Chapter 7

Riemann solvers II

7.1 Roe's linearized Riemann solver

7.1.1 The equations of hydrodynamics revisited

Before we construct our linearized Riemann solver, let us make a slightly modified definition of the state vector q, which allows us an easy generalization of our algorithms to 3-D. We define q to be

$$q = \begin{pmatrix} \rho e_{\rm tot} \\ \rho u \\ \rho v \\ \rho w \\ \rho \end{pmatrix} \tag{7.1}$$

For convenience we shall index it from 0 to 4:

$$q_0 = \rho e_{\rm tot} \quad q_1 = \rho u \quad q_2 = \rho v \quad q_3 = \rho w \quad q_4 = \rho \tag{7.2}$$

The full set of equations for 3-D hydrodynamics is then:

$$\partial_t q + \partial_x f_x(q) + \partial_y f_y(q) + \partial_z f_z(q) = 0 \tag{7.3}$$

where

$$f_x = \begin{pmatrix} \rho h_{\rm tot} u \\ \rho u^2 + P \\ \rho v u \\ \rho w u \\ \rho u \end{pmatrix} \qquad f_y = \begin{pmatrix} \rho h_{\rm tot} v \\ \rho u v \\ \rho v^2 + P \\ \rho w v \\ \rho v \end{pmatrix} \qquad f_z = \begin{pmatrix} \rho h_{\rm tot} w \\ \rho u w \\ \rho v w \\ \rho w^2 + P \\ \rho w \end{pmatrix} \tag{7.4}$$

7.1.2 Linearized Riemann solvers

We have seen that for linear problems the Riemann solver reduces to a characteristic solver. For the full non-linear set of equations of hydrodynamics this is no longer the case. A Riemann solver, such as Godunov's method, is then a rather complex solver because it involves complex and non-linear solutions to the Riemann problem at each cell interface. It is also rather costly to solve numerically. Cheaper and elegant simplifications are *linearized Riemann solvers*. The way this is done is by expressing our interface fluxes as much as possible only using the differences in the state variables:

$$\Delta q_{k,i-1/2} \equiv q_{k,i-1/2,R} - q_{k,i-1/2,L} \tag{7.5}$$

If these differences are small, then much of the algebra can be linearized to first order in $\Delta q_{k,i-1/2}$.

If we now set the state at the beginning of each time step constant within each cell, then the cell interfaces have jumps of the state, i.e. they define a Riemann problem. The way to linearize this is to define an average state at the interface $\hat{q}_{k,i-1/2}$ (note: here we retain the index k of the index notation) in some way:

$$\hat{q}_{k,i-1/2} = \text{Average}[q_{k,i}, q_{k,i-1}] \tag{7.6}$$

where the precise definition of the average will be defined later. For now we can simply set $\hat{q}_{k,i-1/2} = (q_{k,i} + q_{k,i-1})/2$ for instance. Now we can express the Riemann problem in the deviation from this average:

$$\delta q_{k,i-1/2,L} \equiv q_{k,i-1} - \hat{q}_{k,i-1/2} \tag{7.7}$$
$$\delta q_{k,i-1/2,R} \equiv q_{k,i} - \hat{q}_{k,i-1/2} \tag{7.8}$$

If $|\delta q_{k,i-1/2,L/R}| \ll |\hat{q}_{k,i-1/2}|$ then, locally, the Riemann problem can be regarded as a linear Riemann problem, which we have extensively discussed in Section 6.5. The advection matrix in x direction is now simply the Jacobian $\partial f_x(q)/\partial q$, so the equation, locally between $x_{i-1} < x < x_i$ becomes:

$$\partial_t \delta q + \left(\frac{\partial f_x}{\partial q}\right) \partial_x \delta q = 0 \tag{7.9}$$

The eigenvalues of the Jacobian $\partial f_x/\partial q$ at the interface $i - 1/2$ are (for convenience we leave out the $i - 1/2$ index):

$$\lambda_1 = \hat{u} - \hat{C}_s \tag{7.10}$$
$$\lambda_2 = \hat{u} + \hat{C}_s \tag{7.11}$$
$$\lambda_3 = \hat{u} \tag{7.12}$$
$$\lambda_4 = \hat{u} \tag{7.13}$$
$$\lambda_5 = \hat{u} \tag{7.14}$$

with eigenvectors:

$$e_1 = \begin{pmatrix} \hat{h}_{\text{tot}} - \hat{C}_s\hat{u} \\ \hat{u} - \hat{C}_s \\ \hat{v} \\ \hat{w} \\ 1 \end{pmatrix} e_2 = \begin{pmatrix} \hat{h}_{\text{tot}} + \hat{C}_s\hat{u} \\ \hat{u} + \hat{C}_s \\ \hat{v} \\ \hat{w} \\ 1 \end{pmatrix} \tag{7.15}$$

$$e_3 = \begin{pmatrix} \frac{1}{2}\hat{u}^2 \\ \hat{u} \\ \hat{v} \\ \hat{w} \\ 1 \end{pmatrix} e_4 = \begin{pmatrix} \hat{v}^2 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} e_5 = \begin{pmatrix} \hat{w}^2 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{7.16}$$

where $\hat{h}_{\text{tot}} = \hat{e}_{\text{tot}} + \hat{P}/\hat{\rho}$ is the total specific enthalpy and $\hat{C}_s = \sqrt{\gamma\hat{P}/\hat{\rho}}$ is the adiabatic sound speed. In all symbols the caret $\hat{}$ indicates that these are the primitive variables as derived from the "average state" $\hat{q}_{i-1/2}$ at the location of interface $i - 1/2$. Note that we can derive similar expressions for the advection in y and z direction.

We can now directly insert those formulae to Eq. (6.56) and apply this to the values of $\delta q_{k,i-1/2,L/R}$. Now the special form of Eq.(6.56) comes to our advantage, because since this expression (and the expressions for the flux limiters) only depend on the difference $\delta q_{k,i-1/2,R} - \delta q_{k,i-1/2,L} \equiv q_{k,i-1/2,R} - q_{k,i-1/2,L} \equiv \Delta q_{k,i-1/2}$, we can now forget about $\delta q_{k,i-1/2,L/R}$ and focus entirely on $\Delta q_{k,i-1/2}$, which is the jump of the state over the interface. We can now decompose $\Delta q_{k,i-1/2}$ into the eigenvectors Eq. (7.15...7.16):

$$\Delta q_{k,i-1/2} = \sum_{m=1\cdots5} \tilde{\Delta} q_{m,i-1/2} e_{m,k,i-1/2} \tag{7.17}$$

where (again for clarity we omit the index $i-1/2$):

$$\tilde{\Delta} q_1 = \frac{\gamma-1}{2\hat{C}_s^2}\left\{\hat{e}_{\rm kin}\Delta q_4 - \xi\right\} - \frac{\Delta q_1 - \hat{u}\Delta q_4}{2\hat{C}_s} \tag{7.18}$$

$$\tilde{\Delta} q_2 = \frac{\gamma-1}{2\hat{C}_s^2}\left\{\hat{e}_{\rm kin}\Delta q_4 - \xi\right\} + \frac{\Delta q_1 - \hat{u}\Delta q_4}{2\hat{C}_s} \tag{7.19}$$

$$\tilde{\Delta} q_3 = \frac{\gamma-1}{2\hat{C}_s^2}\left\{(\hat{h}_{\rm tot} - 2\hat{e}_{\rm kin})\Delta q_4 + \xi\right\} \tag{7.20}$$

$$\tilde{\Delta} q_4 = \Delta q_2 - v\Delta q_4 \tag{7.21}$$

$$\tilde{\Delta} q_5 = \Delta q_3 - w\Delta q_4 \tag{7.22}$$

where $\hat{e}_{\rm kin} = (\hat{u}^2 + \hat{v}^2 + \hat{w}^2)/2$ and $\xi \equiv u\Delta q_1 + v\Delta q_2 + w\Delta q_3 - \Delta q_0$. With these expressions for $\tilde{\Delta} q_{i-1/2}$ the flux at the interface becomes (cf. Eq. 6.56) becomes[1]

$$\begin{aligned} f^{n+1/2}_{k,i-1/2} =& \tfrac{1}{2}(f^n_{k,i-1/2,R} + f^n_{k,i-1/2,L}) \\ & - \tfrac{1}{2}\sum_{m=1\cdots5} \lambda_{m,i-1/2}\tilde{\Delta} q_{m,i-1/2} e_{m,k,i-1/2}[\theta_{m,i-1/2} + \tilde{\phi}_{m,i-1/2}(\epsilon_{m,i-1/2} - \theta_{m,i-1/2})] \end{aligned} \tag{7.23}$$

where we retained the index k in the expression for the flux: $f^{n+1/2}_{k,i-1/2}$ according to index notation. We now see that the interface flux for this linearized Riemann solver consists of the average of the non-linear fluxes plus a correction term in which the difference of the flux over the interface is decomposed into eigenvectors and each component advected in its own upwind fashion.

A linearized Riemann solver is evidently not an exact Riemann solver, since the Riemann problem is solved in an approximate way only. This is why linearized Riemann solvers are part of the (larger) family of *approximante Riemann solvers*.

7.1.3 Roe's average interface state

The final missing piece of the algorithm is a suitable expression for the "average interface state", or better, the "average interface primitive variables" $\hat{u}_{i-1/2}$, $\hat{v}_{i-1/2}$, $\hat{w}_{i-1/2}$, $\hat{\rho}_{i-1/2}$, $\hat{h}_{{\rm tot},i-1/2}$. As long as the numerical solution is very smooth, i.e. that $|\Delta q_{k,i-1/2}| \ll |\hat{q}_{k,i-1/2}|$, then any reasonable average would do and would probably give the right results. However, when contact discontinuities and/or shock waves are present in the solution, then it becomes extremely important to define the proper average such that the "linearization" (which is then strictly speaking no longer valid) still produces the right propagation of these discontinuities.

[1] Warning: There was a typo in this formula until Philipp Girichidis found it, 10 January 2011.

A *Roe solver* is a linearized Riemann solver with a special kind of averaged state at the interface. These state variables are defined as:

$$\hat{u} = \frac{\sqrt{\rho_L}u_L + \sqrt{\rho_R}u_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{7.24}$$

$$\hat{v} = \frac{\sqrt{\rho_L}v_L + \sqrt{\rho_R}v_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{7.25}$$

$$\hat{w} = \frac{\sqrt{\rho_L}w_L + \sqrt{\rho_R}w_R}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{7.26}$$

$$\hat{h}_{\text{tot}} = \frac{\sqrt{\rho_L}h_{\text{tot},L} + \sqrt{\rho_R}h_{\text{tot},R}}{\sqrt{\rho_L} + \sqrt{\rho_R}} \tag{7.27}$$

With these expressions for the average interface state and the above defined eigenvector decomposition and interface flux expressions, as well as the flux limiter, the Roe solver is complete.

7.1.4 Background of Roe's average interface state

So where do these expressions for the interface state come from? Here we follow the book by LeVeque. The basic idea behind the Roe solver is that it should recognize when a jump of the state $q_R - q_L$ is a pure jump in one characteristic family only, and in that case produce the exact propagation velocity. So for a shock or for a contact discontinuity belonging to just one characteristic family the interface-average propagation matrix $\hat{A}$ (which is the interface average of the Jacobian $\partial f_x/\partial q$) should have the property:

$$\hat{A}(q_R - q_L) = f_{x,R} - f_{x,L} = s(q_R - q_L) \tag{7.28}$$

where s is the propagation speed of this wave. Note that this should not only hold for small $q_R - q_L$, but also when the jump is macroscopic. In case of a contact discontinuity it should get $s = u$ and for a shock wave it should obtain $s = v_s$ where v_s is the shock velocity.

There is an elegant mathematical derivation of how we obtain Roe's parameterization from the above condition, which is based on the definition of a state vector W defined as:

$$W = \sqrt{\rho}\begin{pmatrix} h_{\text{tot}} \\ u \\ v \\ w \\ 1 \end{pmatrix} \tag{7.29}$$

also counted from 0 to 4. The nice property of this parameter vector is that the state vector is perfectly quadratic in the components of W:

$$q = \begin{pmatrix} \frac{1}{\gamma}W_0W_4 + \frac{\gamma-1}{2\gamma}(W_1^2 + W_2^2 + W_3^2) \\ W_1W_4 \\ W_2W_4 \\ W_3W_4 \\ W_4^2 \end{pmatrix} \tag{7.30}$$

and so is the flux f_x:

$$f_x = \begin{pmatrix} W_0 W_1 \\ \frac{\gamma-1}{\gamma} W_0 W_4 + \frac{\gamma+1}{2\gamma} W_1^2 - \frac{\gamma-1}{2\gamma}(W_2^2 + W_3^2) \\ W_1 W_2 \\ W_1 W_3 \\ W_1 W_4 \end{pmatrix} \tag{7.31}$$

and similar for f_y and f_z. This purely quadratic relation is very useful for the following argumentation.

If two states q_R and q_L are connected by a single wave (shock or contact discontinuity), then they obey

$$f_x(q_R) - f_x(q_L) = s(q_R - q_L) \tag{7.32}$$

where s is the propagation speed of this wave. We wish that our matrix $\hat{A}$ (our "Jacobian") recognizes this. So it should have the property that

$$\hat{A}(q_R - q_L) = s(q_R - q_L) \tag{7.33}$$

or in other words:

$$\hat{A}(q_R - q_L) = f_x(q_R) - f_x(q_L) \tag{7.34}$$

One way to obtain such a matrix is to integrate the Jacobian matrix $A(q)$ over a suitable path in parameter space from q_L to q_R. We can do this for instance by defining a straight path:

$$q(\xi) = q_L + (q_R - q_L)\xi \tag{7.35}$$

so that we get

$$\begin{aligned} f_x(q_R) - f_x(q_L) &= \int_0^1 \frac{\partial f_x(q(\xi))}{\partial \xi} d\xi \\ &= \int_0^1 \frac{\partial f_x(q(\xi))}{\partial q} \frac{\partial q(\xi)}{d\xi} d\xi \\ &= \left[\int_0^1 \frac{\partial f_x(q)}{\partial q} d\xi \right] (q_R - q_L) \end{aligned} \tag{7.36}$$

where we used $\partial q(\xi)/\partial \xi = (q_R - q_L)$. This the gives us the matrix $\hat{A}$:

$$\hat{A} = \int_0^1 \frac{\partial f_x(q)}{\partial q} d\xi \tag{7.37}$$

The problem with this integral is that it is not guaranteed that this produces a matrix which has real eigenvalues. Moreover, it is hard to evaluate, and therefore computationally costly.

Now here the parameter vector W comes in. Let us do the same trick of integration, but this time not in q space but in W space:

$$W(\xi) = W_L + (W_R - W_L)\xi \tag{7.38}$$

so that we get

$$\begin{aligned} f_x(q_R) - f_x(q_L) &= \left[\int_0^1 \frac{\partial f_x(W)}{\partial W} d\xi \right] (W_R - W_L) \\ &\equiv \hat{C}(W_R - W_L) \end{aligned} \tag{7.39}$$

We can do the same for $q_R - q_L$:

$$\begin{aligned} q_R - q_L &= \left[\int_0^1 \frac{\partial q(W)}{\partial W} d\xi\right](W_R - W_L) \\ &\equiv \hat{B}(W_R - W_L) \end{aligned} \tag{7.40}$$

So we have now two matrices B and C. We can now construct the matrix $\hat{A}$:

$$\hat{A} = \hat{C}\hat{B}^{-1} \tag{7.41}$$

Now the nice thing of the parameter vector W is that both q and f_x are purely quadratic in W, and therefore the $\partial q/\partial W$ and $\partial f_x/\partial W$ are *linear* in W. This makes it much easier to evaluate the integral through W space! We shall not derive the final expressions. It suffices to say that if we would derive $\hat{A}$ in the above way we obtain a matrix with eigenvalues and eigenvectors as we have given above. Roe's choice of average interface state variables can now be understood as originating from the fact that this particular choice of parameter vector W makes q and f_x quadratic in W.

7.1.5 The complete recipe of a Roe solver

Although we have discussed the complete algorithm already, let us finish this section with a point-by-point recipe for a Roe solver:

1. Determine first the Δt using a CFL number of 0.5
2. Construct the state vector $q = (\rho e_{\rm tot}, \rho u, \rho v, \rho w, \rho)$ at each cell center. This q_i also automatically defines the states at each side of the interfaces: $q_{i-1/2,R} = q_i, q_{i-1/2,L} = q_{i-1}$.
3. Construct Roe's averages at the cell interfaces.
4. Using Roe's averages, create the eigenvectors and eigenvalues
5. Compute the flux jump over the interface and decompose this jump into the eigen-components to obtain the values of $\tilde{\Delta} q_{k,i-1/2}$.
6. Compute the flux limiter for each of the eigen-components.
7. Compute the symmetric flux average $(f_{i-1/2,L} + f_{i-1/2,R})/2$, and add the diffusive correction term using the flux limiter and the $\tilde{\Delta} q$. This creates the interface flux.
8. Now update the state vector using the interface fluxes and the Δt computed using the CFL condition.
9. End of time step; off to the next time step

7.2 Properties of the Roe solver

7.2.1 Strengths of the Roe solver

The Roe solver is a very powerful solver:

- It resolves shocks and contact discontinuities very tightly (in roughly 3 grid points). It is therefore significantly higher resolution than classical hydrodynamics solvers.

- It does not require any artificial viscosity for shocks because it treats shocks directly. It is, so to speak, a *shock-capturing scheme*.
- It has a very low numerical viscosity/diffusivity.
- It is strictly conserving in mass, momentum and energy.
- Because it treats pressure gradients as characteristics, sound waves are propagated with the same precision as moving matter.

7.2.2 Problems of the Roe solver

The Roe solver has excellent performance for many problems, but sometimes it can produce problems. Here is a list of known problems of the solver:

- Under some (rare) conditions it can try to create *expansion shocks* where it should create smooth expansion waves. This is because the Roe solver is built in such a way as to recognize Rankine-Hugoniot jump conditions even if they are the reverse. A Roe solver decomposes any smooth wave into a series of small contact discontinuities or shocks. Even for expansion waves it can happen that one of these shocks becomes strong and produces an expansion shock. This is clearly unphysical and violates the entropy condition. If this happens an *entropy fix* is necessary: a trick that disallows an expansion shock to form. We refer to LeVeque's book for details.
- Just behind strong shocks in 2-D or 3-D flows, when the shock is parallel to the grid, sometimes waves can appear. This is an odd-even decoupling problem, and fixes have been proposed which involve a minute amount of diffusion parallel to the shock, but only in the neiborhood of the shock.
- Also for grid-parallel shocks one can sometimes observe severe protrusions appearing which have a pyramid shape and a checkerboard pattern. This is known as the *carbuncle phenomenon*. This problem is rare, but if it happens it is hard to solve.
- Due to the fact that the total energy equations is used in Roe solvers, in case of extremely supersonic flows it can happen that small errors in the momentum equation can yield negative thermal energies (because $e_{\rm th} = e_{\rm tot} - u^2/2$). This can be a potentially serious problem of all schemes that use the total energy equation. Various fixes can be considered, which can be useful in various regimes, but a generally valid solution is difficult.
- A rare, but possible problem of the Roe solver is that it does not guarantee that the interface flux that it produces is physical. For instance in the Riemann problem with $\gamma = 1.5$, $\rho_L = 1$, $u_L = -2$, $P_L = 4/3$ and $\rho_R = 4$, $u_R = 1$ and $P_R = 13/3$ the first order upwind flux has an energy flux but no mass flux (Eulerink & Mellema 1995). This is clearly unphysical, and can lead to numerical problems.

In general, though, the Roe solver is quite stable and very non-viscous. It is a truly high-resolution method.

7.3 The HLL family of solvers

We have seen that the Roe solver splits the Δq (where q is the state vector) into their projections onto the basis of eigenvectors of the Jacobian. Let us write each of these components of Δq as $\Delta_k q$ where k is the index of which eigenvector is meant. Each of the $\Delta_k q$ jumps is advected with its own speed: the characteristic velocity λ_k, which is the eigenvalue of the Jacobian matrix. This is precisely the scenario depicted in Fig. 6.1. Each of these moving jumps is in fact a *wave*, and λ_k is the wave speed. We know from the true solutions of the Riemann problem for the Euler equations (Fig. 6.3) that in reality not every wave is a jump: in particular the expansion wave is not a jump. In the Roe solver we just approximate each of them to be a jump, based on the semi-linearized set of equations.

Another Riemann solver, which also approximates all waves as jumps, is the HLL Riemann solver and its various derivatives. HLL stands for Harten, Lax and van Leer, who first proposed a method of this kind. The difference between the HLL family of solvers and the Roe solver is the wave propagation speeds λ_k and the wave decompositions of Δq into $\Delta_k q$ are not rigorously derived from the eigenvectors and eigenvalues of some approximation of the Jacobian matrix. Instead some simpler physical arguments are used to "guess" these values. Strictly speaking the HLL family of solvers allow an infinite number of variants because it is partly left to the developer of the HLL method which recipe to take to construct the wave propagation speeds λ_k and the decomposition of Δq into waves $\Delta_k q$.

Let us first look at HLL type solvers in a general sense, and later worry about how exactly to compute the λ_k and the waves $\Delta_k q$.

Suppose we have K waves (i.e. $k = 1 \cdots K$) ordered such that $\lambda_k \leq \lambda_{k+1}$. Once we have decomposed Δq into $\Delta_k q$ with wave speeds λ_k we can write the state $q(x,t)$ near the interface $i + 1/2$ as:

$$q(x, t + \Delta t) = q_i + \sum_{k: x > x_{i+1/2} + \lambda_k \Delta t} \Delta_k q_{i+1/2} \equiv q_{i+1} - \sum_{k: x \leq x_{i+1/2} + \lambda_k \Delta t} \Delta_k q_{i+1/2} \tag{7.42}$$

That is: the state $q(x,t)$ near the cell interface is split into $K+1$ regions. The first and the last have $q = q_i$ and $q = q_{i+1}$ respectively. And in the wave fan region we have $K - 1$ sub-regions of constant state q given by the above equation. For the flux that use for the cell updates we are only interested in $\bar{q}_{i+1/2} \equiv q(x = x_{i+1/2}, t + \Delta t)$, meaning we get:

$$\bar{q}_{i+1/2} \equiv q_i + \sum_{k: \lambda_k \Delta t < 0} \Delta_k q_{i+1/2} \equiv q_{i+1} - \sum_{k: \lambda_k \Delta t \geq 0} \Delta_k q_{i+1/2} \tag{7.43}$$

Strictly speaking, we can now compute a flux $\bar{f}_{i+1/2} = f(\bar{q}_{i+1/2})$ and we are done: we now have a way to express the interface flux of this approximation of the Riemann solution. However, it turns out that a better (more stable and reliable) way to compute the interface flux is to note that one can construct also wave jumps in the flux $\Delta_k f$ and use the property that:

$$\Delta_k f = \lambda_k \Delta_k q \tag{7.44}$$

This property is the equivalent of the Rankine-Hugoniot condition for shocks, but now applied to any wave. It basically guarantees that wave k indeed propagates with speed λ_k. In this way we can now construct the flux at the interface by adding up the jumps, like in the case of the construction of $\bar{q}_{i+1/2}$:

$$\bar{f}_{i+1/2} \equiv f_i + \sum_{k: \lambda_k \Delta t < 0} \lambda_k \Delta_k q_{i+1/2} \equiv f_{i+1} - \sum_{k: \lambda_k \Delta t \geq 0} \lambda_k \Delta_k q_{i+1/2} \tag{7.45}$$

Constructing the interface flux in this way is the basis of the HLL family of approximate Riemann solvers.

The above expression is still first order. To make the scheme second order one can use Eq. (6.56), which we repeat here using the Δ_k notation:

$$\begin{aligned} f_{i+1/2}^{n+1/2} =& \tfrac{1}{2}(f_{i+1/2,R}^n + f_{i+1/2,L}^n) \\ & - \tfrac{1}{2}\sum_{k=1\cdots K} [\theta_{k,i+1/2} + \tilde{\phi}_{k,i+1/2}(\epsilon_{k,i+1/2} - \theta_{k,i+1/2})]\Delta_k f_{i+1/2}^n \end{aligned} \tag{7.46}$$

where $\phi_{k,i+1/2}$ is the flux limiter of wave k. This is still precisely the same method of making the scheme higher order as we used for the Roe solver.

To complete our scheme we must ow choose λ_k and Δq_k in a clever way. This is the topic of the rest of this section.

7.3.1 The HLL solver

The simplest solver of this kind is the original one. It ignores the middle wave (the mass-advection wave) and decomposes Δq into *two* waves $\Delta_{(-)}q$ and $\Delta_{(+)}q$ which are the forward and backward moving sound waves. This gives us a left state $q_{L,i+1/2} = q_i$, a right state $q_{R,i+1/2} = q_{i+1}$ and a middle state $q_{M,i+1/2}$ which is assumed to be:

$$q_M = \frac{\lambda_{(+)}q_R - \lambda_{(-)}q_L + f_L - f_R}{\lambda_{(+)} - \lambda_{(-)}} \tag{7.47}$$

(where we dropped the $i + 1/2$ for notation convenience). Using the above expressions one can derive that the flux at the interface is then:

$$\bar{f}_{i+1/2} = \begin{cases} f_i & \text{if} \quad \lambda_{(-)} \geq 0 \\ \frac{\lambda_{(+)}f_i - \lambda_{(-)}f_{i+1} + \lambda_{(+)}\lambda_{(-)}(q_{i+1} - q_i)}{\lambda_{(+)} - \lambda_{(-)}} & \text{if} \quad \lambda_{(-)} < 0 < \lambda_{(+)} \\ f_{i+1} & \text{if} \quad \lambda_{(+)} \leq 0 \end{cases} \tag{7.48}$$

So what about the expressions for $\lambda_{(-)}$ and $\lambda_{(+)}$? There is a whole variety of proposed expressions for these values. The simplest is due to Davis (1988):

$$\lambda_{(-),i+1/2} = u_i - a_i \qquad \lambda_{(+),i+1/2} = u_{i+1} + a_{i+1} \tag{7.49}$$

where a is the sound speed. But there are many other versions too. See the book by Toro for an in-depth discussion.

One of the main disadvantages of this HLL solver is that it cannot keep contact discontinuities sharp. This is not surprising since we have no middle wave in this scheme.

7.3.2 The HLLC solver

A newer version of the HLL scheme is the HLLC scheme, where the C stands for central wave: this is a method which *does* include the middle wave that is missing from the standard HLL scheme. The general way to construct this scheme is the same as shown above. Instead of 2 waves and 3 regions of constant q we now have 3 waves and 4 regions of constant q. While the general method is the same, the details of the construction of the HLLC method (and its various estimates of the wave speeds) requires some more in-depth discussion. We refer to the book of Toro for these details.

7.4 Source extrapolation methods

One major disadvantage of Riemann solvers in general is that they are, by their structure, less capable of "recognizing" (seme-)static solutions in which pressure gradients are compensated by an external force (typically gravity). To demonstrate what is meant let us take the example of a hydrostatic atmosphere (e.g. Earth's atmosphere) with small perturbations on it (e.g. the formation of clouds). We want that the unperturbed atmosphere is recognized by the method in the sense that the method perfectly keeps the static atmosphere intact and does not produce wiggles, or worse: entropy. A hydrostatic atmosphere obeys:

$$\frac{dP}{dz} = -\rho g \tag{7.50}$$

where z is the vertical coordinate and g is the gravitational constant of the atmosphere ($g \simeq 1000\text{cm/s}^2$). Let us discretize this as:

$$\frac{P_{i+1} - P_i}{z_{i+1} - z_i} = -\tfrac{1}{2}(\rho_{i+1} + \rho_i)g \tag{7.51}$$

and let us construct the solution by choosing $P = K\rho^\gamma$ with K constant (an adiabatic atmosphere) and choosing $\rho(z = 0)$ as the density at the base. We can then integrate (for a choice of K) from bottom to some height above the surface using Eq.(7.51). Since this equation is implicit in ρ_{i+1} one must solve for each new ρ_{i+1} using for instance an iteration at each new grid point until Eq.(7.51) is satisfied. We then get a hydrostatic atmosphere that is consistent.

If we insert this into a time-dependent hydro code we want that this hydrocode leaves this solution exactly intact (i.e. that it does not introduce perturbations). For classical numerical hydro schemes of Chapter 5 this is in fact not difficult, especially not if a staggered grid is used. This is because the $\partial P/\partial z$ term is treated as a source term in such methods. We then have two source terms: the $\partial P/\partial z$ and the $-\rho g$ term. If we discretize these two terms in exactly the same way as in Eq.7.51, then both terms cancel out exactly (for this hydrostatic solution) and the hydrostatic solution is kept exactly intact (to machine precision). We can then safely study tiny perturbatations of this atmosphere without the worry that we may in fact be involuntarily studying the intrinsic noise of the method instead.

For Riemann solvers, however, this is not so easy. It is a fundamental property of these methods to include the $\partial P/\partial z$ term in the advection part, while the gravity force remains a source term (and can not be treated in the advection part). The pressure gradient force and the gravity force are therefore treated in fundamentally different ways and one cannot guarantee that they will *exactly* cancel for hydrostatic solutions.

To solve this problem Eulderink & Mellema (1995, A&ASup 110, 587) introduced the concept of *spatial flux extrapolation*. A very similar, but easier method was proposed by LeVeque (1998, J.Comp.Phys. 146, 346). We will use a combination of both formalisms here.

Consider the generalized hyperbolic equation with a source term:

$$\partial_t q(x,t) + \partial_x f(q(x,t)) = s(x) \tag{7.52}$$

The traditional way to do operator spliting is to first solve $\partial_t q(x,t) + \partial_x f(q(x,t)) = 0$ for one time step using for instance a Riemann solver and then solve $\partial_t q(x,t) = s(x)$ for the same time step (by simply adding the source). This gives us the problem of not recognizing steady states. A steady state is a solution of the equation

$$\partial_x f(q(x,t)) = s(x) \tag{7.53}$$

The *source extrapolation* method for time-dependent hydrodynamics is inspired on this stationary equation. At the start of each time step we construct a *subgrid model* $q(x, t = t_n)$ (for $x_{i-1/2} < x < x_{i+1/2}$) where $q(x = x_i, t = t_n) = q_i$ such that within the cell $q(x, t = t_n)$ is a solution of 7.53. To make things simpler we assume that $s(x)$ is constant within a cell, so we get:

$$\partial_x f(q(x,t)) = s_i \qquad \text{for} \quad x_{i-1/2} < x < x_{i+1/2} \tag{7.54}$$

From this equation we could in principle solve for $q(x, t)$ within cell i and thereby obtain values of q at the left and right sides of each interface: $q_{L,i+1/2}$ and $q_{R,i+1/2}$ (we come back in an minute to this). These are then the values we put into a Riemann solver to produce our interface flux $f_{i+1/2}$ which we use for the state update. It should be noted, however, that since now the states on both sides of the interface are no longer spatially constant (they follow the subgrid model). So in principle we have a *generalized Riemann problem* on the cell interface, in which the states are not constant. Such generalized Riemann problems are very difficult to solve. So instead, we simply ignore the fact that the states are strictly speaking not constant on both sides, and simply use $q_{L,i+1/2}$ and $q_{R,i+1/2}$ as the two states of a classical Riemann problem which we then solve with our favorite method.

Now let's come back to the solution of Eq. 7.54. If $f(q)$ is a linear function of q, say $f(q) = uq$ where u is a velocity, then the solution of Eq. 7.54 is simple:

$$q_{R,i-1/2} = q_i - \tfrac{1}{2}\Delta x_i s_i/u_i \tag{7.55}$$
$$q_{L,i+1/2} = q_i + \tfrac{1}{2}\Delta x_i s_i/u_i \tag{7.56}$$

This works fine as long as $u \neq 0$. Once u approaches zero we are in trouble. Moreover, if $f(q)$ is a *non-linear* function of q then we can write:

$$f_{R,i-1/2} = f(q_i) - \tfrac{1}{2}\Delta x_i s_i \tag{7.57}$$
$$f_{L,i+1/2} = f(q_i) + \tfrac{1}{2}\Delta x_i s_i \tag{7.58}$$

which is fine, but solving $q_{R,i+1/2}$ from a given flux $f_{R,i+1/2}$ is non-trivial, and in the case of the Euler equations typically has two different solutions, or one single solution or no solutions. It is therefore often unpractical to try to determine the $q_{L/R,i+1/2}$ from $f_{L/R,i+1/2}$. It turns out, however, that for Riemann solvers that are based entirely on propagating waves $\Delta_k f = \lambda_k \Delta_k q$ where Δ_k is the k-th wave and λ_k is its propagation speed, one can use $\Delta f_{i+1/2} = f_{R,i+1/2} - f_{L,i+1/2}$ as input to the wave-decomposition routine while keeping $q_{L,i+1/2} = q_i$ and $q_{R,i+1/2} = q_{i+1}$ as the primitive variables used to compute the eigenvectors and eigenvalues. In particular for the Roe solver this hybrid approach works well.

Now how does this solve the problem of recognizing a steady-state solution? The trick is that if the linear subgrid model described in Eq. (**??**) is sufficiently accurate, then one will obtain for the steady state solution:

$$f_{L,i+1/2} = f_{R,i+1/2} \tag{7.59}$$

meaning that

$$\Delta f_{i+1/2} = 0 \tag{7.60}$$

If we now use Eq.(7.46) then we get:

$$f^{n+1/2}_{i+1/2} = \tfrac{1}{2}(f^n_{i+1/2,R} + f^n_{i+1/2,L}) \tag{7.61}$$

because by virtue of $\Delta f_{i+1/2} = 0$ the entire Riemann solver part drops out of the equation now. What is left is:

$$q_i^{n+1} = q_i^n - \frac{\Delta t}{2\Delta x}[f(q_{i+1}^n) - f(q_{i-1}^n)] + \Delta t s_i^n \tag{7.62}$$

So if we construct our steady state initial condition such that

$$\frac{f(q_{i+1}) - f(q_{i-1})}{2\Delta x} + s_i = 0 \tag{7.63}$$

then the method finally recognizes the static solution down to machine precision.

7.5 Employing slope limiters *before* the Riemann solver step

So far we have studied solvers in which we could clearly distinguish two or three wave families. We were then able to apply flux limiters (or equivalently, slope limiters) to each wave (cf. Eq.7.46). This method ensures that the slope in each wave mode is taken into account and therefore gives second order accuracy in space.

Another method of making Riemann solvers higher-order is to apply slope limiters (not flux limiters) *before* the Riemann problem is solved. The trick is to extrapolate the cell-centered primitive variables $\rho, P, \vec{u}$ to the cell-walls using a linear subgrid model. We can also extrapolate the conserved quantities $\rho, \rho\vec{u}, \rho e_{\rm tot}$, whichever produces the best results. This then gives us the left and right states at the cell walls ($q_{L,i+1/2}$ and $q_{R,i+1/2}$), which defines a Riemann problem which we can then solve using any solver we want. Contrary to the flux limiter recipe of Eq.7.46 this slope limiter method can also be applied to Riemann solvers that do not approximate the problem into jump-like waves. This is therefore a way to make the original Godunov solver higher order.

This method is in a way similar to Section 7.4, but there is an essential difference: here we use neighboring points to compute the slope of the linear subgrid model (instead of the source function). We therefore get non-zero slopes also for cases without source. We can employ all the machinery of Chapter 4 to produce the best possible linear subgrid model using e.g. MINMOD or SUPERBEE slope limiters.

One major problem that we encounter if we simply apply this method as-is, is that it quickly becomes unstable. It turns out (see Toro's book) that this instability problem can be elegantly solved by producing adapted left- and right interface states $\tilde{q}_{L,i+1/2}$ and $\tilde{q}_{R,i+1/2}$ from the interface states $q_{L,i+1/2}$ and $q_{R,i+1/2}$ that come out of the slope limiter method by advancing these half a time step in time in the following way:

$$\tilde{q}_{L,i+1/2} = q_{L,i+1/2} + \frac{1}{2}\frac{\Delta t}{\Delta x}[f(q_{R,i-1/2}) - f(q_{L,i+1/2})] \tag{7.64}$$

$$\tilde{q}_{R,i+1/2} = q_{R,i+1/2} + \frac{1}{2}\frac{\Delta t}{\Delta x}[f(q_{R,i+1/2}) - f(q_{L,i+3/2})] \tag{7.65}$$

(see book by Toro, but beware of different notation). Note that the left one is updated using $q_{R,i-1/2}$ and $q_{L,i+1/2}$, while the right one is updated using $q_{R,i+1/2}$ and $q_{L,i+3/2}$. These equations are saying that the subgrid model *within each cell individually* is advanced half a time step. Using this method we obtain a stable higher-order scheme.

The method is now completed by inserting $\tilde{q}_{L,i+1/2}$ and $\tilde{q}_{R,i+1/2}$ into our favorite Riemann solver and obtaining the interface flux with which we update the cell centered state variables.

NOTE: The use of the slope limiters before the Riemann step excludes the use of the source extrapolation method described in Section 7.4, and vice versa. If one wishes to use the source extrapolation method then only the flux limiter method after the Riemann step (Eq. 7.46) can be used.

7.6 The PPM Method

A very well-known Riemann solver method is the "Piecewise Parabolic Method" (Colella & Woodward 1984, J.Comp.Phys. 54, 174). It was designed before many of the above described methods were developed, and therefore it lies a bit off from the main track as described above, but in large part it follows (and indeed pioneered) the same ideas. It has proved to be a pretty powerful method and is still very often used. We will describe it very briefly here, but we will not go into details because the method is rather complex and has many fine-tuning aspects.

The main idea of this Riemann solver is to use a reconstruction step *before* the Riemann step (see Section 7.5). But while we used linear reconstruction in Section 7.5, the PPM method uses quadratic reconstruction. In other words: starting from the cell-center values of the primitive variables ρ, p and $\vec{u}$ each cell subgrid model is a parabola. In the PPM method care is taken that no overshoots happen (TVD scheme) and that in general the reconstruction is well-behaved. Using these parabolic subgrid models, the primitive variables on both sides of each of the interfaces can be calculated. We know from Section 7.5 that if we directly insert these values into our Riemann problem solver, we get an unstable scheme. The PPM method solves this using an approximation. It finds the fastest moving left- and right- characteristics and makes an average of the primitive variables over these domains. This is the PPM version of the half-step-interface-update described in Section 7.5. It is less mathematically rigorous, but in practice it works.

Now these left- and right- interface values can be inserted in a Riemann solver code. Again, as in Section 7.4, the solution to the true (generalized) Riemann problem in this case is very complex and no analytic solutions exist of generalized Riemann problems with parabolic spatial dependency of the variables on both sides. Therefore, as before, the approximation is made to solve the classical Riemann problem, with constant states on both sides, even though we know that the states on both sides are not constant.

We refer to the original paper by Colella & Woodward 1984 for details of the method and how the method if fine-tuned.

7.7 Code testing: the Sod shock tube tests

The Sod shock tube test of Section 6.3.2 can be used to test the performance of our computer program. We leave it to the reader to text the algorithms of the previous chapter on this kind of test. Here we merely shock the performance of the Roe solver on a test with $\rho_L = 10^5$, $P_L = 1$, $u_L = 0$, $\rho_R = 1.25 \times 10^4$, $P_R = 0.1$, $u_R = 0$ and $\gamma = 7/5$. The result is shown in Fig. 7.1.

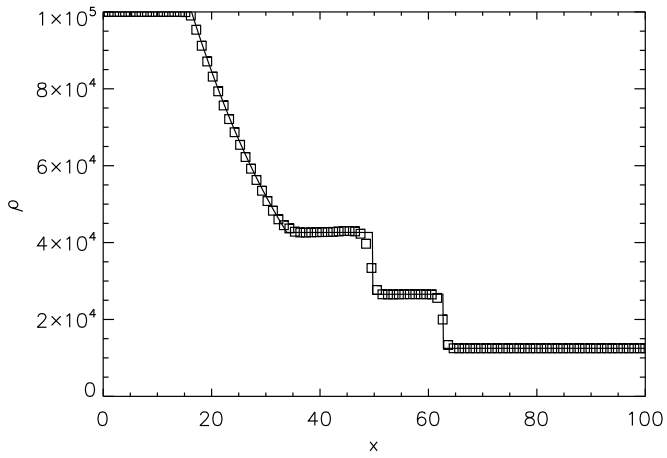

Figure 7.1. The result of the Roe solver with superbee flux limiter on a Sod shocktube test with $\rho_L = 10^5$, $P_L = 1$, $u_L = 0$, $\rho_R = 1.25 \times 10^4$, $P_R = 0.1$, $u_R = 0$ and $\gamma = 7/5$. Solid line: analytic solution; symbols: Roe solver.