Chapter 1

Introduction

Many astrophysical scenarios are modeled using the field equations of fluid dynamics. Fluids are generally challenging systems to describe analytically, as they form a nonlinear coupled set of field equations. Most interesting flows are extremely complicated, and can potentially have structure on all scales. As such, studies of astrophysical fluid dynamics often involve high-resolution integrations of the field equations on a computer.

In order to get the most out of every CPU-hour, applied mathematicians, physicists and astronomers have developed increasingly accurate numerical methods for performing these calculations, designed for various types of problems. Some methods provide high order of accuracy (e.g. WENO [69], spectral methods), others added stability (implicit methods). Some are designed to resolve a large number of scales (adaptive mesh refinement, [30]), and others are designed for adaptivity and geometric flexibility (ALE methods, [96]). As is generally the case, the optimal numerical method will always depend on the physical set-up of the problem being solved. For example, if the flow is supersonic or even transsonic, strong shocks may form which cannot be resolved at higher than first-order accuracy. In this case, for example, it may not gain the user much to use a very high-order method.

My thesis will concentrate on a very new numerical scheme which is well-suited to extremely supersonic flows, and flows which are sensitive to the accurate preservation of contact discontinuities. The idea is to use a moving mesh to follow the fluid flow. This general idea is similar to Lagrangian methods, such as smoothed particle hydrodynamics [115], arbitrary Lagrangian-Eulerian [96], or mesh-less schemes. However, the core of the numerical scheme employs high-resolution shock-capturing Godunov-type methods which are designed for high-accuracy fixed-grid codes [36].

The practical result of this scheme is higher accuracy, greater stability, and much higher efficiency than a typical fixed-grid scheme, for certain very important problems in astrophysics. The goal of this thesis will be to outline the numerical method, and describe some of the scientific results we have been able to obtain using this scheme in two particular important types of astrophysical flows – namely, disks and jets.

1.1 Eulerian Shock-Capturing Methods

Before describing the moving-mesh approach, I will quickly review the basics of Godunov-type shock-capturing schemes. It turns out that there are only a few modifications necessary to transform a fixed-mesh code into a moving-mesh code. Therefore, once we have reviewed the Eulerian technique, it will be a simple matter to upgrade the method to a moving-mesh context.

1.1.1 The Integral Form

In general, the equations we wish to solve can always be expressed in the following conservation-law form:

$$\partial_t u_\alpha + \nabla \cdot \vec{F}_\alpha = S_\alpha, \tag{1.1}$$

where α is an index which runs over the various conserved quantities (e.g. mass, energy, momentum). The canonical example we will use to describe the method will be Euler's equations:

$$\begin{aligned}
\partial_t(\rho) + &\quad \partial_i(\rho v_i) &= 0 \\
\partial_t(\rho v_j) + &\quad \partial_i(\rho v_i v_j + P\delta_{ij}) &= 0 \\
\partial_t(\tfrac{1}{2}\rho v^2 + \epsilon) + &\quad \partial_i((\tfrac{1}{2}\rho v^2 + \epsilon + P)v_i) &= 0
\end{aligned} \tag{1.2}$$

where ρ is the mass density, v is velocity, P is pressure and ϵ is the internal energy density, generally given by an equation of state $\epsilon = \epsilon(P, \rho)$. We will sometimes assume an adiabatic equation of state,

$$P = \epsilon(\gamma - 1) \tag{1.3}$$

but most of what is described here is independent of the specific set of field equations under consideration.

The first golden rule of shock-capturing is that we must assume there are some regions of the domain in which the fluid quantities jump discontinuously, in which case the derivative in (2.1) may not be well-defined. Since our code can never assume the derivative is well-defined, we must express the field equations in their integral form. To this end, we integrate (2.1) over a small spacetime volume and apply Gauss' law and the fundamental theorem of calculus:

$$\int dV(u_\alpha(t+dt) - u_\alpha(t)) + \int dt d\vec{A} \cdot \vec{F}_\alpha = \int dV dt S_\alpha \tag{1.4}$$

By writing the equations this way, our code never has to take a derivative. Now, if we define M_i^n as the amount of conserved quantity (e.g. mass) in zone i at timestep n, and we define $F_{ij}^{n+1/2}$ as the time-averaged and area-averaged flux through the interface between zone i and zone j, and define $\bar{S}_i^{n+1/2}$ as the volume-averaged and time-averaged source term in zone i, then we can re-express the conservation law as follows, assuming we have integrated over a spacetime volume given by the timestep Δt and the volume of a zone ΔV:

$$M_i^{n+1} = M_i^n + \Delta t \left(-\sum_j F_{ij}^{n+1/2} \cdot \hat{n} \Delta A_{ij} + \bar{S}_i^{n+1/2} \Delta V \right) \tag{1.5}$$

It should be made clear how we interpret this equation. This is an *exact* expression for how to calculate the mass in each zone at timestep $n+1$, given the mass in each zone at timestep n and the time-averaged fluxes and source terms (we use the word "mass" as a proxy for any given conserved quantity). In other words, all numerical approximations are housed in our estimated values of the time-averaged fluxes and source terms. In the case of the flux, the numerical approximations can be broken down into an interpolation in space (from zone-averaged to face-centered quantities) and an extrapolation forward in time so as to determine the time-averaged quantities. For now, we ignore the question of how to interpolate in space and focus on how to produce a time-averaged flux. We assume that we have somehow extrapolated fluid quantities to face ij.

1.1.2 Riemann Solvers

To reiterate, we have reduced our numerical approximation to the following question: assuming we know two fluid states $\{u_L\}$ and $\{u_R\}$ immediately on either side of an interface at the beginning of the timestep, during the course of the timestep what are the time-averaged fluxes $\{F\}$ through this interface? These particular initial conditions can be recognized as a shock tube or "Riemann Problem" (Figure 1.1). For Euler's equations, analytic solutions exist to this problem for all choices of the initial conditions $\{u_L\}$ and $\{u_R\}$, so in principle we can calculate a solution exactly. For more general systems of equations like magnetohydrodynamics, however, we might not know the exact solution, in which case we must make some approximations. Choosing different approximations will give us different solutions of varying accuracy, and ultimately such choices will affect the accuracy and stability of the numerical scheme. Different methods for approximating solutions to the Riemann problem are referred to as different "Riemann Solvers".

1.1.3 The HLL Riemann Solver

One of the simplest Riemann solvers which can be written down is due to Harten, Lax, and Van Leer (HLL). It requires very little knowledge of the underlying equations, and therefore the solver is very easy to adapt to various different

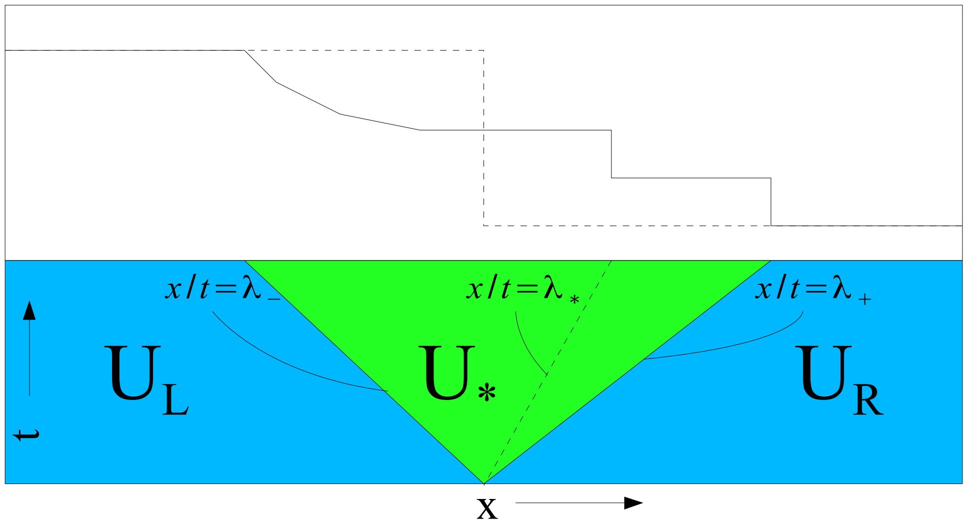

Figure 1.1 A schematic plot of a Riemann Problem. The top panel shows a plot of density at some finite time $t > 0$. The original density field is a step function, shown by the dashed line. A spacetime diagram of the Riemann problem is shown in the lower part of the figure. The blue regions represent parts of the domain which still maintain their initial values. The green ("starred") region is the nontrivial part of the solution, which a Riemann solver attempts to calculate or approximate.

hyperbolic systems.

Figure (1.1) shows a typical solution of a Riemann problem for Euler's equations. Euler's equations have three characteristic waves (in this particular case, the waves are a shockwave, a contact wave, and a rarefaction wave), but any hyperbolic system will have some maximum and minimum wavespeed λ_+, λ_- which defines the region causally connected to the interface at $x = t = 0$ in Figure 1.1 (For Euler's equations, the wavespeeds $\lambda_\pm = v \pm c$ where c is the sound speed). In this figure, the blue regions define parts of spacetime which are causally disconnected from the initial interface, and therefore the solution in one of these regions is simply a constant state equal to the initial conditions. The challenge for a Riemann solver is to find the solution in the green ("starred") region of spacetime. In particular, we wish to know the flux along the curve $x = 0$.

The HLL solver makes the simplified assumption that this starred region is also given by a constant state:

$$u(x,t) = \begin{cases} u_L & x/t < \lambda_- \\ u_* & \lambda_- < x/t < \lambda_+ \\ u_R & x/t > \lambda_+ \end{cases} \tag{1.6}$$

and a similar expression for $F(x,t)$. If it happens, for example, that $\lambda_- > 0$ so that all characteristics are moving to the right, the solution is simply F_L (i.e. we use an upwind flux). If the flow is subsonic, however, the curve $x = 0$ lies between $x/t = \lambda_-$ and $x/t = \lambda_+$. To determine what this "starred" constant state is, we simply enforce the conservation laws. In the spacetime boundaries between regions (given by $x/t = \lambda_\pm$) the flux crossing this characteristic must be a continuous function, otherwise there will be an accumulation of mass along this curve. The statement that the flux in one side is equal to the flux out the other side of the curve can be expressed as a jump condition:

$$[F - \lambda_\pm u] = 0 \tag{1.7}$$

In other words, the quantity $F - \lambda u$ must be continuous. Explicitly,

$$\begin{aligned} F_L - \lambda_- u_L &= F_* - \lambda_- u_* \\ F_R - \lambda_+ u_R &= F_* - \lambda_+ u_* \end{aligned} \tag{1.8}$$

These are two equations which we can solve for the two unknown variables F_* and u_*:

$$\begin{aligned} u_* &= \tfrac{\lambda_+ u_R - \lambda_- u_L - (F_R - F_L)}{\lambda_+ - \lambda_-} \\ F_* &= \tfrac{\lambda_+ F_L - \lambda_- F_R - \lambda_+ \lambda_- (u_L - u_R)}{\lambda_+ - \lambda_-} \end{aligned} \tag{1.9}$$

There are several points to note about this expression. First, as is generally true for approximate Riemann solvers, $F_* \neq F(u_*)$. F_* and u_* are derived specifically assuming continuity of flux; there is no reason to expect that they are consistent with one another. If we had calculated F_* as $F(u_*)$, the the resultant flux would not be consistent with the conservation law, and as a result we lose any guarantee of stability. This point is important, and I plan to bring it up later in the context of moving meshes.

Another point is that this Riemann solver is inherently diffusive. Imagine, for example, the initial conditions are that of a stable contact discontinuity: (ρ, v, P) = $(\rho_L, 0, P_0)$ on the left and $(\rho_R, 0, P_0)$ on the right. This is an equilibrium state, and therefore the exact solution remains in this state for all time. For the HLL solver, we get a mass flux of $F_L = F_R = 0$ and wavespeeds $\lambda_\pm = \pm c_s$. The HLL derived density in the star region is then the mean, $\rho_* = \frac{1}{2}(\rho_L + \rho_R)$. The HLL mass flux, $F_* = \frac{1}{2}c_s(\rho_L - \rho_R)$. The fact that the HLL flux is nonzero reflects that the solver attempts to smooth the contact wave towards the mean in the star region, rather than working to preserve the discontinuity. If we wish to preserve the contact discontinuity, we must use fewer approximations, i.e. we must use a Riemann solver which knows about the contact wave.

1.1.4 The HLLC Riemann Solver

The HLL solver required very little information about the underlying system of equations; we were able to obtain a flux using only the maximum and minimum wavespeeds. If we wish to capture the contact wave, we must input more information about the field equations being solved; at the very least, we must work out the velocity of the contact wave.

While the HLL solver divides Figure 1.1 into three regions, the HLLC solver (C for "contact") divides it into four regions, giving a piecewise constant solution,

$$u(x,t) = \begin{cases} u_L & x/t < \lambda_- \\ u_{*L} & \lambda_- < x/t < \lambda_* \\ u_{*R} & \lambda_* < x/t < \lambda_+ \\ u_R & x/t > \lambda_+ \end{cases} \tag{1.10}$$

and a similar form for $F(x,t)$, where λ_* is the velocity of the contact wave. Note that if we can calculate all of the u_*'s, then the fluxes can be easily derived from the jump conditions (1.7). For example, if we wish to calculate F_{*R} from u_{*R}, we can simply rewrite (1.7) as

$$F_{*R} = F_R - \lambda_+(u_R - u_{*R}). \tag{1.11}$$

Therefore, we can reduce the problem to determining the conserved variables u_{*R}. (In the following derivations, we assume without loss of generality that we wish to determine the right-starred states, i.e. that we are interested in the region $\lambda_* < x/t < \lambda_+$. The formulas for left-starred states are completely analogous.) First, it will be convenient to define the variables $\{q\}$. For a given conserved quantity, q_α tells us the amount of flux crossing the characteristic $x/t = \lambda_+$:

$$q \equiv \lambda_+ u_R - F_R = \lambda_+ u_* - F_*. \tag{1.12}$$

The second equality is simply using continuity of q; flux in one side equals flux out the other. Note that the various q's can be calculated completely from known data, i.e. λ_+ and information in the right state. Explicitly,

$$\begin{array}{rcll} q_1 & = & \rho_R(\lambda_+ - v_R) & : \quad \text{Mass Flux} \\ q_2 & = & \rho_R v_R(\lambda_+ - v_R) - P_R & : \quad \text{Momentum Flux} \\ q_3 & = & E_R(\lambda_+ - v_R) - P_R v_R & : \quad \text{Energy Flux} \end{array} \tag{1.13}$$

where we have defined $E_R = \frac{1}{2}\rho_R v_R^2 + \epsilon_R$ as the energy density in the right state. Let us assume for the moment that we have an expression for the contact wave velocity, λ_*. Then, using $v_* = \lambda_*$ and $q = \lambda_+ u_* - F_*$, we can solve for ρ_{*R} using q_1:

$$q_1 = \lambda_+ \rho_{*R} - \rho_{*R}\lambda_* \tag{1.14}$$

$$\rho_{*R} = \frac{q_1}{(\lambda_+ - \lambda_*)} = \rho_R \frac{(\lambda_+ - v_R)}{(\lambda_+ - \lambda_*)} \tag{1.15}$$

This can be easily understood from the conservation law. The enhancement in density can be alternatively calculated by fixing the amount of mass between the shock and contact discontinuity. This mass is equal to $\rho_{*R}(\lambda_+ - \lambda_*)t$. This should be the same as the amount of mass which has crossed the shock since $t = 0$, which can be calculated as $\rho_R(\lambda_+ - \lambda_R)t$. Equating these masses gives us the above formula. As we calculate the rest of the conserved variables, we can apply a similar interpretation to each formula we derive.

Calculating the momentum is straightforward, as we already know ρ_{*R} and λ_*, so momentum density is just $\rho_{*R}\lambda_*$. Finally, energy density can be calculated by first deriving P_{*R} from setting q_2 equal to $\lambda_* u_* - F_*$ in the momentum equation:

$$\begin{aligned} P_{*R} &= \rho_{*R}\lambda_*(\lambda_+ - \lambda_*) - q_2 \\ &= P_R + \rho_R(\lambda_+ - v_R)(\lambda_* - v_R) \end{aligned} \tag{1.16}$$

Now using q_3 and the energy equation:

$$\begin{aligned} E_{*R} &= \tfrac{q_3 + P_{*R}\lambda_*}{\lambda_+ - \lambda_*} \\ &= E_R \tfrac{\lambda_+ - v_R}{\lambda_+ - \lambda_*} + (P_R + \rho_R(\lambda_+ - v_R)\lambda_*) \tfrac{\lambda_* - v_R}{\lambda_+ - \lambda_*} \end{aligned} \tag{1.17}$$

The first term simply represents the same compression of energy density as we found with the mass density. The rest must therefore represent the amount of energy which passes through the contact wave.

The Riemann solver is now completely specified, provided we choose a value for λ_*. In fact, the stability of our scheme depends sensitively on this choice.

One choice we can make is to enforce that the pressure be continuous across the contact discontinuity. This means $P_{*L} = P_{*R}$, or in other words:

$$P_L + \rho_L(\lambda_- - v_L)(\lambda_* - v_L) = P_R + \rho_R(\lambda_+ - v_R)(\lambda_* - v_R) \tag{1.18}$$

Solving for λ_*, also adding the assumption $\lambda_+ = v_R + c_R$ and $\lambda_- = v_L - c_L$, where c is the sound speed,

$$\lambda_* = \frac{P_L - P_R + \rho_L c_L v_L + \rho_R c_R v_R}{\rho_L c_L + \rho_R c_R} \tag{1.19}$$

This provides the correct limit when the pressure and velocity are equal on either side of the contact. It also gives a crucial nonzero contact wave when there is no initial velocity but there is a pressure difference. We have found this choice to be quite stable.

Now, the procedure for determining the fluxes is as follows: first, calculate the wavespeeds $\lambda_\pm$, and calculate λ_* from (1.19). Then, calculate the conserved variables u_{*R} (or u_{*L} if appropriate) using (1.15) and (1.17), including the momentum density, $\rho_{*R}\lambda_*$. Finally, calculate the fluxes F_{*R} from (1.11).

1.1.5 Piecewise Linear Method

Up to this point we have provided essentially enough information to make a fairly accurate first-order hydro code. If we wish to make the scheme higher-order, we must determine how to interpolate from cell-averaged values to the face-centered values $\{u_L\}$ and $\{u_R\}$, remembering that we cannot always assume the solution is continuous in space.

We shall first proceed pretending that this does not pose any issue. It is straightforward to numerically calculate gradients of all quantities in a given zone, reminding ourselves that the $\{u_i\}$ are zone-averaged quantities and therefore the gradients we calculate must apply to the entire zone, not just one side (We cannot use one slope to extrapolate to the left and another to extrapolate to the right, otherwise we violate our interpretation of u_i as the average value of u in the zone).

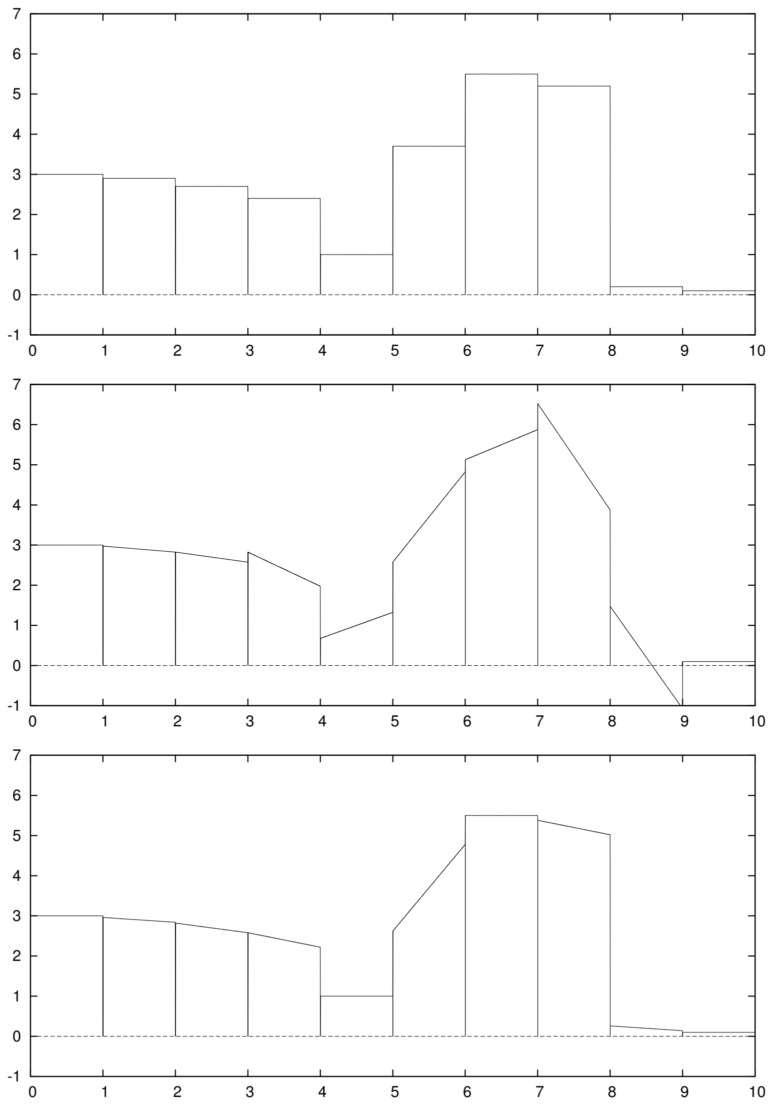

Figure 1.2 Slope limiting for the piecewise linear method. Top panel: A piecewise-constant function. Center panel: A piecewise-linear approximation to the function using centered differences. Lower panel: Slope-limited gradients using the minmod slope limiter. The slope-limited description of the function is second-order in the smooth regions, and reverts to first-order in discontinuous regions.

After calculating the gradients, we now interpret our variables as being represented by a piecewise linear function. Generically, this function might have some poorly-behaved features. In the center panel of Figure 1.2, we show an example of such a piecewise-linear function. At certain points, the gradients we estimated introduce artificial features, whenever the gradient extrapolation "overshoots", potentially introducing non-monotonic behavior in an otherwise monotonic region of the function. These artificial features tend to occur exactly at locations where the derivative is not well-defined anyway; in such regions (in particular, across shocks) a piecewise-constant function may be a better description of the flow.

The strategy we will employ is to use the numerically calculated gradients in the smooth part of the flow, but revert to a first-order method in the discontinuous part of the flow. This is achieved by the use of a slope-limiter. The 1D version of a slope limiter is as follows: first, imagine we have some field ϕ_i sampled at each zone. For any given zone, we may numerically calculate a slope of ϕ using either a left, right, or centered difference. We denote these three slopes m_L, m_R and m_C, respectively. Now, if we wish to prevent overshoots and to preserve monotonicity where it exists, we must first throw out all three of these slopes if they don't all have the same sign. That is, if a given zone houses a local maximum or minimum of ϕ, then we should simply revert to first-order, choosing a slope $m = 0$. On the other hand, if all three slopes have the same sign, then we can guarantee preservation of monotonicity if we choose the slope with the smallest magnitude. This completely defines the minmod function:

$$m = \text{minmod}(m_L, m_R, m_C) \tag{1.20}$$

$$\text{minmod}(x, y, z) = \begin{cases} \min(x, y, z) & x > 0, y > 0, z > 0 \\ \max(x, y, z) & x < 0, y < 0, z < 0 \\ 0 & \text{otherwise} \end{cases} \tag{1.21}$$

Using the minmod function guarantees the stability of the second-order scheme. Actually, we can be slightly more aggressive and still guarantee stability by using

$$m = \text{minmod}(\theta m_L, \theta m_R, m_c), \tag{1.22}$$

where $1 < \theta < 2$.

1.1.6 Time Evolution

It can be shown that these methods are only stable if the Courant-Friedrichs-Lewy condition [15] is satisfied:

$$\Delta t < \min\{\frac{\Delta x_i}{|\lambda_i|}\}, \tag{1.23}$$

where Δx_i is the characteristic size of a zone, and λ_i is the fastest wavespeed in that zone. This formula makes intuitive sense, because if our timestep is larger than this, a physical signal can cross multiple zones in a single step, whereas our method only uses information from nearest neighbors to evolve forward in time. If we want to evolve on larger timesteps, we need more information than just the nearest neighbors; this is essentially the idea behind implicit methods, which will not be discussed here.

In practice, our timesteps are calculated via:

$$\Delta t = C_{CFL} \;\; \min\{\frac{\Delta x_i}{|\lambda_i|}\}, \tag{1.24}$$

where C_{CFL} is some positive constant smaller than unity.

The time-update step as described in (1.5) can be summarized as:

$$M_i^{n+1} = M_i^n + \Delta t \; L(\{\text{state} \;\; n\}) \tag{1.25}$$

where L is some time-derivative operator (not necessarily linear), which is dependent on the state at timestep n, as abstractly indicated in the formula. Here, the "state" at timestep n means essentially everything about the system; the values of $\{M_i^n\}$, the values of any external forces, and in principle the geometry of the computational mesh (which for now we choose to be static). At the moment, this scheme is first-order in time (essentially it is a forward-Euler timestep).

Writing the method in this way it is clearly straightforward to upgrade this to a second-order method, simply by using the method of Runge-Kutta. However, it is important that when we modify the method in this way, we maintain stability. It is also convenient if we use a method that does not require us to store too much information at any one time, as this storage eventually would become problematic. We therefore use the total-variation-diminishing low-storage Runge-Kutta methods of Shu & Osher [120]. For a second-order timestep, we evolve using the following steps:

$$\begin{array}{rccl} M_i^{(1)} = & M_i^n & + \Delta t\ L(\{\text{state}\ \ n\}) \\ M_i^{n+1} = & \frac{1}{2}(M_i^n + M_i^{(1)}) & + \frac{1}{2}\Delta t\ L(\{\text{state}\ \ (1)\}) \end{array} \tag{1.26}$$

For third-order in time:

$$\begin{array}{rccl} M_i^{(1)} = & M_i^n & + \Delta t\ L(\{\text{state}\ \ n\}) \\ M_i^{(2)} = & \frac{3}{4}M_i^n + \frac{1}{4}M_i^{(1)} & + \frac{3}{4}\Delta t\ L(\{\text{state}\ \ (1)\}) \\ M_i^{n+1} = & \frac{1}{3}M_i^n + \frac{2}{3}M_i^{(2)} & + \frac{1}{3}\Delta t\ L(\{\text{state}\ \ (2)\}) \end{array} \tag{1.27}$$

Note that each substep only requires the M_i from the beginning of the timestep, and the state at the beginning of the substep. In principle, any other quantity we wish to evolve in time can be evolved forward in exactly the same way; these formulas are abstract enough to be generally applicable. In practice, we generally use the second-order version of the formula in time, as our method is only second-order in space, and therefore we do not gain much in accuracy if the timestep is third-order.

1.2 Moving-Mesh Methods

Upgrading a fixed-mesh code to a moving-mesh code is (amazingly) straightforward; the only major complications arise from decisions over how to store the mesh data (i.e. how to reference computational zones and faces). A static mesh of arbitrary geometry (Voronoi or otherwise) is completely accounted for in the formulation up to this point. If we wish to move the mesh, changing the volumes of the zones, the areas of the faces, and the mesh topology (i.e. which zones are neighbors), this motion can be completely accounted for by making two small changes to the above formulation.

First, we assume that the mesh motion tells us the velocity of each face. We denote the face velocity by $\vec{w}_{ij}$; the velocity of the face connecting zones i and j. Then, the first change we must make is to the evolution equation itself. The spacetime volume being integrated over has faces whose normal is neither temporal nor spatial, and as a result, the integration gives an extra term in the evolution equations. After integrating over this spacetime volume, Equation (1.5) becomes

$$M_i^{n+1} = M_i^n + \Delta t \left(-\sum_j (\vec{F}_* - \vec{w}_{ij} u_*) \cdot \hat{n} \Delta A_{ij} + \bar{S}_i^{n+1/2} \Delta V \right) \tag{1.28}$$

that is, we must make the replacement $F_* \to F_* - (\vec{w}_{ij} \cdot \hat{n}) u_*$. This replacement has obvious physical meaning; we are subtracting off the amount of flux overtaken by the face as it moves through space. This adjustment is also straightforward, of course, because the Riemann solver automatically gives us a u_* whenever it gives us an F_*. Finally, this replacement is natural as it is consistent with our conservation-law framework. If we have done our jobs well, the advective fluxes will cancel very precisely, so that for example the amount of mass in a zone remains nearly constant.

Now, we turn to the second and final adjustment we need to make to the code. In the section in which we described Riemann solvers, we indicated that we always searched for the solution along the curve $x = 0$. This is the correct curve if the face is stationary. However, if the face is moving, the curve $x = 0$ will give an inappropriate flux through the face. We must adjust the Riemann solver in the following extremely simple manner; we find the solution along the characteristic $x/t = \vec{w} \cdot \hat{n}$, as this is the characteristic followed by the face. Again, ideally the face will move with a velocity close to the contact wave, so the appropriate Riemann solution will almost always be in the green "starred" region, even if the flow is moving supersonically and all characteristics are moving in one direction.

This adjustment to the Riemann solver is also a completely natural choice, as it is again consistent with the framework laid out by Harten, Lax, and Van Leer. We could have made other choices to account for the mesh motion which would not have been so consistent. For example, the moving-mesh code AREPO [121] performs an explicit boost into the frame of the face, solves the Riemann problem, boosts back to the lab frame, then evaluates the flux from the Riemann solution. This choice would normally be problematic, as it implicitly requires making the evaluation $F_* = F(u_*)$, which is not generally consistent with the conservation law and the HLL framework, and there is therefore no guarantee of stability. This does not usually pose a problem for AREPO, since this code uses an exact Riemann solver, and therefore it will happen to be true that $F_* = u_*$. In the case of an exact Riemann solver, such choices do not matter, as there is only one exact solution. It is only when an approximate Riemann solver is used, for which $F_* \neq F(u_*)$, that one should expect this choice to matter.

1.3 Organization

This thesis describes specific implementations of the above numerical scheme, and specific problems in astrophysical gas dynamics which the code has been used to solve. The content is laid out as follows. In Chapter 2 we describe the basic moving-mesh method and provide a great number of code tests to demonstrate its utility. The following two chapters focus on applications to gaseous disks; in chapter 3 I calculate the weakly nonlinear interaction between a protoplanet and the gaseous disk it inhabits during its formative years. Specifically, I focus on whether a gap can open in the vicinity of the protoplanet. In chapter 4 I calculate scaling relations for depth and width of the gap, and empirically find a new gap-opening criterion. The final two chapters focus on relativistic jet dynamics with applications to gamma ray bursts. Chapter 5 focuses on fluid instabilities experienced by decelerating relativistic matter, and finally in chapter 6 I describe a new relativistic jet model which can be used as initial conditions for future numerical studies.