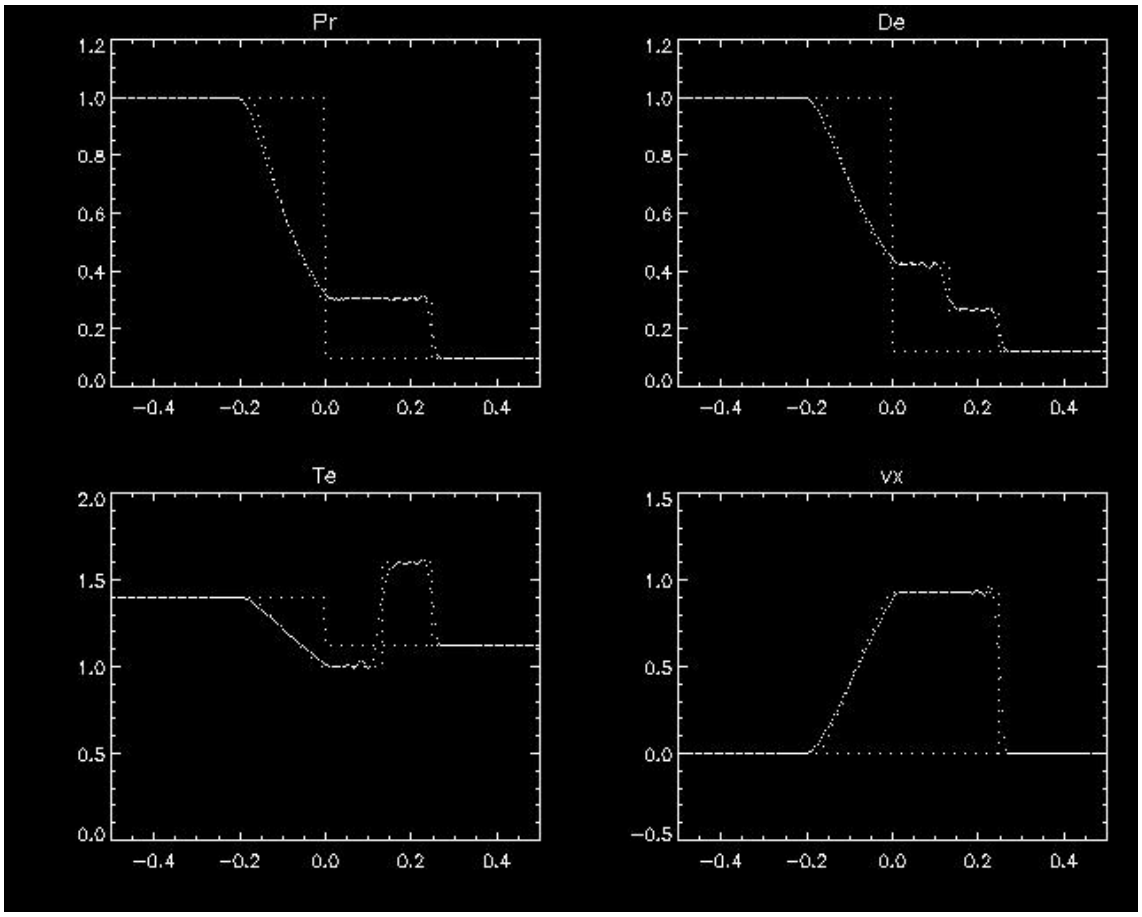

Figure 2.3: Result of the package `md_shktb`

Animation of the simulation results (.r anime)

To make an animation of the simulation results, use the IDL program `anime.pro` as follows:

```
IDL> .r anime
```

Finish an IDL session (exit)

To finish an IDL session, type `exit` after the IDL prompt.

```
IDL> exit
```

2.3 Exercise

2.3.1 Try CANS1D

1. Try the model package "Isothermal shock tube (`md_itshktb`)". Run the program and visualize the results by using IDL.

2. Try the model package "Shock tube (`md_shktb`)". Run the program and visualize the results by using IDL.
3. Try the model package "Shock formation (`md_shkform`)". Run the program and visualize the results by using IDL.
4. Try the model package "MHD shock tube (`md_mhdshktb`)". Run the program and visualize the results by using IDL (Fig. 2.4).
5. Try any of the model packages.

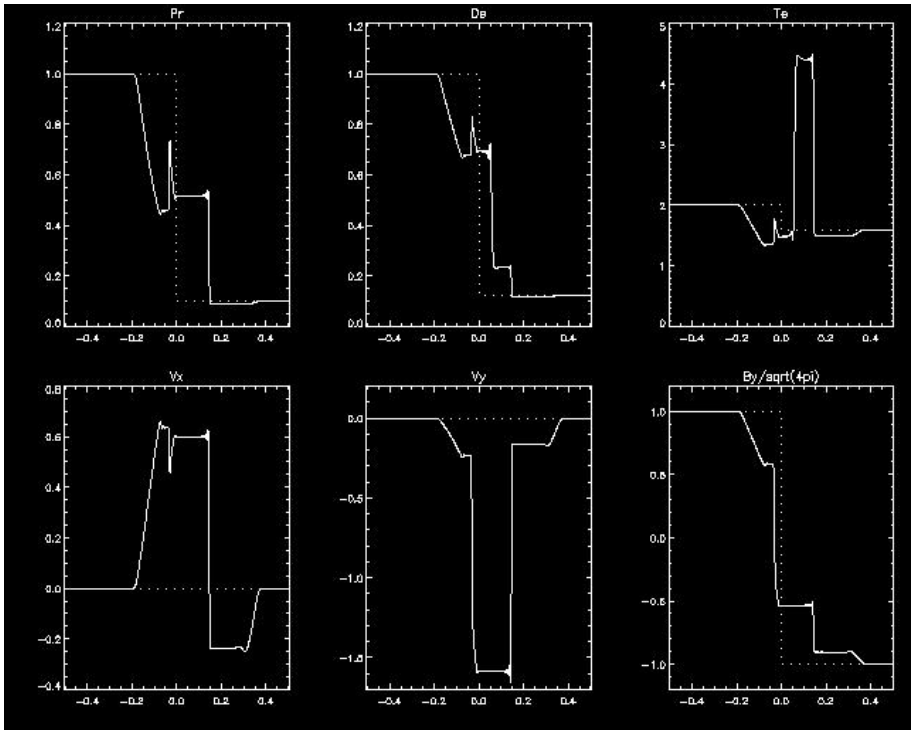

Figure 2.4: Results of `md_mhdshktb`

Note:

- When one runs a Fortran program, the output files, `params.txt` and `***.dac` are all overwritten. Rename these files or back up to any other directory before executing a program to avoid overwriting.
- To remove the object and executable files, type "`make clean`" after the UNIX prompt.

2.3.2 Try and modify the package md_shktb

Change the number of the mesh points by modyfing the appropriate file(s) in the model package "Shock tube (`md_shktb`)", run the program, and compare the results with the original one. Also change the interval of the data output and try an animation in IDL.

2.3.3 Try and modify the package md_sedov

Change the specific heat ratio γ by modyfing the appropriate file(s) in the model package "Supernova: the Sedov solution (`md_sedov`)", run the program, and compare the results with

the original one.

2.4 Advanced Exercise

Referring to Section 1.5, answer the following questions. We consider a one-dimensional hydrodynamic flow. The initial condition is given by,

$$(\rho_j, P_j, u_j) = \begin{cases} (1,\ 1,\ 0) & (j \le 0) \\ (0.81,\ 0.6,\ 0) & (j > 0) \end{cases} .$$

The gas is ideal one and specific heat ratio is $\gamma = 5/3$.

1. Compute the Roe average density from ρ_0 and ρ_1.
2. Compute H_0, H_1, and $\bar{H}$.
3. Compute the sound speed, a.
4. Compute the amplitudes, w_1, w_2 and w_3.
5. Modify the package `md_shktb` and obtain the numerical solution. Explain the numerical solution in terms of a, w_1, w_2 and w_3.
6. Try the package `md_shkin` and compare the results with Figure 1.17.

Chapter 3

Magneto-Hydrodynamical Simulation Software CANS

Originally written by H. Hanayama, translated by S. Miyaji [1]

3.1 What is CANS?

3.1.1 Astrophysical Magneto-Hydrodynamical Simulation Software

Based on the development of computers and numerical computation techniques, numerical computation which simulates the time evolution of the developing system under physical laws with computer is established as the third method to pursue the science except experiment and theory. Since Astrophysical phenomena are impossible to mimic on the earth, the method of computational science is very efficient in Astrophysics, with which we create and simulate ideal universe in the computer, and examine adequacy of the model by comparing its numerical results with observations, in order to explain physical mechanism of the phenomena.

Hence, we construct an Astro/ Space Simulator as the product of ACT-JST project "Development of Space Simulation Net Laboratory System". By this simulator, we construct and maintain code library and provide a virtual laboratory by which one can simulate, analyze, and visualize the phenomena through network. As the Astrophysical component of this Net Laboratory, we create an Astrophysical Magneto-Hydrodynamic Simulation Software library package CANS (Coordinated Astronomical Numerical Software).

3.1.2 What we can do with CANS?

By using CANS, we can perform various Astrophysical Magneto-Hydrodynamical simulations and visualize their results. Among many dynamical and various phenomena in the Universe, there are some phenomena in which macroscopic motion of plasma strongly coupled with magnetic field is the origin of the energy release and mass transport, as like flare and jet. We can simulate with CANS these phenomena which could be described with macroscopic Hydrodynamical/ Magneto-Hydrodynamical equations.

[1] This chapter is based on NetLaboratory home page URL: www.astro.phys.s.chiba-u.ac.jp/netlab/astro/ and Dr. Tanuma's (Kyoto Univ.) "How to use CANS" Web page (in Japanese) URL: www.kwasan.kyoto-u.ac.jp/~tanuma/cans_howtouse.html

In CANS, we can include many physical processes such as self-gravity, anisotropic diffusion coefficients by magnetic field orientation, magnetic diffusion, radiation cooling, and so on, and we can select from 1 dimensional to 3 dimensional codes for various Astrophysical Magneto-Hydrodynamical simulation.

3.1.3 Merit of CANS

Among some Magneto-Hydrodynamical simulation packages, the most merit of CANS is that we provide not only the simulation code itself but also a complete set of simulation model (initial condition, boundary condition, etc.), recommended parameter set, explanation of physics involved, movie of the results, and visualization and analysis tools for typical problems (basic subjects) of Astrophysical simulation. Basic subjects cover wide variety of Astrophysical phenomena such as many Hydrodynamical and Magneto-Hydrodynamical instabilities, shock wave propagation, jet, flare, and star formation.

Normally, when the simulation code becomes to be opened to the public, it is hard to thoroughly use the code by a simple-minded user. Therefore we prepare a various source codes with their initial models, their explanations, and samples of initial parameters so that one can easily reconstruct the simulation model based on these supplies and can start new simulation. We adopted netCFD format as a standard input and output format of the data so that one can use many visualization software which offers netCFD format.

3.1.4 Structure of CANS

CANS consists of two parts; common part for Hydrodynamical/ Magneto-Hydrodynamical simulation code and special part for basic subjects with modules and explanations of each subject. As the simulation engine to solve Magneto-Hydrodynamical equation, there are 3 engines such as Modified Lax-Wendroff scheme, Roe scheme, and CIP-MOCCT scheme. One can execute the simulation with other scheme by exchanging the engine without changing other parts. Not only the modules for scalar machine but also parallelized module package with MPI for parallel machine are included in 1, 2, and 3 dimension packages.

3.2 Let's Use CANS!

At first. we briefly explain how to use CANS from installation, execution, and visualization of the result. Please refer following Web pages from developers; Dr. R. Matsumoto (Chiba Univ.) and Dr. T. Yokoyama (Univ. of Tokyo).

CANS Web page (over all description and some English documents)

`http://www.astro.phys.s.chiba-u.ac.jp/netlab/astro/`

CANS download page

`http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/`

CANS document (in Japanese)

`http://www.kwasan.kyoto-u.ac.jp/~tanuma/JST.html`

3.2.1 Working Environment

CANS code is written in Fortran. Therefore its working environment should be at least Fortran compilable. Based on compilation tests on various UNIX environment by the developers, on the following environments CANS is compilable and executable. But, CANS is written to be

executable on many UNIX environments so that even on other environments there should be only a slight or less change needed.

FreeBSD4.2
Linux 2.2
IRIX64 6.5
SunOS5.7
HP-UX 11.0
SUPER-UX 11.1 (SX5)
UXP/V (VPP)

3.2.2 Installation

Download CANS data file from CANS home page to your home directory and extract the data from compressed format.

There are "beta version" and "released version (version 3.0)" of CANS; the beta version is under the test period for developing the code and it includes the newest patches but may contain some bugs. The released version is already examined with various code checks and simulation tests and certified to use by the developers. Therefore released version is reliable but may be slower than the beta version.

Here, we explain the released version. Its full code is downloadable from following Web page.

`http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/src/cans-release.tgz`

Download CANS released version to your home directory and extract the files by tar command.

```
>cd
>wget http://www-space.eps.s.u-tokyo.ac.jp/~yokoyama/etc/cans/src/cans-release.tgz
>tar -zxvf cans-release.tgz
```

3.2.3 Check the Contents

After the extraction of CANS, check the contents of CANS sub-directory.

```
>cd cans
>ls
Develop.txt Models.pdf README     cans1d/ cans3d/ htdocs/
Makefile    NonLTE     Readme.pdf cans2d/ cansnc/ idl/
```

'*.pdf' files and README file is written by Dr. R. Matsumoto (Chiba Univ.), Dr. T. Yokoyama (Univ. of Tokyo) and Dr. N. Fukuda (Okayama Univ. Sci.) and describe explanations and guides of various CANS modules so that you should read them first (if you can read Japanese). In each sub-directory, there are README files which describe functions of each programs and we recommend you to read them before you use these modules/ subroutines.

In the directories of 'cans1d', 'cans2d', and 'cans3d', there is basic subjects and

common modules for 1, 2, and 3 dimensional simulation, respectively. In the 'cansnc' directory, there are the programs for input and output by netCFD format. In the 'idl' subdirectory, there are visualization programs for IDL visualization software. IDL itself is a commercial software and please refer following Web page.

```
http://www.rsinc.com/idl/
```

Let's check the contents of cans2d sub-directory.

```
>cd cans2d
>ls
bc/          cndbicg/        cndsor/       cndsormpi/    common/
commonmpi/   hdmlw/          htcl/         md_advect/    md_awdecay/
md_cloud/    md_cme/         md_cndsp/     md_cndtb/     md_corjet/
md_efr/      md_itmhdshktb/  md_itshktb/   md_jetprop/   md_kh/
md_mhd3dkh/  md_mhd3dshktb/  md_mhdcloud/  md_mhdcondtb/ md_mhdgwave/
md_mhdkh/    md_mhdshktb/    md_mhdsn/     md_mhdwave/   md_mri/
md_parker/   md_reccnd/      md_recon/     md_recon3/    md_rt/
md_sedov/    md_shkref/      md_shktb/     md_sndwave/   md_thinst/
mdp_awdecay/ mdp_cme/        mdp_cndsp/    mdp_cndtb/    mdp_corjet/
mdp_efr/     mdp_itmhdshktb/ mdp_itshktb/  mdp_jetprop/  mdp_kh/
mdp_mhd3kh/  mdp_mhd3shktb/  mdp_mhdcndtb/ mdp_mhdkh/    mdp_mhdshktb/
mdp_mhdsn/   mdp_mhdwave/    mdp_mri/      mdp_recon/    mdp_recon3/
mdp_rt/      mdp_sedov/      mdp_shkref/   mdp_shktb/    mdp_thinst/
```

Followings are the explanations of each sub-directory.

```
hdmlw/  module to solve Hydrodynamical/ MHD equations with Modified Lax-Wendroff
        + artificial viscosity.
bc/     procedure to define boundary condition.
common/ module for various common routines for computation.
cndsor/ module to solve thermal conduction with implicit method (1st order in time:
        matrix conversion is based on Red Black SOR scheme).
cndbicg/ module to solve thermal conduction with implicit method (1st order in time:
        matrix conversion is based on BICG scheme)
htcl/   module to solve radiation cooling and static heating with explicit method.
commonmpi/ module of common routines for MPI.
cndsormpi/ module to solve thermal conduction with implicit method (1st order in time:
          matrix conversion is based on Red Black SOR scheme) for MPI.
md_***/ module for 2 dimension basic subjects for scalar machine.
mdp_***/ module for 2 dimension basic subjects for parallel machine.
(For *** part, please refer the next screen.)
```

'md' and 'mdp' directories are the modules for basic subjects. For almost all 2 dimensional basic subjects, there are packages for scalar machine ('md' module for 1CPU computation) and for parallel machine ('mdp' module for plural CPUs computation). For example, 'md_mhdsn' and 'mdp_mhdsn' compute basically the same supernova remnant model except by scalar and parallel procedures, respectively.

Following is a summary of 2 dimensional basic subjects.

```
md_advect/ simple advection [advection]
md_awdecay/ large amplitude Alfv\'en wave attenuation instability [isothermal 3 component MHD]
md_cloud/ isothermal self-gravity collapse [isothermal hydrodynamics, self-gravity]
md_cme/ coronal mass ejection: Low solution [MHD, spherical coordinate]
md_cndsp/ spherical symmetric thermal conduction [thermal conduction, cylindrical/ spherical
            coordinate]
md_cndtb/ simple thermal conduction [thermal conduction]
md_corjet/ solar coronal jet [MHD, gravity, resistivity]
md_efr/ solar emerging flux of magnetic fields: Parker instability [MHD, gravity]
md_itmhdshktb/ isothermal MHD shock tube [isothermal MHD]
md_itshktb/ isothermal shock tube [isothermal Hydrodynamics]
md_jetprop/ propagation of jet [Hydrodynamics, cylindrical coordinate]
md_kh/ Kelvin-Helmholtz instability [Hydrodynamics]
md_mhd3kh/ 3 components MHD Kelvin-Helmholtz instability [3 components MHD]
md_mhd3shktb/ 3 components MHD shock tube [3 components MHD]
md_mhdcloud/ isothermal MHD self-gravity collapse [isothermal MHD, gravity]
md_mhdcndtb/ simple MHD thermal conduction [MHD, thermal conduction]
md_mhdgwave/ MHD wave transport in stratosphere [MHD]
md_mhdkh/ MHD Kelvin-Helmholtz instability [MHD]
md_shktb/ shock tube [Hydrodynamics]
md_mhdsn/ MHD supernova remnant [MHD, cylindrical/ spherical coordinate]
md_mhdwave/ linear MHD wave transport [MHD]
md_mri/ magneto-rotational (Balbus-Hawley instability [MHD, tidal Coriolis force]
md_parker/ Parker instability in galaxy [MHD, gravity]
md_mhdshktb/ MHD shock tube [MHD]
md_reccnd/ magnetic field reconnection with thermal conduction [MHD, resistivity,
            thermal conduction]
md_recon/ magnetic field reconnection [MHD, resistivity]
md_recon3/ 3 components magnetic field reconnection [3 components MHD, resistivity]
md_rt/ Rayleigh-Taylor instability [Hydrodynamics, gravity]
md_sedov/ supernova remnant: Sedov solution [Hydrodynamics, cylindorical/ spherical
            coordinate]
md_shkref/ reflection of shock wave [Hydrodynamics]
md_shktb/ shock tube [Hydrodynamics]
md_thinst/ thermal instability [radiation cooling]
```

3.2.4 Compilation

In '~/cans' directory, compile the program by "make" command and create library files such as 'libcans1d.a' etc. (If you change the machine or something wrong has happened, you should recompile the program.)

```
>cd ~/cans
>make 'FC=f77'
>ls
libcans1d.a, libcans2d.a, libcans3d.a, libcansnc.a
```

Here, 'FC=f77' means the command name of the Fortran compiler of your system. If your system has other version (f77 always mean Fortran 77 compiler) such as f90 (for Fortran 90), g77 (for GNU project Fortran 77), frt, and so on, use proper name of the Fortran compiler of your system.

If you want to execute parallel computation, please follow following procedure.

1. At ~/cans/cansnc directory, "make clean" for assurance.
2. Compile the module by "make 'FC=mpif77'".
3. Move to ~/cans/cans2d directory, and "make clean".
4. Compile again by "make 'FC=mpif77'".
5 "make mpi" at the same directory.

```
>cd ~/cans/cansnc
>make clean
>make 'FC=mpif77'
>cd ../cans2d
>make clean
>make 'FC=mpif77'
>make mpi
```

3.2.5 Execution of CANS program

As an example, we will describe in the followings for the case to compute 'MHD supernova remnant' model from the 2 dimensional basic subjects.

At first, move to '`cans2d/md_mhdsn`', let's check whether the files listed in Table 3.1 are created.

```
>cd cans2d/md_mhdsn
>ls
Makefile Makefile-nc Makefile-pgnc anime.pro bnd.f
main.f   model.f     pldt.pro      rddt.pro  rdnc.pro
```

File Name	Content/Function
Makefile	content of 'make' command
Makefile-nc	use when output to netCFD format
Makefile-pgnc	use when visualize by pgplot
main.f	main program
model.f	initialization subroutine
bnd.f	boundary condition subroutine
rddt.pro	file to read output data by IDL
rdnc.pro	file to read output data in netCFD format
pldt.pro	file to plot 2 dimensional contour map by IDL
anime.pro	file to show animation by IDL

Table 3.1: Explanation of Initial File in MHD Supernova Remnant Module

When "make" command is executed, beside 'a.out', output files of simulation result (extension: dac) such as 'ay.dac' and simulation parameter output file 'params.txt' are created[2]

[2]If you faced "Segmentation fault" error, you might face compilation quota limitation. In such a case, you may avoid the error by changing dimension parameters at line 5 of '/md_mhdsn/main.f' such as 'parameter(ix=50,jx=50)'. Of course you can change the quota limit of your system by checking "limit" command.

```
>make 'FC=f77'   <= For parallel computation, set 'FC=mpif77'.
>ls
Makefile Makefile-nc Makefile-pgnc a.out    anime.pro
ay.dac   bnd.f       bnd.o         bx.dac   bz.dac
main.f   main.o      model.f       model.o  params.txt
pldt.pro pr.dac      rddt.pro      rdnc.pro ro.dac
t.dac    vx.dac      vz.dac        x.dac    z.dac
```

3.2.6 Preparation for IDL

We expect that in your system, visualization software IDL IDL (The Interactive Data Language) has already installed. IDL is a commercial software but it is popular at least in Astrophysical community. Detailed explanation of the visualization by IDL will be given in somewhere.

In your '~/.cshrc', please add a path to IDL as follows.

```
setenv IDL_PATH +/usr/local/rsi/idl/lib:~/cans/idl/
```

Then you can use IDL programs in ~/cans/idl (dacgetparam.pro etc.). However, the path to IDL software itself (/usr/local/rsi/idl/lib part) is system dependent so that you should consult with your system administrator.

3.2.7 Read Data by IDL and Display

Starting IDL, read the result data outputted file 'dac file'.

```
>idl         <= start IDL.
IDL>.r rddt     <= read data.
```

[Sample of Read Data]

Display 2 dimensional plot of density contour.

```
IDL>.r pldt   <= plot data.
Plot columns & rows ? : 1,1   <= set plotting matrix for plot window.
Variable for color-maps ? (ro,pr,te) : ro <=select physical variables to be plotted
                                          (ro: density, pr: pressure, te: temperature).
Start step ? : 10 <= set time (output sequence number).
```

When input parameters like above, the simulation result will be displayed in the window like Figure 3.1.

If you want to display by color contour, set color table like follows.

```
IDL>device,decomposed=0
IDL>loadct,5
```

When you want to change the color table, type 'xloadct' and select new color table.

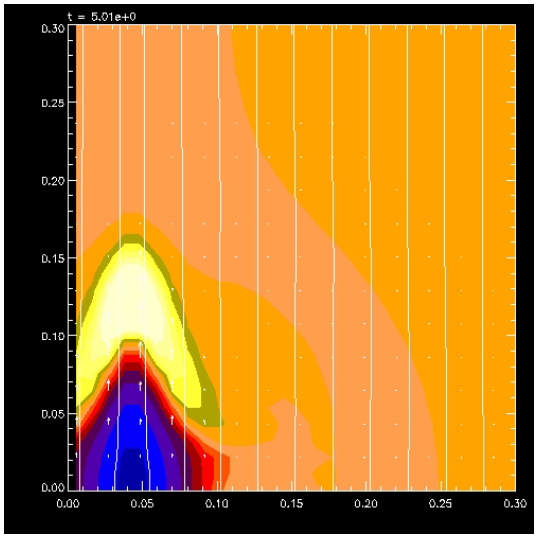

Figure 3.1: 2 Dimensional Plot of Density Contour. White lines show magnetic field line.

By a similar procedure, you can display the animation.

```
IDL>.r anime   <= animation display.
Select a varibale ? (ro,pr,te,vx,vy,bx,by,az):ro
```

For the case of other modules, please refer following Web page in which their results are disclosed as demo pages.

CANS 2D demo page

`http://www.astro.phys.s.chiba-u.ac.jp/netlab/cans/movie2/frame.html`

3.3 Computation Flow

The flow of computation procedure is based on the following main program.

`~/cans/cans2d/md_mhdsn/main.f`

Though 'main.f' program consists of many subroutines which represents model setting, computation engine, and so on, basically we can categorize them into three parts; initial setup, main computation, and end processing. In the followings, we will explain these parts according to refer an actual program.

3.3.1 Initial Set Up - main.f

In the initial setup part, you should set up initial condition to start the computation; model setup, definition of each parameters, output setup, grid size, continuation program setting, boundary condition, magnetic field setting, error check setup, and so on.

```
c=======================================================================|
c     array definitions
c=======================================================================|
      implicit double precision (a-h,o-z)
      parameter (ix=103,jx=102)
! ix is the number of mesh points in X axis.
! jx is the number of mesh points in Z axis.
      dimension x(ix),xm(ix),dx(ix),dxm(ix)
      dimension z(jx),zm(jx),dz(jx),dzm(jx)
      dimension ro(ix,jx),pr(ix,jx)
      dimension vx(ix,jx),vz(ix,jx)
      dimension bx(ix,jx),bz(ix,jx)
      dimension ay(ix,jx)
! define  parameter matrix
c=======================================================================|
c     prologue
c=======================================================================|
      mcont=0
      ndi=1000
! 'mcont' and 'ndi' are parameters for continuation. mcont=1; start continuation
! "continuation" means that re-start the computation after the 1st round computation
! as a continued computation.  Set up parameters for continuation such as terminate
! time 'tend' and/or terminate step number 'nstop'.
c-----------------------------------------------------------------------|
c   parameters
c   margin: 1 in MLW, 2 in Roe, 4 in CIP
      margin=4
! margin is the grid number which will be placed as a ample margin at the outside of
! computation area.
c-----------------------------------------------------------------------|
c  file open
! (skip netCFD part)
! we will not discuss 'netCFD' format here.
      mf_params=9
      call dacdefparam(mf_params,'params.txt')
      mf_t =10
      call dacdef0s(mf_t,'t.dac',6)
      mf_ro=20
      call dacdef2s(mf_ro,'ro.dac',6,ix,jx)
      mf_pr=21
      call dacdef2s(mf_pr,'pr.dac',6,ix,jx)
      mf_vx=22
      call dacdef2s(mf_vx,'vx.dac',6,ix,jx)
      mf_vz=24
      call dacdef2s(mf_vz,'vz.dac',6,ix,jx)
      mf_bx=25
      call dacdef2s(mf_bx,'bx.dac',6,ix,jx)
      mf_bz=27
      call dacdef2s(mf_bz,'bz.dac',6,ix,jx)
      mf_ay=28
      call dacdef2s(mf_ay,'ay.dac',6,ix,jx)
      call dacputparamc(mf_params,'comment','cans2d md_mhdsn')
      call dacputparami(mf_params,'ix',ix)
      call dacputparami(mf_params,'jx',jx)
      call dacputparami(mf_params,'margin',margin)
! output of the data is done by subroutine 'dacdefparam.f' etc. in '/cansnc' directory.
! There, creation of output file ('dac' files) and parameters output have done.
```

```
c-----------------------------------------------------------------------|
c  initialize counters
      nd=1
! 'nd' is output sequence number.
      time  = 0.0
! 'time' is time, of course.
      timep = 0.0
      ns    = 0
! 'ns' is the number of computational step.
      merr  = 0
! 'merr' is error account number.
c-----------------------------------------------------------------------|
c   time control parameters
c     nstop : number of total time steps for the run
      tend=5.0
! 'tend' is the time when the computation should be stopped.
      dtout=0.5
! 'dtout' is the time interval for writing the data.
      nstop=1000000
! 'nstop' is the step number at where the computation should be stopped.
! when 'time > tend' or 'ns > nstop', the computation should be stopped there.
c     dtout=1.d-10
c     nstop=3
c-----------------------------------------------------------------------|
c   setup numerical model (grid, initial conditions, etc.)
      call model(idf,ro,pr,vx,vz,bx,bz,gm,margin,x,ix,z,jx
     &    ,mf_params)
! subroutine model.f
! setting of grid interval, environmental gas, magnetic field,
! initial explosion velocity, etc.
! details of each subroutine will be given later.
c-----------------------------------------------------------------------|
! skip continuation (re-computation) program part
c-----------------------------------------------------------------------|
c     ready

      call grdrdy(dx,xm,dxm,x,ix)
      call grdrdy(dz,zm,dzm,z,jx)
! subroutine grdrdy.f
! set intervals between cells and grids.

      call bbtoaa_c(ay,bz,bx,dzm,dxm,x,ix,jx)
! subroutine bbtoaa_c.f
! create a vector potential in order to draw magnetic field lines.

      call bnd(margin,ro,pr,vx,vz,bx,bz,ay,ix,jx)
! subroutine bnd.f
! apply boundary condition

      floor=1.d-9
! floor is the lower limit of physical parameters; pressure,
! density, etc.

      call chkdav(n_floor,ro,vx,vz,floor,ix,jx)
      call chkdav(n_floor,pr,vx,vz,floor,ix,jx)
! subroutine chkdav.f
! check the value of physical variables whether they are below 'floor',
! if they are lower than 'floor' set these value to be 'floor'.
```

```
c-----------------------------------------------------------------------|
c     data output
! (skip 'netCFD' part.)
      mf_x=11
      call dacdef1d(mf_x,'x.dac',6,ix)
      write(mf_x) x
      mf_z=12
      call dacdef1d(mf_z,'z.dac',6,jx)
      write(mf_z) z
      call dacputparamd(mf_params,'gm',gm)
      write(mf_t) time
      write(mf_ro) ro
      write(mf_pr) pr
      write(mf_vx) vx
      write(mf_vz) vz
      write(mf_bx) bx
      write(mf_bz) bz
      write(mf_ay) ay
      write(6,913) ns,time,nd
      nd=nd+1
! output initial value.
```

3.3.2 Selection of computation scheme - main.f

Here computation time span dt is determined under CFL (Courant-Friedrich- Lewy) condition. Modified Lax-Wendroff scheme is adopted for advection calculation. After applying boundary condition, error check is performed, and outputs the result under output setup condition.

```
c=======================================================================|
c     time integration
c=======================================================================|
1000  continue
         ns = ns+1
         mwflag=0
! 'mwflag' is a flag when subroutine 'cfl_m.f' is terminated under abnormal condition.
! In such a case, flag is kept to be '0' and output the values.
c-----------------------------------------------------------------------|
c     obtain time spacing

         safety=0.4d0
! safety is a safety parameter to keep 'dt' value less than the Courant number.

         dtmin=1.d-10
! 'dtmin' is the minimum value of 'dt'.  When 'dt' is less than 'dtmin', the program
! regards that some instabilities take place so that computation will be terminated.
     call cfl_m(dt,safety,dtmin,merr,gm,ro,pr,vx,vz,bx,bz
     &      ,dx,ix,dz,jx)
! subroutine cfl_m.f
! determine 'dt' with CFL condition.
```

```
         if (merr.ne.0) goto 9999
! check error flag and if the error takes place terminate here.

         timep = time
         time = time+dt
! advance time by adding 'dt' to previous 'time'.
c-----------------------------------------------------------------------|
c     solve hydrodynamic equations
c                                                    hdmlw - start >>>
         qav=3.d0
! qav is a parameter to define the strength of artificial visicosity.
         call mlw_m_c(ro,pr,vx,vz,bx,bz,ay
     &          ,dt,qav,gm,x,xm,dx,dxm,ix,dz,dzm,jx)
! subroutine mlw_m_c.f
! main engine which solves differential equations with Modified Lax-Wendroff scheme
c                                                    hdmlw - end   <<<
! (skip Roe scheme)

         call bnd(margin,ro,pr,vx,vz,bx,bz,ay,ix,jx)
! subroutine bnd.f
! apply boundary condition

         floor=1.d-9
         call chkdav(n_floor,ro,vx,vz,floor,ix,jx)
         call chkdav(n_floor,pr,vx,vz,floor,ix,jx)
! subroutine chkdav.f
! check pressure & density are lower than lower limit.
c-----------------------------------------------------------------------|
c     data output
! here, data is outputted.
         mw=0
! 'mw' is an output parameter and normally has '0' value.
         nt1=int(timep/dtout)
         nt2=int(time/dtout)
         if (nt1.lt.nt2) mw=1
! when 'dtout' has past from previous output time, 'mw' flag is set to be '1'
! and the output data is written.

         if (mw.ne.0) then
!(skip netCFD part)
       write(mf_t) time
       write(mf_ro) ro
       write(mf_pr) pr
       write(mf_vx) vx
       write(mf_vz) vz
       write(mf_bx) bx
       write(mf_bz) bz
       write(mf_ay) ay
             write(6,913) ns,time,nd
             nd=nd+1
             mwflag=1
! when the computation is terminated without any problem, 'mwflag' is set to be '1'
! and no output at the end.
         endif

c-----------------------------------------------------------------------|
c     loop test
       if (ns .lt. nstop .and. time .lt. tend) goto 1000
! if 'ns' exceeds 'nstop' or 'time' exceeds 'tend', the computaion is stopped here.
! Otherwise, it will go back to 1000 and loop the sequence.
```

3.3.3 End Procedure - main.f

At the end, the computation would be stopped under initial termination condition. When the error took place, by checking the errors, the system would output the data of previous stage before the error took place.

```
c=====================================================================|
c     epilogue
c=====================================================================|
9999  continue
! when error took place, error routine starts from here.
c---------------------------------------------------------------------|
c  data output
      if (mwflag.eq.0) then
! if error took place in subroutine 'cfl_m.f', because of error flag
! 'mwflag=0', the result would be output here.
           write(6,913) ns,time,nd
! (skip netcdf part)
           write(mf_t) time
           write(mf_ro) ro
           write(mf_pr) pr
           write(mf_vx) vx
           write(mf_vz) vz
           write(mf_bx) bx
           write(mf_bz) bz
           write(mf_ay) ay
      endif
c---------------------------------------------------------------------|
c  file close
!(skip netCFD part)
c---------------------------------------------------------------------|
c  ending message
      write(6,915) ns,time
      if (merr.eq.0) then
        write(6,*) '  ### normal stop ###'
      else
        write(6,*) '  ### abnormal stop ###'
      endif
! when there is no error; 'normal stop', or with error; 'abnormal
! stop' will be printed out.
      stop
913   format (1x,' write    ','step=',i8,' time=',e10.3,' nd =',i3)
915   format (1x,' stop     ','step=',i8,' time=',e10.3)
      end
```

3.3.4 Model Definition - model.f

Subroutine 'model.f' is in the same directory with 'main.f'. In this subroutine, the coordinates of X and Z axes and their values will be defined.

For this model (supernova remnant), the central density 'ro(1.0)' and the pressure of interstellar matter 'prism (10^{-8})' and the plasma 'beta (10^{-5})'[3], the strength of magnetic field 'b_0(0.1853)' are defined.

Explosion energy is given as the pressure (peak value = 1) with Gauss distribution.

[3]plasma beta = gas pressure / magnetic pressure

```
c=======================================================================|
      subroutine model(idf,ro,pr,vx,vz,bx,bz,gm,margin,x,ix,z,jx
     &   ,mf_params)
c=======================================================================|
      implicit double precision (a-h,o-z)
c-----------------------------------------------------------------------|
      dimension x(ix),dxm(ix)
      dimension z(jx),dzm(jx)
      dimension ro(ix,jx),pr(ix,jx),vx(ix,jx)
      dimension vz(ix,jx),bx(ix,jx),bz(ix,jx)
! 'x(i)' and 'z(j)' are the value of grid points on X- and Z-axes.
! For example, x(5)=0.d0 and z(5)=0.d0 mean the point where
! (i,j)=(5,5) corresponds the origin (0,0) of XZ coordinates.
! dxm(i) and dzm(j) are intervals of grid points in XZ coordinates.

c-----------------------------------------------------------------------|
c   parameters
c-----------------------------------------------------------------------|
      gm=5./3.
! 'gm' is the specific heat gamma.

c-----------------------------------------------------------------------|
c   grid
c-----------------------------------------------------------------------|
      dx0=1./real(ix-margin*2+1)
! define the basis of grid interval in X-axis.
! be aware that 'i' starts from 1.

      do i=1,ix
         dxm(i)=dx0
      enddo
! up to 'ix', grid interval is kept constant 'dx0'.

      izero=margin+1
      izero=1
! in order to place the origin on the mirror boundary (on the axis), 'izero' would be set
! 'margin+1' but is defined 'izero=1' here in order to avoid numerical problem on Z-axis.
  x(izero)=0.5*dx0
! this means 'i' is between 'izero' and 'izero-1', i.e., the origin of the X-axis is on
! the cell.
      do i=izero+1,ix
         x(i) = x(i-1)+dxm(i-1)
      enddo
      do i=izero-1,1,-1
         x(i) = x(i+1)-dxm(i)
      enddo
! define coordinate from 'x(izero)' and add 'dxm' for each grid point.

      dz0=1./real(jx-margin*2+1)
! define interval for Z-axis. Pay attention that j starts from '1'.

      do j=1,jx
         dzm(j)=dz0
      enddo
! up to 'jx' mesh, interval is kept 'dz0'.
```

```
      jzero=margin+1
      z(jzero)=0.
! for X-axis, there is no problem for computation so that the
! origin of Z-axis on the grid (it could be on the cell).
      do j=jzero+1,jx
         z(j) = z(j-1)+dzm(j-1)
      enddo

      do j=jzero-1,1,-1
         z(j) = z(j+1)-dzm(j)
      enddo
! define coordinate from 'z(jzero)' and add 'dzm' for each grid point.

c-----------------------------------------------------------------------|
c     store initial condition into common area
c-----------------------------------------------------------------------|
      prism=1.e-8
! 'prism' is the pressure of interstellar gas
      wexp=0.02
! 'wexp' is the width of Gaussian distribution where the pressure has
! 1/e of peak value.
c     prism=1/gm
      betai=1.0d5
! 'betai' is the inverse of plasma beta (=P-gas /P-magnetic field)
      pi = acos(-1.0d0)
      b0=sqrt(prism*8*pi*betai)
! set the strength of magnetic field from pressure and plasma beta

      do j=1,jx
      do i=1,ix
         ro(i,j)  = 1.
! set initial interstellar gas density to be '1.'.
         vx(i,j) = 0.0
         vz(i,j) = 0.0
! set initial velocity of the interstellar gas to be '0'.
         ss=sqrt(x(i)**2+z(j)**2)
! 'ss' is the distance from the origin in XZ coordinate.
         pr(i,j)  = prism+(1.-prism)*exp(-(ss/wexp)**2)
! explosion is given as pressure in Gauss distribution.
! center of the explosion is the origin.
c        pr(i,j) = 1/gm+0.1/gm*exp(-(ss/wexp)**2)
         bx(i,j) = 0.0
         bz(i,j) = b0
! magnetic field is assumed uniform in Z-direction.
      enddo
      enddo
c-----------------------------------------------------------------------|
c     write parameters to file
c-----------------------------------------------------------------------|
! (skip netcdf part)
      call dacputparamd(mf_params,'gm',gm)
      call dacputparamd(mf_params,'wexp',wexp)
      call dacputparamd(mf_params,'prism',prism)
      call dacputparamd(mf_params,'betai',betai)
! parameters needed have been output.

      return
      end
```

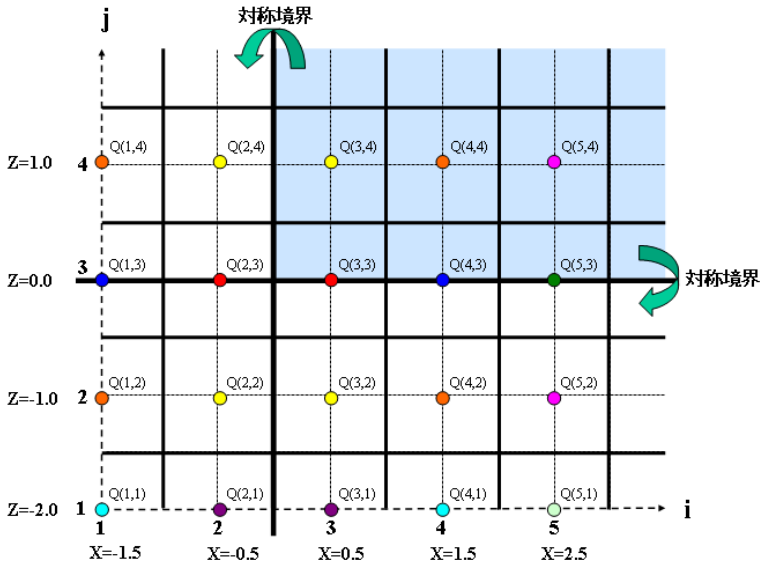

Figure 3.2: Boundary Condition near X-axis and Z-axis. It is shown with the color coding that physical quantities of each grid point is symmetric at the boundary (thick solid lines).

3.3.5 Setup of Grid Interval - grdrdy.f

Subroutine 'grdrdy.f' is in 'cans2d/common' directory. There, cell interval 'dx', grid interval 'dxm', and the middle point of grids 'xm' are defined. Details is given in README of 'cans2d/common' directory.

3.3.6 Setup for Vector Potential - bbtoaa_c.f

Subroutine 'bbtoaa_c.f' is in 'cans2d/common' directory. There, vector potential 'a_v' is calculated in order to draw magnetic field lines. Details is given in README of 'cans2d/common' directory.

3.3.7 Setup for Boundary Condition - bnd.f

Though 'bnd.f' itself is in the same directory where 'main.f' is, but subroutines of which apply boundary condition to the computating region is in 'cans2d/bc' directory and subroutines assigned in 'bnd.f' will work at the boundary. In case of the MHD supernova remnant module, because it is computed in 2 dimensional cylindrical coordinate, mirror boundary on the axes and free boundary at the outer boundary are applied. Details is given in README of 'cans2d/bc' directory. Figure 3.2 is the physical quantities at the boundary near axes for the case of 'margin=2'.

3.3.8 Check the Lower Limit - chkdav.f

Subroutine 'chkdav.f' is in 'cans2d/common' directory. When a parameter is lower than 'floor', set the parameter to be the value defined by 'floor'. Simultaneously, it is counted in 'n_events' that how many times 'floor' has used during the computation. Details is given in README of 'cans2d/common' directory.

3.3.9 Setup of CFL Condition - cfl_m.f

Subroutine 'cfl_m.f' is in 'cans2d/common' directory. According to CFL condition, it determines dt. It calculates the Alfvén velocity, the interstellar sound speed, and the gas velocity for each cell and get the maximum value among them for each cell. Using the intervals between cells, 'dx' and 'dy', it calculate 'dt' according to CFL condition with safety parameter '$safety$' in order to constrain the growth of any numerical instability. Details is given in README of 'cans2d/common' directory.

3.4 Modified Lax-Wendroff Scheme

For your convenience to understand the contents in the subroutine 'mlw_m_c.f', we will describe the solving method of Magneto-Hydrodynamic differential equations hereafter. As an actual example to differentiate basic physical equations, we adopt a numerical computation scheme, Modified Lax-Wendroff scheme here.

3.4.1 Basic Equations

As basic equations for Magneto-Hydrodynamics in 2 dimensional cylindrical coordinate, mass conservation, momentum conservation, energy conservation, and induction equation of the magnetic field are written as follows;

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0. \tag{3.1}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\rho \mathbf{v}\mathbf{v}) = -\nabla \mathrm{p} - \nabla \left(\frac{\mathbf{B}^2}{8\pi} \right) + \frac{1}{4\pi} (\mathbf{B} \cdot \nabla)\mathbf{B}. \tag{3.2}$$

$$\frac{\partial}{\partial t} \left(e + \frac{\mathbf{B}^2}{8\pi} \right) + \nabla \cdot \left[(e + \mathrm{p})\mathbf{v} + \frac{1}{4\pi} \{ \mathbf{B} \times (\mathbf{v} \times \mathbf{B}) \} \right] = 0. \tag{3.3}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}). \tag{3.4}$$

Here, $\rho, \mathbf{v}, p, t, \mathbf{B}$ mean the density, velocity, pressure, time, and magnetic field of the interstellar gas, respectively. We used that $e = \mathrm{p}/(\gamma - 1) + \rho v^2/2$ (total energy is equal to the sum of thermal and kinetic energies par volume) and specific heat $\gamma = 5/3$.

We assume isothermal evolution of the remnant so that we can neglect ϕ component of the cylindrical coordinate and only include radial (R-coordinate) and vertical (Z-coordinate) directions.

For the case of 2 dimensional cylindrical coordinate, we should pay attention on the following equations, assuming $\mathbf{A}$ and ψ are any vector and scalar.

$$\nabla \cdot \mathbf{A} = \frac{1}{r} \frac{\partial}{\partial r} (r A_r) + \frac{\partial A_z}{\partial z}. \tag{3.5}$$

$$\nabla\psi = \left(\frac{\partial\psi}{\partial r}, \frac{\partial\psi}{\partial z}\right). \tag{3.6}$$

Here, $\rho\mathbf{vv}$ is a tensor product,

$$\begin{pmatrix} \rho v_r v_r & \rho v_r v_z \\ \rho v_z v_r & \rho v_z v_z \end{pmatrix}, \tag{3.7}$$

so that

$$(\nabla\cdot\rho\mathbf{vv})_r = \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r v_r) + \frac{\partial\rho v_r v_z}{\partial z}, \tag{3.8}$$

$$(\nabla\cdot\rho\mathbf{vv})_z = \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_z v_r) + \frac{\partial\rho v_z v_z}{\partial z}. \tag{3.9}$$

Before explaining the solving method of basic equations with the use of advection equation, we will briefly describe the Magneto-Hydrodynamics itself.

3.4.2 Magneto Hydrodynamics

Maxwell equations for Electro-Magnetic fields are written as;

$$\nabla\times\mathbf{E} = -\frac{1}{c}\frac{\partial\mathbf{B}}{\partial t}, \tag{3.10}$$

$$\nabla\times\mathbf{B} = \frac{4\pi}{c}\mathbf{j}. \tag{3.11}$$

Here, 'c' is the speed of light and '$\mathbf{j}$' is current density. Since, for the case of interstellar gas, hydrodynamic time scale is much longer than the vibration period of electro-magnetic waves, we neglect displacement current. And also, since the gas density is very low, we assumed that magnetic permeability is equal to 1 as like true vacuum. With (3.10), (3.11) and Ohm's law

$$\mathbf{j} = \sigma\left(\mathbf{E} + \frac{\mathbf{v}\times\mathbf{B}}{c}\right), \tag{3.12}$$

we get induction equation for magnetic field as;

$$\frac{\partial\mathbf{B}}{\partial t} = \nabla\times(\mathbf{v}\times\mathbf{B}) + \frac{c^2}{4\pi\sigma}\nabla^2\mathbf{B}. \tag{3.13}$$

Here, 'σ' is the electric conductivity.

If the fluid is the perfect conductor, the electric conductivity should be $\sigma\to\infty$, then induction equation of the magnetic field becomes eq. (3.4). We call this limit of the electric conductivity $\sigma\to\infty$ as ideal Magneto-Hydrodynamical limit.

Lorentz force by the magnetic field is

$$\begin{aligned} \frac{\mathbf{j}\times\mathbf{B}}{c} &= \frac{(\nabla\times\mathbf{B})\times\mathbf{B}}{4\pi} \\ &= -\nabla\left(\frac{B^2}{8\pi}\right) + \frac{1}{4\pi}(\mathbf{B}\cdot\nabla)\mathbf{B}, \end{aligned} \tag{3.14}$$

so that the equation of motion is

$$\rho\left\{\frac{\partial\mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v}\right\} = -\nabla\mathrm{p} - \nabla\left(\frac{B^2}{8\pi}\right) + \frac{1}{4\pi}(\mathbf{B}\cdot\nabla)\mathbf{B}. \tag{3.15}$$

From the equation of motion eq. (3.15) and the mass conservation eq. (3.1), the momentum conservation is derived as eq. (3.2). The energy conservation eq. (3.3) is led by adding the magnetic energy to the total energy and the Poynting flux '$(c/4\pi)\mathbf{E}\times\mathbf{B}$' to the energy flux.

3.4.3 Conservation Forms of Basic Equations

In order to solve basic equations of cylindrical coordinate (3.1) - (3.4), we will rewrite basic equations (3.1) - (3.4) to advection equation in conservation form.

Assuming 'Q' is the physical variable, 'F_r' and 'F_z' are the flow to each direction, and the source term 'S'', we will rewrite advection equation in conservation form with substituting eqs. (3.5) and (3.6) as

$$\frac{\partial Q}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(rF_r) + \frac{\partial}{\partial z}F_z = S', \tag{3.16}$$

eq. 1 so that, for each component, they are rewritten as follows. Eq. (3.1) is

$$\frac{\partial \rho}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(r\rho v_r) + \frac{\partial}{\partial z}(\rho v_z) = 0. \tag{3.17}$$

eq. 2 The radial (R) component of eq. (3.2) is

$$\frac{\partial \rho v_r}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}[r\{\rho v_r^2 + \mathrm{p} + \frac{1}{8\pi}(B_z^2 - B_r^2)\}] + \frac{\partial}{\partial z}(\rho v_r v_z - \frac{1}{4\pi}B_r B_z) = \frac{1}{r}(\mathrm{p} + \frac{B_r^2 + B_z^2}{8\pi}). \tag{3.18}$$

eq. 3 The vertical(Z) component of eq. (3.2) is

$$\frac{\partial \rho v_z}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}\{r(\rho v_z v_r - \frac{B_r B_z}{4\pi})\} + \frac{\partial}{\partial z}(\rho v_z^2 + \mathrm{p} + \frac{B_r^2 - B_z^2}{8\pi}) = 0. \tag{3.19}$$

eq. 4 Eq. (3.3) is

$$\frac{\partial}{\partial t}(e + \frac{B^2}{8\pi}) + \frac{1}{r}\frac{\partial}{\partial r}\{r(e_p v_r + \frac{B_z e_y}{4\pi})\} + \frac{\partial}{\partial z}(e_p v_z - \frac{B_r e_y}{4\pi}) = 0. \tag{3.20}$$

eq. 5 The radial (R) component of eq. (3.4) is

$$\frac{\partial B_r}{\partial t} + \frac{\partial}{\partial z}(-e_y) = 0, \tag{3.21}$$

eq. 6 and its vertical (Z) component is

$$\frac{\partial B_z}{\partial t} + \frac{1}{r}\frac{\partial}{\partial r}(re_y) = 0. \tag{3.22}$$

eq. 7 Here, we use $e_p = \gamma \mathrm{p}/(\gamma - 1) + \rho v^2/2$ and $e_y = -v_z B_r + v_r B_z$.

3.4.4 Differential Equation

In the MHD supernova remnant module, 2 steps Modified Lax-Wendroff scheme is adopted as the solving scheme of advection equation. In order to apply Modified Lax-Wendroff scheme, equations (eq. (3.16) and below) are re-written to the advection equations in conservation form in Cartesian Coordinate as follows.

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial r}F_r + \frac{\partial}{\partial z}F_z = S' - \frac{F_r}{r}. \tag{3.23}$$

Here, we set $S = S' - F_r/r$ and rewrite 'r' as 'x' in eq. (3.23). Then we have

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}F_x + \frac{\partial}{\partial z}F_z = S. \tag{3.24}$$

Physical Quantity Q	Flux F_x	Flux F_z	Source Term S
ρ	ρv_x	ρv_z	$-\frac{\rho v_x}{x}$
E	$e_p v_x + \frac{B_z e_y}{4\pi}$	$e_p v_z - \frac{B_x e_y}{4\pi}$	$-\frac{e_p v_x + \frac{B_z e_y}{4\pi}}{x}$
ρv_x	$\rho {v_x}^2 + p + \frac{{B_z}^2 - {B_x}^2}{8\pi}$	$\rho v_x v_z - \frac{B_x B_z}{4\pi}$	$\frac{\rho {v_x}^2}{x} + \frac{{B_x}^2}{4\pi x}$
ρv_z	$\rho v_x v_z - \frac{B_x B_z}{4\pi}$	$\rho {v_z}^2 + p + \frac{{B_x}^2 - {B_z}^2}{8\pi}$	$-\frac{\rho v_x v_z - \frac{B_x B_z}{4\pi}}{x}$
B_x	0	$-e_y$	0
B_z	e_y	0	$-\frac{e_y}{x}$

Table 3.2: Physical Quantities Q, Fluxes F_x, F_z and Source Terms S

Applying same procedure of rewriting on eqs. (3.17) - (3.22), and rearranging to each component, we will get as listed in Table 3.2.

Where,

$$e = \rho\varepsilon + \frac{1}{2}\rho v^2, \ \rho\varepsilon = \frac{p}{\gamma - 1}, \tag{3.25}$$

$$\begin{aligned} E &= \rho\varepsilon + \frac{1}{2}\rho v^2 + \frac{B^2}{8\pi} \\ &= \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + \frac{B^2}{8\pi}, \end{aligned} \tag{3.26}$$

$$\begin{aligned} e_p &= e + p = \frac{p}{\gamma - 1} + \frac{1}{2}\rho v^2 + p \\ &= \frac{\gamma p}{\gamma - 1} + \frac{1}{2}\rho v^2, \end{aligned} \tag{3.27}$$

$$e_y = -v_z B_x + v_x B_z. \tag{3.28}$$

Now, we will consider how to solve this 2 dimensional advection equations by differentiating them (here, we adopt 2 order Modified Lax-Wendroff scheme).

In order to solve 2 dimensional scalar advection equation with difference method, we will obtain the value of 'Q' after the time step Δt by replacing differentiations ($\partial Q/\partial r$, $\partial F_x/\partial x$, $\partial F_z/\partial z$) of 2 dimensional scalar advection equations with differences ($\Delta Q/\Delta x$, $\Delta F_x/\Delta x$, $\Delta F_z/\Delta z$). Here, Δ is the operator to take the difference between grids.

Therefore, we regard eq. (3.24) as

$$\frac{\Delta Q}{\Delta t} + \frac{\Delta F_x}{\Delta x} + \frac{\Delta F_z}{\Delta z} = S. \tag{3.29}$$

Then, we will consider how to obtain $Q_{i,j}^{n+1}$ from $Q_{i,j}^{n}$ when subscripts n is time and i and j are grid number of X and Z axes, respectively.

Lax-Wendroff scheme is the difference method based on Taylor expansion, so that it is led as follows for the case of 1 dimensional advection equation.

$$Q_i^{n+1} = Q_i^n + \Delta t \frac{\partial F}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 F}{\partial t^2} + O(\Delta t^3). \tag{3.30}$$

Substituting following equation into second and third terms of right hand side,

$$\frac{\partial Q}{\partial t} = -\frac{\partial F}{\partial t}, \tag{3.31}$$

$$\frac{\partial^2 Q}{\partial t^2} = \frac{\partial^2 F}{\partial t^2}, \tag{3.32}$$

then we obtain

$$Q_i^{n+1} = Q_i^n - \Delta t \frac{\partial F}{\partial x} + \frac{1}{2}\Delta t^2 \frac{\partial^2 F}{\partial x^2} + O(\Delta t^3). \tag{3.33}$$

Approximating space differentials ($\partial Q/\partial x$, $\partial^2 Q/\partial x^2$) with their central differences, we have

$$Q_i^{n+1} = Q_i^n - \frac{1}{2}\Delta t \frac{F_i^n - F_{i-1}^n}{\Delta x} + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2 (F_{i+1}^n - 2F_i^n + F_{i-1}^n). \tag{3.34}$$

This is the Lax-Wendroff scheme. As you can see from above derivation,

Lax-Wendroff scheme is the solver of 2nd order both in time and space. In case of 1 dimensional scalar advection equation, Lax-Wendroff scheme is the same following 2 steps scheme. This 2 steps scheme is called 2 steps Lax-Wendroff scheme.

$$Q_{i+1/2}^{n+1/2} = \frac{Q_{i+1}^n + Q_i^n}{2} - \frac{1}{2}\frac{\Delta t}{\Delta x}(F_{i+1}^n - F_i^n). \tag{3.35}$$

$$Q_i^{n+1} = Q_i^n - \frac{\Delta t}{\Delta x}(F_{i+1/2}^{n+1/2} - F_{i-1/2}^{n+1/2}). \tag{3.36}$$

The 2 steps Modified Lax-Wendroff scheme which is used in CANS is the scheme which is improved this 2 steps Lax-Wendroff. For simplicity, we will describe the 2 steps Modified Lax-Wendroff scheme as a solver of 1 dimensional advection equation.

In this scheme, as 0-th step, physical quantities on the cell (the boundary between grids) is derived from the physical quantities on the grid Q_i^n, and physical quantities after Δt $Q_{i+1/2}^{n+1}$ are computed with the flux on grid points F_i^n, F_{i-1}^n (cf. Table 3.2 and Figure 3.3)

0-th step: Derivation of physical quantities on the cell.

$$\tilde{Q}_{i+1/2}^{n+1} = Q_{i+1/2}^n - \Delta t \frac{F_{i+1}^n - F_i^n}{\Delta x} + \Delta t \cdot S_{i+1/2}^n. \tag{3.37}$$

Hence,

$$\tilde{Q}_{i+1/2}^{n+1} = \frac{Q_{i+1}^n + Q_i^n}{2} - \Delta t \frac{F_{i+1}^n - F_i^n}{\Delta x} + \Delta t \cdot \frac{S_{i+1}^n + S_i^n}{2}. \tag{3.38}$$

Here, physical quantities on the cell are expressed with ˜.

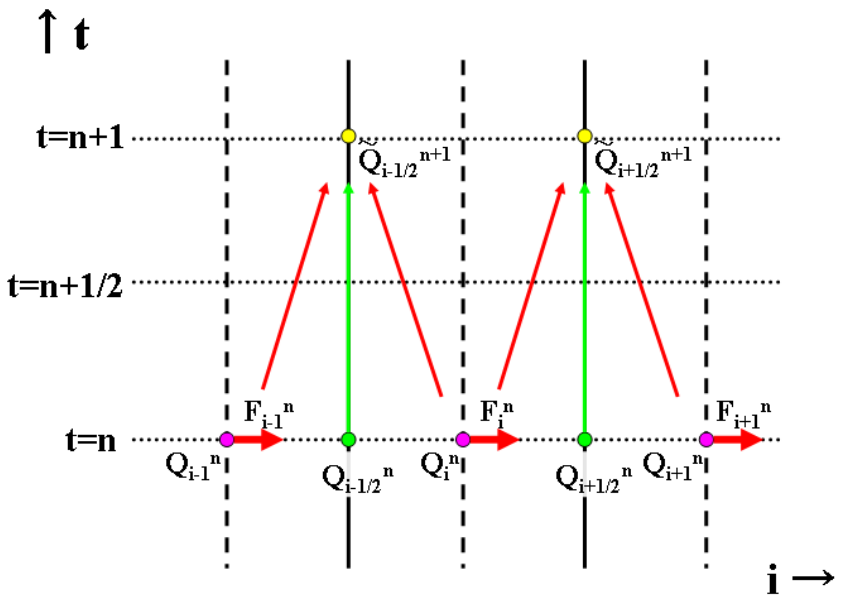

Figure 3.3: 1 Dimension 0-th Step Image

Next as 1-st step, from the central difference of both sides fluxes on grids F_{i-1}^{n}, F_{i+1}^{n} (cf. Table 3.2) are used for computing physical quantities $Q_i^{n+1/2}$ after $\Delta t/2$ has elapsed (cf. Figure 3.4).

1-st step: Central Difference of the Flux.

$$Q_i^{n+1/2} = Q_i^n - \frac{\Delta t}{2}\frac{F_{i+1}^n - F_{i-1}^n}{2\Delta x} + \frac{\Delta t}{2}\cdot S_i^n. \tag{3.39}$$

At last as 2nd step, the quantities Q_i^{n+1} after Δt are calculated by adding the difference of the fluxes on the cell $\tilde{F}_{i+1/2}^{n+1} and \tilde{F}_{i-1/2}^{n+1}$ (they have been derived from physical quantities already calculated at 0-th step $\tilde{Q}$ based on Table 3.2) on to the physical quantities $Q_i^{n+1/2}$ (cf. Figure 3.5)

2-nd step: Central Difference of Flux On the Cell.

$$Q_i^{n+1} = Q_i^{n+1/2} - \frac{\Delta t}{2}\frac{\tilde{F}_{i+1/2}^{n+1} - \tilde{F}_{i-1/2}^{n+1}}{\Delta x} + \frac{\Delta t}{2}\cdot \tilde{S}_i^{n+1}. \tag{3.40}$$

Hence,

$$Q_i^{n+1} = Q_i^{n+1/2} - \frac{\Delta t}{2}\frac{\tilde{F}_{i+1/2}^{n+1} - \tilde{F}_{i-1/2}^{n+1}}{\Delta x} + \frac{\Delta t}{2}\cdot \frac{\tilde{S}_{i+1/2}^{n+1} + \tilde{S}_{i-1/2}^{n+1}}{2}. \tag{3.41}$$

In the similar way, equations expanded into 2 dimensions are described as follows (cf. Figures 3.6, 3.7, 3.8).

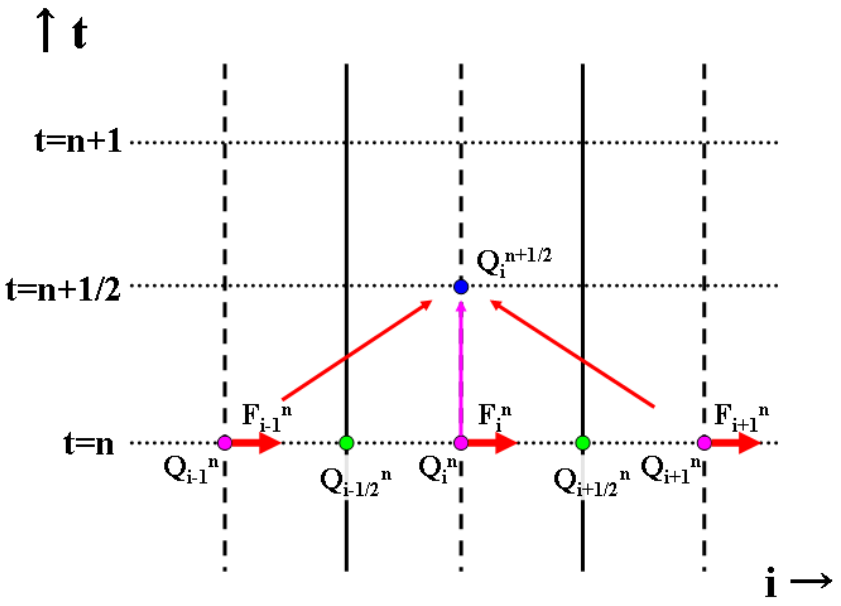

Figure 3.4: 1 Dimension 1-st Step Image

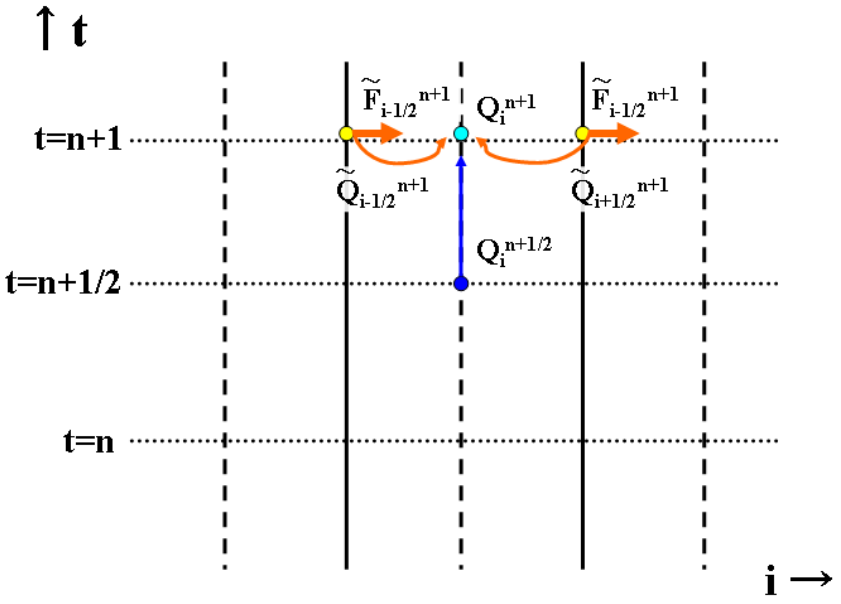

Figure 3.5: 1 Dimensional 2nd Step Image

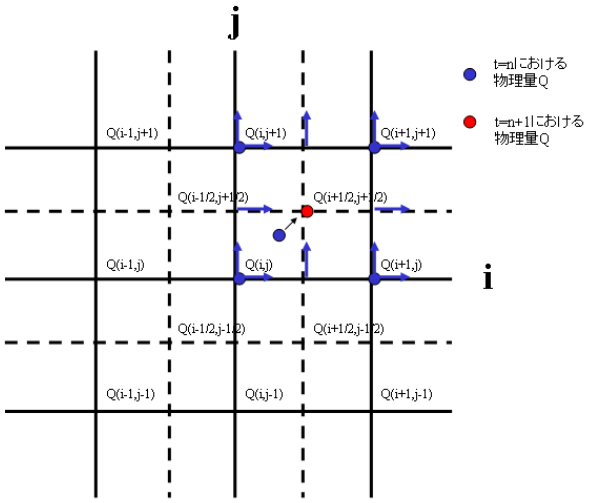

Figure 3.6: 2 Dimensional 0th Step Image

0-th step: Derivation of physical quantities on the cell.

$$\begin{aligned}\tilde{Q}^{n+1}_{i+1/2,j+1/2} &= Q^{n}_{i+1/2,j+1/2} - \Delta t \frac{F^{n}_{i+1,j+1/2} - F^{n}_{i,j+1/2}}{\Delta x} - \Delta t \frac{F^{n}_{i+1/2,j+1} - F^{n}_{i+1/2,j}}{\Delta z} \\ &\quad + \Delta t \cdot S^{n}_{i+1/2,j+1/2}.\end{aligned} \tag{3.42}$$

Hence,

$$\begin{aligned}\tilde{Q}^{n+1}_{i+1/2,j+1/2} &= \frac{1}{2}\left(\frac{Q^{n}_{i+1,j+1} + Q^{n}_{i+1,j}}{2} + \frac{Q^{n}_{i,j+1} + Q^{n}_{i,j}}{2}\right) \\ &\quad - \Delta t \frac{1}{\Delta x}\left(\frac{F^{n}_{i+1,j+1} + F^{n}_{i+1,j}}{2} - \frac{F^{n}_{i,j+1} + F^{n}_{i,j}}{2}\right) \\ &\quad - \Delta t \frac{1}{\Delta z}\left(\frac{F^{n}_{i+1,j+1} + F^{n}_{i,j+1}}{2} - \frac{F^{n}_{i+1,j} + F^{n}_{i,j}}{2}\right) \\ &\quad + \Delta t \cdot \frac{1}{2}\left(\frac{S^{n}_{i,j+1} + S^{n}_{i,j}}{2} + \frac{S^{n}_{i+1,j+1} + S^{n}_{i+1,j}}{2}\right).\end{aligned} \tag{3.43}$$

1-st step: Central Difference of the Flux.

$$\begin{aligned}Q^{n+1/2}_{i,j} &= Q^{n}_{i,j} - \frac{\Delta t}{2}\frac{F^{n}_{i+1,j} - F^{n}_{i-1,j}}{2\Delta x} - \frac{\Delta t}{2}\frac{F^{n}_{i,j+1} - F^{n}_{i,j-1}}{2\Delta z} \\ &\quad + \frac{\Delta t}{2} \cdot S^{n}_{i,j}.\end{aligned} \tag{3.44}$$

2-nd step: Central Difference of Flux On the Cell.

$$\begin{aligned}Q^{n+1}_{i,j} &= Q^{n+1/2}_{i,j} - \frac{\Delta t}{2}\frac{\tilde{F}^{n+1}_{i+1/2,j} - \tilde{F}^{n+1}_{i-1/2,j}}{\Delta x} - \frac{\Delta t}{2}\frac{\tilde{F}^{n+1}_{i,j+1/2} - \tilde{F}^{n+1}_{i,j-1/2}}{\Delta z}. \\ &\quad + \frac{\Delta t}{2} \cdot \tilde{S}^{n+1}_{i,j}.\end{aligned} \tag{3.45}$$

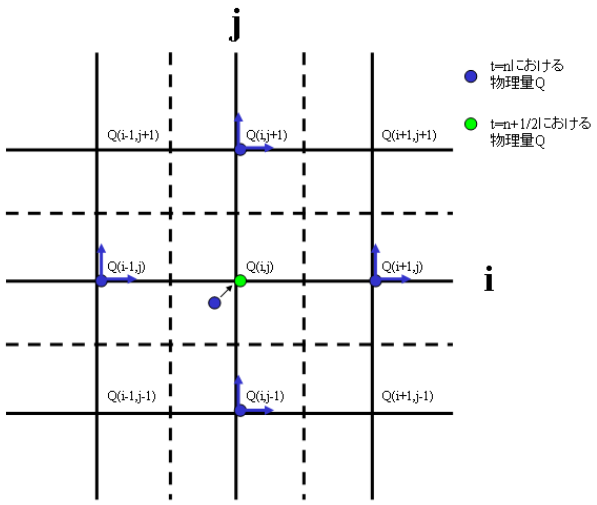

Figure 3.7: 2 Dimensional 1-st Step Image

Hence,

$$
\begin{aligned}
Q_{i,j}^{n+1} \;=\; & Q_{i,j}^{n+1/2} \\
& -\frac{\Delta t}{2}\frac{1}{\Delta x}\left(\frac{\tilde{F}_{i+1/2,j+1/2}^{n+1}+\tilde{F}_{i+1/2,j-1/2}^{n+1}}{\Delta 2}-\frac{\tilde{F}_{i-1/2,j+1/2}^{n+1}+\tilde{F}_{i-1/2,j-1/2}^{n+1}}{\Delta 2}\right) \\
& -\frac{\Delta t}{2}\frac{1}{\Delta z}\left(\frac{\tilde{F}_{i+1/2,j+1/2}^{n+1}+\tilde{F}_{i-1/2,j+1/2}^{n+1}}{\Delta 2}-\frac{\tilde{F}_{i+1/2,j-1/2}^{n+1}+\tilde{F}_{i-1/2,j-1/2}^{n+1}}{\Delta 2}\right) \\
& +\frac{\Delta t}{2}\cdot\frac{1}{2}\left(\frac{\tilde{S}_{i+1/2,j+1/2}^{n+1}+\tilde{S}_{i-1/2,j+1/2}^{n+1}}{2}+\frac{\tilde{S}_{i+1/2,j-1/2}^{n+1}+\tilde{S}_{i-1/2,j-1/2}^{n+1}}{2}\right).
\end{aligned}
\tag{3.46}
$$

3.4.5 Introduction of the Artificial Viscosity

Though the Modified Lax-Wendroff scheme is the second order scheme in both time and space, it has a defect that at the discontinuity it may create numerical instability. Details are easily found in many textbook of numerical simulation (see e.g., 'Numerical Simulation of Hydrodynamics', (1994) Univ. Tokyo Press, K. Fujii). In order to avoid this numerical instability, the term of artificial viscosity is inserted in every basic equation as a diffusion term. The diffusion term normally has diffusion coefficient κ and is given as $\kappa\nabla^2 Q$ with using physical quantity Q.

The artificial viscosity diffusion coefficients (κ_x, κ_z) used in CANS have defined using Q_v as a parameter in order to have large values at the velocity discontinuity plane,

$$
(\kappa_x, \kappa_z) = Q_v\left(\left|\frac{\partial v_x}{\partial x}\right|, \left|\frac{\partial v_z}{\partial z}\right|\right). \tag{3.47}
$$

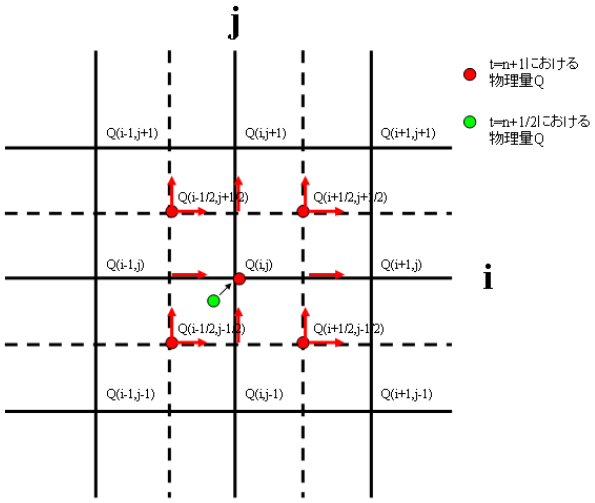

Figure 3.8: 2 Dimensional 2nd Step Image

Then, basic equations with diffusion term in CANS are written with physical quantities Q of each equations as follows.

$$\frac{\partial Q}{\partial t} + \frac{\partial}{\partial x}F_x + \frac{\partial}{\partial z}F_z = S + \left(\frac{\partial}{\partial x}(\kappa_x \nabla Q) + \frac{\partial}{\partial z}(\kappa_z \nabla Q)\right). \tag{3.48}$$

3.4.6 Computational Engine - mlw_m_c.f

Computational engine for Modified Lax-Wendroff scheme is in '/cans2d/hdmlw' directory. It consists of subroutines of which corresponds each step of 2 steps Modified Lax-Wendroff scheme; 'mlwhalf.f', 'mlwsrch.f','mlwfull.f','mlwsrcf.f', and the subroutine to calculate artificial viscosity; 'mlwartv.f'. The 0-th and 1-st steps are computed in 'mlwhalf.f' and 'mlwsrch.f', and the 2-nd step is computed in 'mlwfull.f' and 'mlwsrcf.f'. In the previous section, we described the procedure of the Modified Lax-Wendroff scheme, we will describe the contents of the actual program 'mlw_m_c.f' here.

```
c=======================================================================|
      subroutine mlw_m_c(ro,pr,vx,vz,bx,bz,ay
     &                  ,dt,qav,gm,x,xm,dx,dxm,ix,dz,dzm,jx)
! 'ro' is density, 'pr' is pressure, 'vx' is velocity in X-direction
! 'vz' is velocity in Z-direction, 'bx' is magnetic field in X-direction,
! 'bz' is magnetic field in Z-direction, 'ay' is vector potential.
c=======================================================================|
      implicit double precision (a-h,o-z)
      dimension dx(ix),dxm(ix)
      dimension dxi(ix),dxim(ix)
      dimension ux0(ix),ux1(ix)
      dimension x(ix),xm(ix)
      dimension dz(jx),dzm(jx)
      dimension dzi(jx),dzim(jx)
      dimension uz0(jx),uz1(jx)
      dimension ro(ix,jx),pr(ix,jx),vx(ix,jx),vz(ix,jx)
     &         ,bx(ix,jx),bz(ix,jx)
      dimension ey(ix,jx)
      dimension ay(ix,jx)
      dimension ee(ix,jx),rx(ix,jx),rz(ix,jx)
      dimension roh(ix,jx),eeh(ix,jx),rxh(ix,jx),rzh(ix,jx)
     &         ,bxh(ix,jx),bzh(ix,jx)
      dimension eyh(ix,jx)
      dimension ayh(ix,jx)
      dimension prh(ix,jx),vxh(ix,jx),vzh(ix,jx)
      dimension dro(ix,jx),dee(ix,jx),drx(ix,jx),drz(ix,jx)
     &         ,dbx(ix,jx),dbz(ix,jx)
     &         ,day(ix,jx)
      dimension fx(ix,jx),qx(ix,jx)
      dimension fz(ix,jx),qz(ix,jx)
      dimension ss(ix,jx)
c-----------------------------------------------------------------------|
      pi = acos(-1.0d0)
      pi8i=1./pi/8.
      pi4i=1./pi/4.
c-----------------------------------------------------------------------|
c     ready
c-----------------------------------------------------------------------|
      do i=1,ix
         dxi(i) = 1.0/dx(i)
         dxim(i) = 1.0/dxm(i)
      enddo
! define the inverses of dx and dxm
! both dx and dxm are created in 'grdrdy.f'.

      do i=2,ix-1
         ux1(i)  = 0.5*dxm(i-1)/dx(i)
         ux0(i)  = 0.5*dxm(i)/dx(i)
      enddo
! ux1(i) and ux0(i) will be used in 'mlwfull.f' and 'mlwsrcf.f'.
      do j=1,jx
         dzi(j)  = 1.0/dz(j)
         dzim(j) = 1.0/dzm(j)
      enddo
      do j=2,jx-1
         uz1(j)  = 0.5*dzm(j-1)/dz(j)
         uz0(j)  = 0.5*dzm(j)/dz(j)
      enddo
! same as in X-direction.
```

```
c-----------------------------------------------------------------------|
c     initialize dro etc.
c-----------------------------------------------------------------------|
      do j=1,jx
      do i=1,ix
         dro(i,j)= 0.0
         dee(i,j)= 0.0
         drx(i,j)= 0.0
         drz(i,j)= 0.0
         dbx(i,j)= 0.0
         dbz(i,j)= 0.0
         day(i,j)= 0.0
      enddo
      enddo
! 'dro' etc. are the variation of the quantities (e.g., 'ro') from time 'n' to 'n+1'.
! Variables which will be calculated later should be initial zero cleared here.
c-----------------------------------------------------------------------|
c     calculate energy from pressure
c-----------------------------------------------------------------------|
      do j=1,jx
      do i=1,ix
         vv=vx(i,j)**2+vz(i,j)**2
         bb=bx(i,j)**2+bz(i,j)**2
         ee(i,j) = pr(i,j)/(gm-1)+0.5*ro(i,j)*vv+pi8i*bb
         rx(i,j) = ro(i,j)*vx(i,j)
         rz(i,j) = ro(i,j)*vz(i,j)
      enddo
      enddo
! physical quantities needed are calculated from 'ro', 'pr', 'vx', 'vz', 'bx', and 'bz'.
! 'ee' is the total energy which corresponds 'Q' in energy conservation equation.
! 'rx' and 'rz' are the momentum which correspond 'Q' in momentum conservation equation.
! Though 'rx' and 'rz' also correspond fluxes 'Fx' and 'Fz' in mass conservation
! equation, at mass conservation equation calculated below these fluxes are calculated
! from 'ro*vx' and 'ro*vz' instead of 'rx' and 'rz'.     Refer Table 1.2.
      do j=1,jx
      do i=1,ix
         ey(i,j) = -vz(i,j)*bx(i,j)+vx(i,j)*bz(i,j)
      enddo
      enddo
! ey is the quantity to be used to compute the flux in energy conservation equation and
! also used as the flux in induction equation of the magnetic field.  Refer Table 1.2.
c-----------------------------------------------------------------------|
c     step intermediate results for flux calculation
c-----------------------------------------------------------------------|
! from here, 0-th and 1-st steps computation start.
c---  density ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vx(i,j)
         fz(i,j)= ro(i,j)*vz(i,j)
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! 'fx' is the flux Fx, 'fz' is the flux Fz, 'ss' means source term.
! Fluxes Fx and Fz and source term S of mass conservation equation
! are calculated here. Refer Table 1.2.
      call mlwhalf(ro ,roh ,dro,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(roh ,dro ,dt,ss,ix,jx)
```

```
! The calculation of the 0-th and 1-st steps for physical quantity 'ro'
! which corresponds eq. (1.43) and (1.44) are done here.
! Pay attention that at 0-th step 'roh' is the physical quantity on the cell updated
! after time span 'dt' and that at 1-st step 'dro' is the difference of physical
! quantity in time in order to update time span 'dt'/2.
c---  energy ---
      do j=1,jx
      do i=1,ix
         vv=vx(i,j)**2+vz(i,j)**2
! 'vv' is used for calculating 'ep'.
         ep    = pr(i,j)*gm/(gm-1.)+0.5*ro(i,j)*vv
! 'ep' is used for calculating flux 'Fx' and 'Fz' in energy conservation equation.
! Refer Table 1.2.
         fx(i,j)= ep*vx(i,j) +(bz(i,j)*ey(i,j))*pi4i
         fz(i,j)= ep*vz(i,j) +(-bx(i,j)*ey(i,j))*pi4i
         ss(i,j)= -fx(i,j)/x(i)
! 'ss' is the source term which corresponds Table 1.2.
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S'.  Refer Table 1.2.
      call mlwhalf(ee ,eeh ,dee ,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(eeh ,dee ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'eeh' is the physical quantity
! on the cell updated after time span 'dt' and that at 1-st step 'dee' is the difference
! of physical quantity in time in order to update time span 'dt'/2.
c---  x-momentum ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vx(i,j)**2+pr(i,j)
     &          +pi8i*(bz(i,j)**2-bx(i,j)**2)
         fz(i,j)= ro(i,j)*vx(i,j)*vz(i,j)-pi4i*bx(i,j)*bz(i,j)
         ss(i,j)= -(ro(i,j)*(vx(i,j)**2)
     &              +pi4i*(-bx(i,j)**2))/x(i)
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S' of X-direction.  Refer Table 1.2.

      call mlwhalf(rx,rxh,drx,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rxh ,drx ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'rxh' is the physical quantity
! on the cell updated after time span 'dt', and that at 1-st step 'drx' is the difference
! of physical quantity in time in order to update time span 'dt'/2.
c---  z-momentum ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ro(i,j)*vz(i,j)*vx(i,j)-pi4i*bz(i,j)*bx(i,j)
         fz(i,j)= ro(i,j)*vz(i,j)**2+pr(i,j)
     &          +pi8i*(bx(i,j)**2-bz(i,j)**2)
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! calculation of fluxes 'Fx' and 'Fz' and source term 'S'
! of Z-direction.  Refer Table 1.2.
      call mlwhalf(rz,rzh,drz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rzh ,drz ,dt,ss,ix,jx)
        ! Be careful about that at 0-th step 'rzh' is the physical quantity
    ! on the cell updated after time span 'dt', and that at 1-st step 'drz' is the
    ! difference of physical quantity in time in order to update time span 'dt'/2.
```

```
c---  x-magnetic ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= 0.
         fz(i,j)=-ey(i,j)
      enddo
      enddo
! calculate fluxes 'Fx' and 'Fz' in induction equation of the magnetic field of X-direction.
! Refer Table 1.2.
! Because the source term is 0 so that it is not calculated here.
      call mlwhalf(bx,bxh,dbx,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
! Be careful about that at 0-th step 'bxh' is the physical quantity on the cell updated
! after time span 'dt' and that at 1-st step 'dbx' is the difference of physical
! quantity in time in order to update time span 'dt'/2.
! Because the source term is 0 so that 'mlwsrch.f' is not used.
c---  z-magnetic ---
      do j=1,jx
      do i=1,ix
         fx(i,j)= ey(i,j)
         fz(i,j)= 0.
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! calculate fluxes 'Fx' and 'Fz' and source term 'S' in induction equation of the
! magnetic field of X-direction.  Refer Table 1.2.

      call mlwhalf(bz,bzh,dbz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(bzh ,dbz ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'bzh' is the physical quantity
! on the cell updated after time span 'dt' and that at 1-st step 'dbz' is the difference
! of physical quantity in time in order to update time span 'dt'/2.

c---  y-magnetic potential ---
      do j=1,jx
      do i=1,ix
         ss(i,j)= -ey(i,j)
      enddo
      enddo
      call mlwsrch(ayh ,day ,dt,ss,ix,jx)
! calculate vector potential
! 'ss(i,j)= -ey(i,j)' is easily derived from vector potential and induction equations.

c-----------------------------------------------------------------------|
c     convert from total energy to pressure
c-----------------------------------------------------------------------|
      do j=1,jx-1
      do i=1,ix-1
         vxh(i,j)   = rxh(i,j)/roh(i,j)
         vzh(i,j)   = rzh(i,j)/roh(i,j)
         vv=vxh(i,j)**2+vzh(i,j)**2
         bb=bxh(i,j)**2+bzh(i,j)**2
         prh(i,j)   = (gm-1)*(eeh(i,j)-0.5*roh(i,j)*vv-pi8i*bb)
      enddo
      enddo      call mlwhalf(rz,rzh,drz,dt,fx,dxi,dxim,ix,fz,dzi,dzim,jx)
      call mlwsrch(rzh ,drz ,dt,ss,ix,jx)
! Be careful about that at 0-th step 'rzh' is the physical quantity
! on the cell updated after time span 'dt', and that at 1-st step 'drz' is the
! difference of physical quantity in time in order to update time span 'dt'/2.
```

```
! Physical quantities such as 'vxh', 'vzh' and 'prh' of which have not calculated from
! 'dt' updated variables  'roh', 'eeh', 'rxh', 'rzh', 'bxh', and 'bzh'.
! With this procedure, we can calculate the flux on the cell at 'dt' updated.
      do j=1,jx
      do i=1,ix
        eyh(i,j)=-vzh(i,j)*bxh(i,j)+vxh(i,j)*bzh(i,j)
      enddo
      enddo
! 'eyh' corresponds 'ey' in the 1-st step.
! This quantity is used in calculating the flux of energy conservation and also used
! as the flux in induction equation of the magnetic field.  Refer table 1.2.
c-----------------------------------------------------------------------|
c     step intermediate results for full step
c-----------------------------------------------------------------------|
! From here, 2-nd step calculation starts.
c---  density  ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vxh(i,j)
         fz(i,j)= roh(i,j)*vzh(i,j)
         ss(i,j)= -fx(i,j)/xm(i)
      enddo
      enddo
! 'fx' is flux Fx, 'fz' is flux Fz, and 'ss' is the source term.
! Calculation of fluxes 'Fx' and 'Fz' and source term 'S' in
! mass conservation equation.  Refer table 1.2.
      call mlwfull(dro ,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dro,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'roh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dro'.  Therefore, if we add this 'dro'
! to 'ro', time step is updated 'dt' from 'n' to 'n+1'.
c---  energy     ---
      do j=1,jx-1
      do i=1,ix-1
         vv=vxh(i,j)**2+vzh(i,j)**2
! 'vv' is used for calculating 'ep'.
         ep    = prh(i,j)*gm/(gm-1.)+0.5*roh(i,j)*vv
! 'ep' is used for calculating fluxes 'Fx' and 'Fz' in energy
! conservation equation.  Refer table 1.2.
         fx(i,j)= ep*vxh(i,j)
     &            +(bzh(i,j)*eyh(i,j))*pi4i
         fz(i,j)= ep*vzh(i,j)
     &            +(-bxh(i,j)*eyh(i,j))*pi4i
         ss(i,j)= -fx(i,j)/xm(i)
! 'ss' is the source term which corresponds table 1.2.
      enddo
      enddo
! 'fx' is flux Fx, 'fz' is flux Fz, and 'ss' is the source term.
! Calculation of fluxes 'Fx' and 'Fz' and source term 'S' in
! energy conservation equation.  Refer table 1.2.

      call mlwfull(dee ,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dee,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'eeh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dee'.  Therefore, if we add this 'dee'
! to 'ee', time step is updated 'dt' from 'n' to 'n+1'.
```

```
c---  x-momentum ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vxh(i,j)**2+prh(i,j)
     &          +pi8i*(bzh(i,j)**2-bxh(i,j)**2)
         fz(i,j)= roh(i,j)*vxh(i,j)*vzh(i,j)-pi4i*bxh(i,j)*bzh(i,j)
         ss(i,j)= -(roh(i,j)*(vxh(i,j)**2)
     &              +pi4i*(-bxh(i,j)**2))/xm(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in
! the momentum equation of X-direction. Refer table 1.2.

      call mlwfull(drx,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(drx,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'rxh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'drx'.  Therefore, if we add this 'drx'
! to 'rx', time step is updated 'dt' from 'n' to 'n+1'.
c---  z-momentum ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= roh(i,j)*vzh(i,j)*vxh(i,j)-pi4i*bzh(i,j)*bxh(i,j)
         fz(i,j)= roh(i,j)*vzh(i,j)**2+prh(i,j)
     &          +pi8i*(bxh(i,j)**2-bzh(i,j)**2)
         ss(i,j)= -fx(i,j)/xm(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in the momentum equation
! of Z-direction.  Refer table 1.2.
      call mlwfull(drz,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(drz,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'rzh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'drz'.  Therefore, if we add this 'drz'
! to 'rz', time step is updated 'dt' from 'n' to 'n+1'.
c---  x-magnetic ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= 0.
         fz(i,j)= -eyh(i,j)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' in the induction equation of the magnetic field of
! Z-direction. Refer table 1.2.  Because the source term is 0 and is not calculated.
      call mlwfull(dbx,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
! For physical quantity 'bxh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dbx'.  Therefore, if we add this 'dbx'
! to 'bx', time step is updated 'dt' from 'n' to 'n+1'.
c---  z-magnetic ---
      do j=1,jx-1
      do i=1,ix-1
         fx(i,j)= eyh(i,j)
         fz(i,j)= 0.
         ss(i,j)= -fx(i,j)/x(i)
      enddo
      enddo
! Calculate fluxes 'Fx' and 'Fz' and the source term 'S' in the induction equation
! of the magnetic field of Z-direction.  Refer table 1.2.
```

```
      call mlwfull(dbz,dt,fx,dxi,ux0,ux1,ix,fz,dzi,uz0,uz1,jx)
      call mlwsrcf(dbz,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! For physical quantity 'bzh', 2nd step calculation for eq. (1.46) is done here.
! Be careful on that 'time difference of the physical quantity' from already updated
! 'dt'/2 value to be updated again 'dt'/2 is 'dbz'.  Therefore, if we add this 'dbz'
! to 'bz', time step is updated 'dt' from 'n' to 'n+1'.
c---  y-magnetic potential ---
      do j=1,jx
      do i=1,ix
         ss(i,j)= -eyh(i,j)
      enddo
      enddo
      call mlwsrcf(day,dt,ss,ux0,ux1,ix,uz0,uz1,jx)
! calculate vector potential
! 'ss(i,j)= -eyh(i,j)' is derived from vector potential and the induction equations.
c-----------------------------------------------------------------------|
c     diffusion coefficients for artificial viscosity
c-----------------------------------------------------------------------|
c     qav=3.0
      zero=0.0
      do j=1,jx-1
      do i=1,ix-1
         qx(i,j)=qav*dxm(i)*max(zero,abs(vx(i+1,j)-vx(i,j))-1.0e-4)
      enddo
      enddo
      do j=1,jx-1
      do i=1,ix
         qz(i,j)=qav*dzm(j)*max(zero,abs(vz(i,j+1)-vz(i,j))-1.0e-4)
      enddo
      enddo
! Here, diffusion coefficients for artificial viscosity is calculated.  Please refer
! section 1.4.5 Introduction of the artificial viscosity and equation (1.47)
c-----------------------------------------------------------------------|
c     apply artificial viscosity
c-----------------------------------------------------------------------|
      call mlwartv(ro,dro,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(ee,dee,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(rx,drx,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(rz,drz,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(bx,dbx,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
      call mlwartv(bz,dbz,dt,qx,dxi,dxim,ix,qz,dzi,dzim,jx)
! The amount of artificial viscosity is calculated in the subroutine 'mlwartv.f' and
! added on to time differences of all physical quantities.
! Please refer section 1.4.5 Indtoduction of the artificail viscosity and eq. (1.47)
c-----------------------------------------------------------------------|
c     update internal points
c-----------------------------------------------------------------------|
      do j=2,jx-1
      do i=2,ix-1
         ro(i,j) = ro(i,j) +dro(i,j)
         ee(i,j) = ee(i,j) +dee(i,j)
         rx(i,j) = rx(i,j) +drx(i,j)
         rz(i,j) = rz(i,j) +drz(i,j)
         bx(i,j) = bx(i,j) +dbx(i,j)
         bz(i,j) = bz(i,j) +dbz(i,j)
         ay(i,j) = ay(i,j) +day(i,j)
      enddo
      enddo
! The differences calculated through steps from 0 to 2 are added on the previous
! quantities (at time $t=n$) and update the time 'dt'.
```

```
c-------------------------------------------------------------------------|
c     convert from total energy to pressure
c-------------------------------------------------------------------------|
      do j=2,jx-1
      do i=2,ix-1
         vx(i,j) = rx(i,j)/ro(i,j)
         vz(i,j) = rz(i,j)/ro(i,j)
         vv=vx(i,j)**2+vz(i,j)**2
         bb=bx(i,j)**2+bz(i,j)**2
         pr(i,j) = (gm-1)*(ee(i,j) - 0.5*ro(i,j)*vv - pi8i*bb)
      enddo
      enddo
! At last, using 'dt' updated variables 'ro', 'ee', 'rx', 'rz', 'bx', and 'bz',
! new quantities such as 'vx', 'vz', and 'pr' are calculated.
! With this procedure, all physical quantities on the grid point
! have updated to time span 'dt'.
      return
      end
```

3.5 Acknowledgment

For this chance I would thank Prof. T. Hanawa at Chiba University. My Ph. D. adviser Prof. K. Tomisaka at NAOJ recommends me to attend the "Summer School for Astro and Space Plasma Simulation" and looks after the preparation of this draft so that I would express my deep gratitude to him. The developers of CANS, Prof. R. Matsumoto at Chiba University and Prof. T. Yokoyama at University of Tokyo gave me many comments throughout this manual. Prof. T. Matsumoto at Hosei University, Dr. K. Saigo at NAOJ, and Dr. M. Machida at Chiba University gave me valuable comments on the computation method. And I thank all co-authors who cooperate during revision and peoples of Astrophysical laboratory at Chiba University.

Bibliography

[1] Space Hydrodynamics (1997) Baifukan, Shiro Sakashita and Satoru Ikeuchi

[2] Computation Method for Hydrodynamics (1994) Univ. Tokyo Press, Kouzou Fujii

[3] Text of Numerical Astrophysics Summer School (2000) Koji Tomisaka and Tomoyuki Hanawa

[4] Hydrodynamics (1972) Baifukan, Tomomasa Tatsumi

Chapter 4

List of Group Projects

Attendants will choose one of the following projects and carry out simulations under the supervision of the instructor. We expect 5 – 6 members for each project. The afternoon session on Friday is allocated for the presentation of achievements of each group.

1. Cloud Collision with the Galactic Gas Disk

   T. Kudoh (NAOJ)

   You will perform 2D-axisymmetric hydrodynamic or magnetohydrodynamic simulations to study the impact of the cloud with the Galactic disk. One of the motivations of this simulation is to explain the mushroom- shaped cloud of the Galactic worm candidate GW 123.4-1.5. Kudoh & Basu (2004) have recently performed the hydrodynamic simulation with adiabatic gas and reproduced the mushroom-shaped structure of the gas. You will perform the similar simulations with cooling effect of the gas or the magnetic field, both of which are probably not negligible in the real Galactic gas. You will research how different shapes you get when you include these effects. You could also perform the similar simulations with the 2D-Cartesian coordinate. You may think about the limitations of 2D-axisymmetric or 2D-Cartesian simulations.

   Reference
   "A mushroom-shaped structure from the impact of a cloud with the Galactic disk"
   Kudoh, T. & Basu, S., 2004 Astron. Astrophys., 423, 183

2. Hot Gas around Moving Clusters of Galaxies

   N. Fukuda (Okayama Univ. of Science) and N. Asai (Chiba Univ.)

   Chandra observations revealed the detailed structure in hot gas around moving clusters of galaxies (e.g. cold front and tail). We will model the gas around the clusters in hot intergalactic medium and simulate it including the heat conduction.

3. Magnetic Reconnection (Stability of Craig-Henton Solution)

   M. Oka and T. Miyagoshi (Kwasan Observatory, Kyoto Univ.)

   Magnetic reconnection is an important energy-release process in the universe and is considered to be responsible for solar flares and substorms in the Earth's magnetosphere. The analytic solution of magnetic reconnection has been found quite recently (Craig & Henton, 1995), but its non-linear evolution and stability are still unclear. Then, in this group, we

will perform parametric survey of the Craig-Henton solution with a help of numerical simulation. Preliminary results have been already obtained by Hirose et al. 2004 and previous simulation school assignments of the years of 2002 and 2004. We will extend these results and discuss their physical meaning.

4. Nonlinear Evolution of the Magnetic Buoyancy Instability

   S. Nozawa (Ibaraki Univ.)

   By means of magnetohydrodynamic simulations, we study nonlinear evolutions of the magnetic buoyancy instability such as the Parker instability in a gravitationally stratified magnetized gas layer. You will set up initial conditions for the Galaxy or for the Sun, and simulate the growth of the instability. We discuss how the magnetic loops created by the instability will actually be observed.

5. Magnetorotational Instability in Accretion Disks

   M. Machida (NAOJ) and R. Matsumoto (Chiba Univ.)

   We study the effects of resistivity, initial configuration of magnetic fields, vertical gravity, and radiative cooling on the growth of the magnetorotational instability. Both local simulations and global simulations will be assigned. When the radiative cooling is included, the disk will shrink in the vertical direction when the density exceeds some critical value. We will explore the possibility that magnetic energy is released explosively in such disks.

6. Relativistic hydrodynamic/MHD simulations

   S. Koide (Toyama Univ.)

   We plan to perform relativistic hydrodynamic/MHD simulations with CANS and general relativistic MHD code. The distinct theme of the simulations are as follows:

   - Relativistic shock tube problems with CANS
   - Interaction of relativistic flow and magnetic field with CANS
   - Stability of a rotating disk around a black hole
   - Plasma falling into a black hole
   - Interaction of plasma and magnetic field around rapidly rotating black hole

7. Disk Flare Model

   K.E. Nakamura (Matsue National College of Technology)

   The disk flare model successfully explains strong X-ray activities of young stellar objects. According to this model, magnetic field loops connecting a central star and its surrounding disk are twisted by the disk rotation. The strongly wound magnetic loops expand and change their initial dipole configuration to an open one. Since a current sheet is formed inside the expanding loop, magnetic reconnection occurs in the presence of resistivity. A large amount of energy is released by the magnetic reconnection. We will study the influence of the disk flare on the disk structure by using the 2-dimensional MHD code including heat conduction. Our aim is to simulate the evaporation of cool disk plasma to the hot corona.