21
Multidimensional Systems

In this chapter the high-resolution wave-propagation algorithms developed in Chapter 20 for scalar problems are extended to hyperbolic systems. We start with constant-coefficient linear systems, where the essential ingredients are most easily seen. A Riemann problem is first solved normal to each cell edge (a simple eigendecomposition in the linear case). The resulting waves are used to update cell averages on either side. The addition of correction terms using wave limiters (just as in one dimension) gives high-resolution terms modeling the pure x- and y-derivative terms in the Taylor series expansion (19.5). The cross-derivative terms are handled by simple extension of the corner-transport upwind (CTU) idea presented for the advection equation in Sections 20.2 through 20.5. In general this requires solving a second set of Riemann problems transverse to the interface. For a linear system this means performing a second eigendecomposition using the coefficient matrix in the transverse direction. Extending the methods to variable-coefficient or nonlinear systems is then easy, using ideas that are already familiar from one space dimension. The solutions (or approximate solutions) to the more general Riemann problems are used in place of the eigendecompositions, and the method is implemented in a wave-propagation form that applies very generally.

21.1 Constant-Coefficient Linear Systems

We again consider the constant-coefficient linear system $q_t + Aq_x + Bq_y = 0$ discussed in Chapter 19, where in particular the Lax–Wendroff and Godunov methods for this system were presented. The numerical fluxes for these two methods are given by (19.14) and (19.18) respectively. Our goal is to create a high-resolution version of the Lax–Wendroff method that incorporates upwinding and limiting. In addition to upwinding the first-order term (to rewrite it as the Godunov flux plus a high-resolution correction), we also wish to upwind the cross-derivative terms, as motivated by the CTU method for advection presented in Section 20.2. This can be accomplished by replacing the approximation (19.15) to ABq_y used in the Lax-Wendroff method with

$$\begin{aligned} ABq_y\big(x_{i-1/2}, y_j\big) \approx \frac{1}{\Delta y}[&A^-B^-(Q_{i,j+1} - Q_{ij}) + A^+B^-(Q_{i-1,j+1} - Q_{i-1,j}) \\ &+ A^-B^+(Q_{ij} - Q_{i,j-1}) + A^+B^+(Q_{i-1,j} - Q_{i-1,j-1})], \end{aligned} \tag{21.1}$$

generalizing the expression (20.28) for the CTU method on the advection equation. A similar approximation is used for the BAq_x term in the G-fluxes. Here $A^{\pm}$ and $B^{\pm}$ are defined as usual by (19.17). Note the important fact that

$$A^-B^- + A^+B^- + A^-B^+ + A^+B^+ = (A^- + B^+)(A^- + B^+) = AB \qquad (21.2)$$

for arbitrary matrices A and B, which ensures that we are still using a consistent approximation to the cross-derivative terms. Rather than multiplying each of the four jumps in Q by $\frac{1}{4}AB$ as in the Lax–Wendroff method, the product AB is split into four unequal pieces based on upwinding.

In practice we do not compute these matrices or the matrix products indicated above. Instead these terms in the flux are computed by solving Riemann problems and accumulating contributions to the fluxes based on the direction of wave propagation, exactly as was done for the advection equation in Section 20.2. This approach makes it easy to extend the method to variable-coefficient or nonlinear problems.

In spite of the fact that we do not actually compute the fluxes in terms of these matrices, it is useful to display them in this form for comparison with the Lax–Wendroff fluxes (19.14). We have

$$\begin{aligned} F_{i-1/2,j} = A^+Q_{i-1,j} + A^-Q_{ij} &+ \frac{1}{2}\sum_{p=1}^{m}|\lambda^{xp}|\left(1 - \frac{\Delta t}{\Delta x}|\lambda^{xp}|\right)\widetilde{\mathcal{W}}^p_{i-1/2,j} \\ &- \frac{\Delta t}{2\Delta y}[A^-B^-(Q_{i,j+1} - Q_{ij}) + A^+B^-(Q_{i-1,j+1} - Q_{i-1,j}) \\ &+ A^-B^+(Q_{ij} - Q_{i,j-1}) + A^+B^+(Q_{i-1,j} - Q_{i-1,j-1})], \end{aligned} \qquad (21.3)$$

and a similar expression for $G_{i,j-1/2}$. Here $\mathcal{W}^p_{i-1/2,j} = \alpha^p_{i-1/2,j}r^{xp}$ is the pth wave in the Riemann solution, with $\alpha^p_{i-1/2,j} = (R^x)^{-1}(Q_{ij} - Q_{i-1,j})$. The limited version $\widetilde{\mathcal{W}}^p_{i-1/2,j}$ is obtained by comparing this wave with $\widetilde{\mathcal{W}}^p_{I-1/2,j}$, where

$$I = \begin{cases} i-1 & \text{if } \lambda^{xp} > 0, \\ i+1 & \text{if } \lambda^{xp} < 0. \end{cases}$$

If no limiter is used, then, as in one dimension,

$$\sum_{p=1}^{m}|\lambda^{xp}|\left(1 - \frac{\Delta t}{\Delta x}|\lambda^{xp}|\right)\mathcal{W}^p_{i-1/2,j} = |A|\left(I - \frac{\Delta t}{\Delta x}|A|\right)(Q_{ij} - Q_{i-1,j})$$

and

$$\begin{aligned} &A^+Q_{i-1,j} + A^-Q_{ij} + \frac{1}{2}\sum_{p=1}^{m}|\lambda^{xp}|\left(1 - \frac{\Delta t}{\Delta x}|\lambda^{xp}|\right)\mathcal{W}^p_{i-1/2,j} \\ &\qquad = A^+Q_{i-1,j} + A^-Q_{ij} + \frac{1}{2}|A|\left(I - \frac{\Delta t}{\Delta x}|A|\right)(Q_{ij} - Q_{i-1,j}) \\ &\qquad = \frac{1}{2}A(Q_{i-1,j} + Q_{ij}) - \frac{1}{2}\frac{\Delta t}{\Delta x}A^2(Q_{ij} - Q_{i-1,j}), \end{aligned} \qquad (21.4)$$

which agrees with the corresponding terms in the Lax–Wendroff flux (19.14). On the other hand, if all waves are fully limited so that $\widetilde{\mathcal{W}}^p_{i-1/2,j} = 0$, then these terms in the flux reduce to the Godunov flux (19.18).

21.2 The Wave-Propagation Approach to Accumulating Fluxes

To implement the method described above, we take an approach very similar to what was done in Sections 20.5 and 20.6 for the advection equation. We use the form (19.19), which means we need fluctuations and correction fluxes. We present the algorithm for computing each of these in a framework that easily extends to nonlinear systems of equations by using approximate Riemann solvers:

1. Initialize $\tilde{F}_{i-1/2,j} = 0$ and $\tilde{G}_{i,j-1/2} = 0$ at each interface.
2. Sweep through the grid, solving each Riemann problem in x. At the interface between cells $\mathcal{C}_{i-1,j}$ and $\mathcal{C}_{ij}$ we use data $Q_{i-1,j}$ and Q_{ij} to compute waves $\mathcal{W}^p_{i-1/2,j}$ and speeds $s^p_{i-1/2,j}$. We also compute fluctuations $\mathcal{A}^-\Delta Q_{i-1/2,j}$ and $\mathcal{A}^+\Delta Q_{i-1/2,j}$ exactly as in one space dimension. For the constant-coefficient linear case the $\mathcal{W}$ and s will be eigenvectors and eigenvalues of A and we will have

$$\begin{aligned}\mathcal{A}^-\Delta Q_{i-1/2,j} &= \sum_{p=1}^m \left(s^p_{i-1/2,j}\right)^- \mathcal{W}^p_{i-1/2,j} = A^-\Delta Q_{i-1/2,j},\\ \mathcal{A}^+\Delta Q_{i-1/2,j} &= \sum_{p=1}^m \left(s^p_{i-1/2,j}\right)^+ \mathcal{W}^p_{i-1/2,j} = A^+\Delta Q_{i-1/2,j}.\end{aligned} \tag{21.5}$$

3. The waves are limited to obtain $\widetilde{\mathcal{W}}^p_{i-1/2,j}$ and these are used to update the correction fluxes at this interface:

$$\tilde{F}_{i-1/2,j} := \tilde{F}_{i-1/2,j} + \frac{1}{2}\sum_{p=1}^m \left|s^p_{i-1/2,j}\right| \left(1 - \frac{\Delta t}{\Delta x}\left|s^p_{i-1/2,j}\right|\right) \widetilde{\mathcal{W}}^p_{i-1/2,j}. \tag{21.6}$$

4. The right-going fluctuation $\mathcal{A}^+\Delta Q_{i-1/2,j}$ is used to compute an up-going transverse fluctuation $\mathcal{B}^+\mathcal{A}^+\Delta Q_{i-1/2,j}$ and a down-going transverse fluctuation $\mathcal{B}^-\mathcal{A}^+\Delta Q_{i-1/2,j}$ by solving a *transverse Riemann problem*. We have seen an exmple of this for the advection equation $q_t + uq_x + vq_y = 0$ in Section 20.5, where $\mathcal{A}^+\Delta Q_{i-1/2,j} = u^+_{i-1/2,j}(Q_{ij} - Q_{i-1,j})$ and the transverse fluctuations are defined by (20.25) and (20.26),

$$\mathcal{B}^\pm\mathcal{A}^+\Delta Q_{i-1/2,j} = v^\pm_{i,j\pm1/2} u^+_{i-1/2,j}(Q_{ij} - Q_{i-1,j}). \tag{21.7}$$

In general the symbols $\mathcal{B}^+\mathcal{A}^+\Delta Q$ and $\mathcal{B}^-\mathcal{A}^+\Delta Q$ each represent a single m-vector obtained by some decomposition of the fluctuation $\mathcal{A}^+\Delta Q$. The notation is motivated by the linear case, in which case we want

$$\mathcal{B}^\pm\mathcal{A}^+\Delta Q_{i-1/2,j} = B^\pm A^+(Q_{ij} - Q_{i-1,j}). \tag{21.8}$$

In the linear system case these are computed by decomposing the fluctuation $\mathcal{A}^+\Delta Q_{i-1/2,j}$ into eigenvectors of B,

$$\mathcal{A}^+\Delta Q_{i-1/2,j} = \sum_{p=1}^{m} \beta^p r^{yp},$$

and then setting

$$\mathcal{B}^\pm\mathcal{A}^+\Delta Q_{i-1/2,j} = \sum_{p=1}^{m} (\lambda^{yp})^\pm \beta^p r^{yp}. \tag{21.9}$$

This *wave decomposition* of $\mathcal{A}^+\Delta Q_{i-1/2,j}$ can be viewed as solving a second Riemann problem in the transverse direction, even though it is not based on left and right states as we normally interpret a Riemann solver. The net contribution of all right-going waves is split up into up-going and down-going parts based on the eigenvectors corresponding to plane waves in the y-direction.

5. These fluctuations $\mathcal{B}^\pm\mathcal{A}^+\Delta Q_{i-1/2,j}$ are used to update the correction fluxes above and below cell $\mathcal{C}_{ij}$:

$$\begin{aligned} \tilde{G}_{i,j+1/2} &:= \tilde{G}_{i,j+1/2} - \frac{\Delta t}{2\,\Delta x}\mathcal{B}^+\mathcal{A}^+\Delta Q_{i-1/2,j}, \\ \tilde{G}_{i,j-1/2} &:= \tilde{G}_{i,j-1/2} - \frac{\Delta t}{2\,\Delta x}\mathcal{B}^-\mathcal{A}^+\Delta Q_{i-1/2,j}. \end{aligned} \tag{21.10}$$

6. In a similar manner, the left-going fluctuation $\mathcal{A}^-\Delta Q_{i-1/2,j}$ is split into transverse fluctuations $\mathcal{B}^\pm\mathcal{A}^-\Delta Q_{i-1/2,j}$, which are then used to update the fluxes above and below cell $\mathcal{C}_{i-1,j}$:

$$\begin{aligned} \tilde{G}_{i-1,j+1/2} &:= \tilde{G}_{i-1,j+1/2} - \frac{\Delta t}{2\,\Delta x}\mathcal{B}^+\mathcal{A}^-\Delta Q_{i-1/2,j}, \\ \tilde{G}_{i-1,j-1/2} &:= \tilde{G}_{i-1,j-1/2} - \frac{\Delta t}{2\,\Delta x}\mathcal{B}^-\mathcal{A}^-\Delta Q_{i-1/2,j}. \end{aligned} \tag{21.11}$$

Note that these updates to nearby $\tilde{G}$ fluxes are exactly analogous to what was done in (20.24) for the scalar advection equation.

7. Steps 2–6 are now repeated for each Riemann problem in y, at interfaces between cells $\mathcal{C}_{i,j-1}$ and $\mathcal{C}_{ij}$. The resulting waves $\mathcal{W}_{i,j-1/2}$ are limited by comparisons in the y-direction and used to update $\tilde{G}_{i,j-1/2}$. In solving these Riemann problems we also compute fluctuations $\mathcal{B}^\pm\Delta Q_{i,j-1/2}$, which are then split transversely into $\mathcal{A}^\pm\mathcal{B}^+\Delta Q_{i,j-1/2}$ and $\mathcal{A}^\pm\mathcal{B}^-\Delta Q_{i,j-1/2}$. These four transverse fluctuations are used to modify four nearby $\tilde{F}$ fluxes, as was done in (20.25) for the advection equation.
8. Finally, the updating formula (19.19) is applied to advance by time Δt,

$$\begin{aligned} Q_{ij}^{n+1} = Q_{ij} &- \frac{\Delta t}{\Delta x}\left(\mathcal{A}^+\Delta Q_{i-1/2,j} + \mathcal{A}^-\Delta Q_{i+1/2,j}\right) \\ &- \frac{\Delta t}{\Delta y}\left(\mathcal{B}^+\Delta Q_{i,j-1/2} + \mathcal{B}^-\Delta Q_{i,j+1/2}\right) \\ &- \frac{\Delta t}{\Delta x}\left(\tilde{F}_{i+1/2,j} - \tilde{F}_{i-1/2,j}\right) - \frac{\Delta t}{\Delta y}\left(\tilde{G}_{i,j+1/2} - \tilde{G}_{i,j-1/2}\right). \end{aligned} \tag{21.12}$$

21.3 CLAWPACK Implementation

This wave-propagation algorithm is implemented in CLAWPACK by assuming that the user has provided two Riemann solvers. One, called `rpn2`, solves the Riemann problem normal to any cell interface and is similar to the one-dimensional Riemann solver `rp1`. For dimensional splitting only this one Riemann solver is needed. The new Riemann solver needed for the unsplit algorithm is called `rpt2` and solves Riemann problems of the sort just described in the transverse direction.

Each Riemann solver must be capable of solving the appropriate Riemann problem in either the x-direction or the y-direction, depending on which edge of the cell we are working on. The computations are organized by first taking sweeps in the x-direction along each row of the grid and then sweeps in the y-direction along each column. In each sweep a one-dimensional slice of the data is passed into the Riemann solver `rpn2`, so that `ql` and `qr` in this routine are exactly analogous to `ql` and `qr` in the one-dimensional Riemann solver `rp1`. A flag `ixy` is also passed in to indicate whether this is an x-slice (if `ixy=1`) or a y-slice (if `ixy=2`) of the data. The corresponding one-dimensional slice of the auxiliary array is also passed in.

The subroutine returns vectors of fluctuations (`amdq, apdq`) and waves and speeds (`wave, s`) obtained by solving the one-dimensional Riemann problem at each interface along the slice, as in `rp1`. In order to use the dimensional-splitting methods described in Section 19.5, only this Riemann solver `rpn2` is required.

To perform the transverse Riemann solves required in the multidimensional wave-propagation algorithms, each of the fluctuations `amdq` $(= \mathcal{A}^-\Delta Q)$ and `apdq` $(= \mathcal{A}^+\Delta Q)$ must be passed into the transverse solver `rpt2`, so this routine is called twice. Within the subroutine this parameter is called `asdq` $(= \mathcal{A}^*\Delta Q)$, and a parameter `imp` indicates which fluctuation this is (`imp=1` if `asdq` $= \mathcal{A}^-\Delta Q$, and `imp=2` if `asdq` $= \mathcal{A}^+\Delta Q$). For many problems the subroutine's action may be independent of the value of `imp`. In particular, for a constant-coefficient linear system the vector $\mathcal{A}^*\Delta Q$ is simply decomposed into eigenvectors of B in either case. For a variable-coefficient problem, however, the matrix B may be different to the left and right of the interface, and so the decomposition may depend on which direction the fluctuation is propagating.

The routine `rpt2` returns `bmasdq` $(= \mathcal{B}^-\mathcal{A}^*\Delta Q)$ and `bpasdq` $(= \mathcal{B}^+\mathcal{A}^*\Delta Q)$, the splitting of this fluctuation in the transverse direction. These terms are used to update the correction fluxes $\tilde{G}$ nearby.

The same routine is used during the y-sweeps to split $\mathcal{B}^\pm\Delta Q$ into $\mathcal{A}^\pm\mathcal{B}^\pm\Delta Q$, so when `ixy=2` it is important to realize that the input parameter `asdq` represents either $\mathcal{B}^-\Delta Q$ or $\mathcal{B}^+\Delta Q$, while the outputs `bmasdq` and `bpasdq` now represent $\mathcal{A}^\pm\mathcal{B}^*\Delta Q$. Similarly, in `rpn2` the parameter `asdq` represents $\mathcal{A}^*\Delta Q$ in the x-sweeps, as described above, and represents $\mathcal{B}^*\Delta q$ in the y-sweeps. It may be easiest to simply remember that in these routines "`a`" always refers to the normal direction and "`b`" to the transverse direction.

For many systems of equations the Riemann solver for the x-sweeps and y-sweeps take a very similar form, especially if the equations are isotropic and have exactly the same form in any direction (as is the case for many physical systems such as acoustics, shallow water, or gas dynamics). Then the cases `ixy=1` and `ixy=2` may be distinguished only by which component of q represents the normal velocity and which is the transverse velocity. This is the reason that a single Riemann solver `rpn2` with a flag `ixy` is required rather than separate Riemann solvers in the x- and y-directions.

The parameter values `method(2)` and `method(3)` determine what method is used in CLAWPACK. If `method(2)=1` then the first-order updates are used but the second-order corrections based on limited waves are not used, i.e., step 3 in the above algorithm is skipped. If `method(3)=0` then no transverse propagation is done, i.e., steps 4–6 are skipped. If `method(3)=1` then these steps are performed. If `method(3)=2` then an additional improvement is made to the algorithm, in which the correction terms from step 3 are also split in the transverse direction. This is accomplished by applying the transverse solver `rpt2` to the vectors

$$\mathcal{A}^-\Delta Q_{i-1/2,j} + \sum_{p=1}^{m} \left|s^p_{i-1/2,j}\right| \left(1 - \frac{\Delta t}{\Delta x}\left|s^p_{i-1/2,j}\right|\right) \widetilde{\mathcal{W}}^p_{i-1/2,j}$$

and

$$\mathcal{A}^+\Delta Q_{i-1/2,j} - \sum_{p=1}^{m} \left|s^p_{i-1/2,j}\right| \left(1 - \frac{\Delta t}{\Delta x}\left|s^p_{i-1/2,j}\right|\right) \widetilde{\mathcal{W}}^p_{i-1/2,j}$$

instead of to $\mathcal{A}^-\Delta Q_{i-1/2,j}$ and $\mathcal{A}^+\Delta Q_{i-1/2,j}$. The rationale for this is explained in [283].

If `method(3)<0` is specified, then dimensional splitting is used instead of this unsplit algorithm, as has already been described in Section 19.5.1.

See the CLAWPACK *User Guide* and sample programs for more description of the normal and transverse Riemann solvers. As an example we consider the acoustics equations in the next section. The CLAWPACK Riemann solver for this system may be found in `[claw/book/chap21/acoustics]`.

21.4 Acoustics

As an example, consider the two-dimensional acoustics equations (18.24) with no background flow ($u_0 = v_0 = 0$). In this case the eigenvectors of A and B are given in (18.31) and (18.32), and the eigenvalues of each are $\lambda^{x1} = -c_0$, $\lambda^{x2} = 0$, and $\lambda^{x3} = c_0$.

The Riemann solver `rpn2` must solve the Riemann problem $q_t + Aq_x = 0$ in the x-direction when `ixy=1` or $q_t + Bq_y = 0$ in the y-direction when `ixy=2`.

If `ixy=1`, then we decompose $\Delta Q_{i-1/2,j} = Q_{ij} - Q_{i-1,j}$ as

$$\Delta Q = \alpha^1 r^{x1} + \alpha^2 r^{x2} + \alpha^3 r^{x3}, \tag{21.13}$$

where the eigenvectors are given in (18.31). For clarity the subscript $i - 1/2$, j has been dropped from ΔQ and also from the coefficients α, which are different at each interface of course.

Solving the linear system (21.13) for α yields

$$\begin{aligned} \alpha^1 &= \frac{-\Delta Q^1 + Z_0\,\Delta Q^2}{2Z_0}, \\ \alpha^2 &= \Delta Q^3, \\ \alpha^3 &= \frac{\Delta Q^1 + Z_0\,\Delta Q^2}{2Z_0}. \end{aligned} \tag{21.14}$$

The waves are then given by

$$\mathcal{W}^1 = \alpha^1 \begin{bmatrix} -Z_0 \\ 1 \\ 0 \end{bmatrix}, \qquad \mathcal{W}^2 = \alpha^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathcal{W}^3 = \alpha^3 \begin{bmatrix} Z_0 \\ 1 \\ 0 \end{bmatrix}, \tag{21.15}$$

and the corresponding wave speeds are $s^1 = -c_0$, $s^2 = 0$, and $s^3 = c_0$.

The fluctuations are then given by

$$\mathcal{A}^- \Delta Q = s^1 \mathcal{W}^1, \qquad \mathcal{A}^+ \Delta Q = s^3 \mathcal{W}^3.$$

Note that the 2-wave makes no contribution to these fluctuations or to the second-order correction terms, where the contribution is also weighted by s^2, so the implementation can be made slightly more efficient by propagating only the 1-wave and 3-wave. (This is done in `[claw/book/chap21/acoustics]`, where `mwaves=2` is used.) If there were a nonzero background flow (u_0, v_0), then the wave speeds would be $s^1 = u_0 - c_0$, $s^2 = u_0$, and $s^3 = u_0 + c_0$. In this case it would be necessary to use all three waves and consider the sign of each s^p in computing the fluctuations.

If `ixy=2` then we are sweeping in the y-direction. We then need to decompose $\Delta Q = \Delta Q_{i,j-1/2} = Q_{ij} - Q_{i,j-1}$ as

$$\Delta Q = \alpha^1 r^{y1} + \alpha^2 r^{y2} + \alpha^3 r^{y3},$$

where the eigenvectors are given in (18.32). This yields

$$\begin{aligned} \alpha^1 &= \frac{-\Delta Q^1 + Z_0\,\Delta Q^3}{2Z_0}, \\ \alpha^2 &= \Delta Q^2, \\ \alpha^3 &= \frac{\Delta Q^1 + Z_0\,\Delta Q^3}{2Z_0}. \end{aligned} \tag{21.16}$$

The waves are

$$\mathcal{W}^1 = \alpha^1 \begin{bmatrix} -Z_0 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathcal{W}^2 = \alpha^2 \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \qquad \mathcal{W}^3 = \alpha^3 \begin{bmatrix} Z_0 \\ 0 \\ 1 \end{bmatrix}, \tag{21.17}$$

and the corresponding wave speeds are $s^1 = -c_0$, $s^2 = 0$, and $s^3 = c_0$. The fluctuations are then

$$\mathcal{B}^- \Delta Q = s^1 \mathcal{W}^1, \qquad \mathcal{B}^+ \Delta Q = s^3 \mathcal{W}^3,$$

but recall that in the CLAWPACK Riemann solver these are again denoted by `amdq` and `apdq`.

Note that these formulas are essentially the same for each value of `ixy`, except that the roles of the second and third components of Q are switched, depending on which velocity u or v is the velocity normal to the interface. In the CLAWPACK Riemann solver [claw/book/chap21/acoustics/rpn2ac.f], this is easily accomplished by using indices

$$\mathtt{mu} = \begin{cases} 2 & \text{if } \mathtt{ixy} = 1, \\ 3 & \text{if } \mathtt{ixy} = 2, \end{cases} \qquad \mathtt{mv} = \begin{cases} 3 & \text{if } \mathtt{ixy} = 1, \\ 2 & \text{if } \mathtt{ixy} = 2 \end{cases} \tag{21.18}$$

for the normal (mu) and transverse (mv) components of Q.

In fact, these formulas are easily generalized to solve a Riemann problem at any angle to the x and y axes. This is discussed in Section 23.6, where acoustics on a general quadrilateral grid is discussed.

The transverse Riemann solver rpt2 must take a fluctuation $\mathcal{A}^*\Delta Q$ and split it into $\mathcal{B}^-\mathcal{A}^*\Delta Q$ and $\mathcal{B}^+\mathcal{A}^*\Delta Q$, or take a fluctuation $\mathcal{B}^*\Delta q$ and split it into $\mathcal{A}^-\mathcal{B}^*\Delta Q$ and $\mathcal{A}^+\mathcal{B}^*\Delta Q$. This requires another splitting into eigenvectors of these matrices and multiplication by the corresponding eigenvalues. This is described in the next section for the more general problem of acoustics in heterogeneous media. For the constant-coefficient case, see also the simpler transverse Riemann solver [claw/book/chap21/acoustics/rpt2ac.f].

21.5 Acoustics in Heterogeneous Media

In Section 21.4 the normal and transverse Riemann solvers for acoustics in a homogeneous material were discussed. In this section we extend this to the case of a heterogeneous material, where the density $\rho(x, y)$ and the bulk modulus $K(x, y)$ may vary in space. The one-dimensional case has been studied in Section 9.6, and here we develop the two-dimensional generalization.

As in one dimension, the linear hyperbolic system can be solved in the nonconservative form

$$q_t + A(x, y)q_x + B(x, y)q_y = 0, \tag{21.19}$$

where

$$q = \begin{bmatrix} p \\ u \\ v \end{bmatrix}, \quad A = \begin{bmatrix} 0 & K(x, y) & 0 \\ 1/\rho(x, y) & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 0 & K(x, y) \\ 0 & 0 & 0 \\ 1/\rho(x, y) & 0 & 0 \end{bmatrix}. \tag{21.20}$$

This equation is not in conservation form, but can still be handled by high-resolution methods if we use the fluctuation form. The solution to the Riemann problem normal to each cell interface is computed exactly as in the one-dimensional case of Section 9.6. Let ρ_{ij} and c_{ij} be the density and sound speed in the (i, j) cell, where $c_{ij} = \sqrt{K_{ij}/\rho_{ij}}$. Then the Riemann problem at the $(i - 1/2, j)$ edge, for example, gives

$$\mathcal{W}^1 = \alpha^1 \begin{bmatrix} -Z_{i-1,j} \\ 1 \\ 0 \end{bmatrix}, \qquad \mathcal{W}^2 = \alpha^2 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \qquad \mathcal{W}^3 = \alpha^3 \begin{bmatrix} Z_{ij} \\ 1 \\ 0 \end{bmatrix},$$

where $Z_{ij} = \rho_{ij} c_{ij}$ is the impedance in the (i, j) cell, and

$$\begin{aligned}\alpha^1 &= \frac{-\Delta Q^1 + Z_{ij}\,\Delta Q^2}{Z_{i-1,j} + Z_{ij}},\\ \alpha^2 &= \Delta Q^3,\\ \alpha^3 &= \frac{\Delta Q^1 + Z_{i-1,j}\,\Delta Q^2}{Z_{i-1,j} + Z_{ij}}.\end{aligned} \tag{21.21}$$

The subscript $i - 1/2,\ j$ has been omitted from α and ΔQ here for clarity.

These reduce to (21.14) in the case of constant impedance. As usual, the fluctuations $\mathcal{A}^-\Delta Q$ and $\mathcal{A}^+\Delta Q$ are given by the product of the waves and wave speeds,

$$\mathcal{A}^-\Delta Q_{i-1/2,j} = s^1_{i-1/2,j}\mathcal{W}^1_{i-1/2,j}, \qquad \mathcal{A}^+\Delta Q_{i-1/2,j} = s^3_{i-1/2,j}\mathcal{W}^3_{i-1/2,j},$$

where $s^1_{i-1/2,j} = -c_{i-1,j}$ and $s^3_{i-1/2,j} = c_{ij}$ are the appropriate wave speeds.

21.5.1 Transverse Propagation

The right-going fluctuation $\mathcal{A}^+\Delta Q$ is split into up-going and down-going fluctuations $\mathcal{B}^+\mathcal{A}^+\Delta Q$ and $\mathcal{B}^-\mathcal{A}^+\Delta Q$ that modify the fluxes $\tilde{G}_{i,j+1/2}$ and $\tilde{G}_{i,j-1/2}$ above and below the cell (i, j), respectively. To compute the down-going fluctuation $\mathcal{B}^-\mathcal{A}^+\Delta Q$, for example, we need to decompose the vector $\mathcal{A}^+\Delta Q$ into eigenvectors corresponding to up-going and down-going waves arising from the interface at $(i, j - 1/2)$,

$$\mathcal{A}^+\Delta Q_{i-1/2,j} = \beta^1 \begin{bmatrix} -Z_{i,j-1} \\ 0 \\ 1 \end{bmatrix} + \beta^2 \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} + \beta^3 \begin{bmatrix} Z_{ij} \\ 0 \\ 1 \end{bmatrix}, \tag{21.22}$$

with speeds $-c_{i,j-1}, 0, c_{ij}$ respectively. Solving this linear system gives

$$\beta^1 = \frac{-\left(\mathcal{A}^+\Delta Q_{i-1/2,j}\right)^1 + \left(\mathcal{A}^+\Delta Q_{i-1/2,j}\right)^3 Z_{ij}}{Z_{i,j-1} + Z_{ij}}, \tag{21.23}$$

where $(\mathcal{A}^+\Delta Q_{i-1/2,j})^p$ is the pth element of the vector $\mathcal{A}^+\Delta Q_{i-1/2,j}$. The coefficient β^1 is the only one needed to compute the down-going fluctuation, which is obtained by multiplying the first wave in (21.22) by the speed of this down-going wave,

$$\mathcal{B}^-\mathcal{A}^+\Delta Q_{i-1/2,j} = -c_{i,j-1}\beta^1 \begin{bmatrix} -Z_{i,j-1} \\ 0 \\ 1 \end{bmatrix}. \tag{21.24}$$

To compute the up-going fluctuation $\mathcal{B}^-\mathcal{A}^+\Delta Q$, we instead decompose the vector $\mathcal{A}^+\Delta Q$ into eigenvectors corresponding to up-going and down-going waves arising from the interface at $(i, j+1/2)$,

$$\mathcal{A}^+\Delta Q_{i-1/2,j} = \beta^1 \begin{bmatrix} -Z_{ij} \\ 0 \\ 1 \end{bmatrix} + \beta^2 \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} + \beta^3 \begin{bmatrix} Z_{i,j+1} \\ 0 \\ 1 \end{bmatrix}, \tag{21.25}$$

with speeds $-c_{ij}$, 0, $c_{i,j+1}$ respectively. Solving this linear system gives

$$\beta^3 = \frac{\left(\mathcal{A}^+\Delta Q_{i-1/2,j}\right)^1 + \left(\mathcal{A}^+\Delta Q_{i-1/2,j}\right)^3 Z_{i,j+1}}{Z_{ij} + Z_{i,j+1}}. \tag{21.26}$$

The coefficient β^3 is the only one needed to compute the up-going fluctuation, which is obtained by multiplying the third wave in (21.25) by the speed of this up-going wave,

$$\mathcal{B}^+\mathcal{A}^+\Delta Q_{i-1/2,j} = c_{i,j+1}\beta^3 \begin{bmatrix} Z_{i,j+1} \\ 0 \\ 1 \end{bmatrix}. \tag{21.27}$$

The left-going fluctuation $\mathcal{A}^-\Delta Q_{i-1/2,j}$ must similarly be decomposed in two different ways to compute the transverse fluctuations $\mathcal{B}^-\mathcal{A}^-\Delta Q_{i-1/2,j}$ and $\mathcal{B}^+\mathcal{A}^-\Delta Q_{i-1/2,j}$. The formulas are quite similar with i replaced by $i-1$ in the sound speeds c and impedances Z above. See [claw/book/chap21/corner/rpt2acv.f].

Example 21.1. Figure 21.1(a) shows a heterogeneous medium with piecewise constant density and bulk modulus. Figure 21.2 shows the calculation of an acoustic pulse propagating in this medium. The pulse is initially a square rightward-propagating plane-wave pulse in pressure, as indicated in Figure 21.1(b). The pressure perturbation is nonzero only for

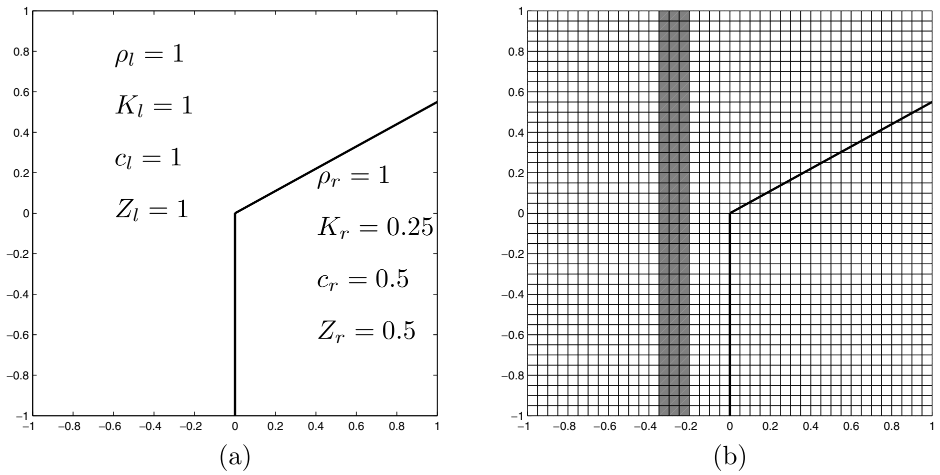

Fig. 21.1. (a) Piecewise-constant heterogeneous material for Example 21.1. (b) Illustration of how the interface cuts through a Cartesian grid, and the initial pressure pulse.

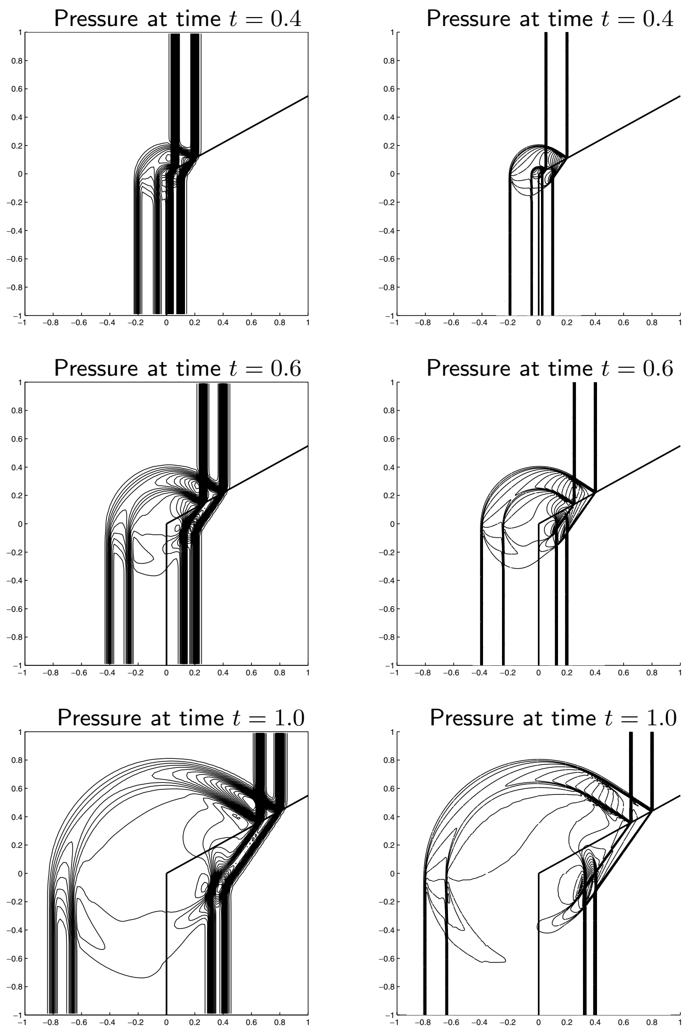

Fig. 21.2. Contours of pressure for an acoustic pulse propagating in the material shown in Figure 21.1, at three different times (from bottom to top). The calculation on the left was done on a 100×100 uniform Cartesian grid. `[claw/book/chap21/corner]` The calculation on the right is highly resolved using adaptive mesh refinement. `[claw/book/chap21/corner/amr]`

$-0.35 < x < -0.2$. When the pulse hits the interface, it is partially reflected and partially transmitted. As the pulse moves up the ramp portion of the interface, observe that the usual law of reflection is satisfied: the angle of incidence of the original pulse is equal to the angle of reflection. The transmitted wave is also oblique to the grid, at an angle determined by Snell's law that depends on the difference in wave speeds between the two media.

Two calculations are shown in Figure 21.2: a coarse-grid calculation on a 100×100 grid on the left, and a highly refined adaptive mesh refinement (AMR) calculation on the right. Fine grids are used only where required near the discontinuities in pressure. To obtain the same resolution on a uniform grid would require a 960×960 grid. The AMR code used for this computation is also part of CLAWPACK (AMRCLAW), and is described in [32].

These calculations were all performed on a Cartesian grid in spite of the fact that the interface cuts obliquely through the grid cells as illustrated in Figure 21.1(b). Values of the impedance and sound speed in each grid cell are determined by using appropriate averages of the density and bulk modulus in the cells and then computing Z and c from these. We first determine what fraction of the grid cell lies in each of the two materials, the *left* material with density ρ_l and bulk modulus K_l, and the *right* material with density ρ_r and bulk modulus K_r. If w_l, w_r is the fraction lying in each state, then we set

$$\rho_{ij} = w_l\rho_l + w_r\rho_r, \qquad K_{ij} = (w_l/K_l + w_r/K_r)^{-1}. \tag{21.28}$$

We use the arithmetic average of the densities and the harmonic average of the bulk moduli, as suggested by the discussion of Section 9.14. We then set

$$c_{ij} = \sqrt{K_{ij}/\rho_{ij}}, \qquad Z_{ij} = \rho_{ij}c_{ij}. \tag{21.29}$$

21.6 Transverse Riemann Solvers for Nonlinear Systems

We now consider a nonlinear conservation law $q_t + f(q)_x + g(q)_y = 0$ and will concentrate on the procedure we must perform at the interface between cells $(i-1, j)$ and (i, j) to split fluctuations $\mathcal{A}^\pm\Delta Q_{i-1/2,j}$ into $\mathcal{B}^\pm\mathcal{A}^\pm\Delta Q_{i-1/2,j}$. For the constant-coefficient linear problem we simply multiply by the matrices B^- and B^+, but for a nonlinear system there is no single matrix B, but rather a Jacobian matrix $g'(q)$ that depends on the data. However, if we solve Riemann problems normal to each edge by using a linearized approximate Riemann solver, as discussed in Section 15.3, then this linear approach is easily extended to the nonlinear case. In solving the Riemann problem $q_t + f(q)_x = 0$ we determined a matrix $\hat{A}$ so that the fluctuations are defined simply by multiplying $Q_{ij} - Q_{i-1,j}$ by $\hat{A}^-$ and $\hat{A}^+$. The matrix $\hat{A}$ depends on certain averaged values obtained from the states $Q_{i-1,j}$ and Q_{ij}. To define the transverse Riemann solver we can now simply use these same averaged values to define a matrix $\hat{B}$ that approximates $g'(q)$ near the interface. The transverse Riemann solver then returns

$$\begin{aligned} \mathcal{B}^-\mathcal{A}^*\Delta Q &= \hat{B}^-(\mathcal{A}^*\Delta Q),\\ \mathcal{B}^+\mathcal{A}^*\Delta Q &= \hat{B}^+(\mathcal{A}^*\Delta Q). \end{aligned} \tag{21.30}$$

This is illustrated in the next section for the shallow water equations. Examples for the Euler equations can be found on the webpage `[claw/book/chap21/euler]`.

21.7 Shallow Water Equations

The numerical methods developed above for multidimensional acoustics can be extended easily to nonlinear systems such as the shallow water equations. For the dimensional-splitting method we only need a normal Riemann solver (`rpn2` in CLAWPACK). This is essentially identical to the one-dimensional Riemann problem for the shallow water equations with a passive tracer as discussed in Section 13.12.1. In practice an approximate

Riemann solver is typically used, e.g., the Roe solver developed in Section 15.3.3. This is very easily extended to the two-dimensional case. In the x-direction, for example, the velocity v does not affect the nonlinear waves, and so the Roe averages $\bar{h}$ and $\hat{u}$ are computed as in (15.32) and (15.35) respectively,

$$\bar{h} = \frac{1}{2}(h_l + h_r), \qquad \hat{u} = \frac{\sqrt{h_l}\,u_l + \sqrt{h_r}\,u_r}{\sqrt{h_l} + \sqrt{h_r}}, \tag{21.31}$$

where $h_l = h_{i-1,j}$, $h_r = h_{ij}$, etc. We also need an average value $\hat{v}$, discussed below. The Roe matrix is then

$$\hat{A} = \begin{bmatrix} 0 & 1 & 0 \\ -\hat{u}^2 + g\bar{h} & 2\hat{u} & 0 \\ -\hat{u}\hat{v} & \hat{v} & \hat{u} \end{bmatrix}, \tag{21.32}$$

with eigenvalues and eigenvectors

$$\begin{aligned} &\hat{\lambda}^{x1} = \hat{u} - \hat{c}, && \hat{\lambda}^{x2} = \hat{u}, && \hat{\lambda}^{x3} = \hat{u} + \hat{c}, \\ &\hat{r}^{x1} = \begin{bmatrix} 1 \\ \hat{u} - \hat{c} \\ \hat{v} \end{bmatrix}, && \hat{r}^{x2} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, && \hat{r}^{x3} = \begin{bmatrix} 1 \\ \hat{u} + \hat{c} \\ \hat{v} \end{bmatrix}, \end{aligned} \tag{21.33}$$

where $\hat{c} = \sqrt{g\bar{h}}$. Entropy fixes and limiters are applied just as in one dimension.

For the average velocity $\hat{v}$ it seems possible to simply use the Roe average

$$\hat{v} = \frac{\sqrt{h_l}\,v_l + \sqrt{h_r}\,v_r}{\sqrt{h_l} + \sqrt{h_r}} \tag{21.34}$$

and obtain good results in general. This does not, however, give a matrix $\hat{A}$ that satisfies the usual requirement

$$\hat{A}(Q_r - Q_l) = f(Q_r) - f(Q_l) \tag{21.35}$$

for the Roe matrix. Choosing $\bar{h}$ and $\hat{u}$ as in (21.31) insures that the first two equations of the system (21.35) hold, but the third equation requires

$$-\hat{u}\hat{v}\delta^1 + \hat{v}\delta^2 + \hat{u}\delta^3 = h_r u_r v_r - h_l u_l v_l,$$

where $\delta = Q_r - Q_l$. This equation can be used to define $\hat{v}$, obtaining

$$\begin{aligned} \hat{v} &= \frac{(h_r u_r v_r - h_l u_l v_l) - \hat{u}(h_r u_r - h_l u_l)}{(h_r u_r - h_l u_l) - \hat{u}(h_r - h_l)} \\ &= \frac{a_l v_l + a_r v_r}{a_l + a_r}, \end{aligned} \tag{21.36}$$

where

$$a_l = h_l(\hat{u} - u_l), \qquad a_r = h_r(u_r - \hat{u}).$$

A difficulty with this approach is that the denominator is zero whenever $u_l = u_r$, and this case must be handled separately. In practice the weighting (21.34) has been successfully used.

To use the method developed in Section 21.2, we must also provide a transverse Riemann solver, similar to the one developed in Section 18.4 for the two-dimensional acoustics equations. For the nonlinear shallow water equations we wish to take the right-going flux $\mathcal{A}^+\Delta Q_{i-1/2,j}$, for example, and split it into an up-going part $\mathcal{B}^+\mathcal{A}^+\Delta Q_{i-1/2,j}$ and a down-going part $\mathcal{B}^-\mathcal{A}^+\Delta Q_{i-1/2,j}$. As developed in Sections 21.1 through 21.6, the basic idea is to split the vector $\mathcal{A}^+\Delta Q_{i-1/2,j}$ into eigenvectors of a matrix B approximating $g'(q)$. For this nonlinear system the Jacobian varies with q. However, we can use the Roe-averaged quantities $\bar{h}$, $\hat{u}$, and $\hat{v}$ to define a natural approximate matrix $\hat{B}$ to use for this decomposition. The eigenvalues and eigenvectors are as in (18.42) but with (h, u, v) replaced by the Roe averages. The formula (21.30) is then used to define the transverse fluctuations.

Normal and transverse Riemann solvers for the shallow water equations can be found in the directory [claw/book/chap21/radialdam]. This approach has been used in the example shown below.

21.7.1 A Radial Dam-Break Problem

Figure 21.3 shows a radial dam-break problem for the two-dimensional shallow water equations. The depth is initially $h = 2$ inside a circular dam and $h = 1$ outside. When the dam is removed, a shock wave travels radially outwards while a rarefaction wave moves inwards. This is similar to the structure of the one-dimensional dam-break Riemann problem. The fluid itself is moving outwards, and is accelerated either abruptly through the shock wave or smoothly through the rarefaction wave. Figure 21.4 shows the time evolution of both the depth and the radial momentum as a function of r, the distance from the origin, over a longer time period. This was computed by solving the one-dimensional equations

$$\begin{aligned} h_t + (hU)_r &= -\frac{hU}{r}, \\ (hU)_t + \left(hU^2 + \frac{1}{2}gh^2\right)_r &= -\frac{hU^2}{r}, \end{aligned} \tag{21.37}$$

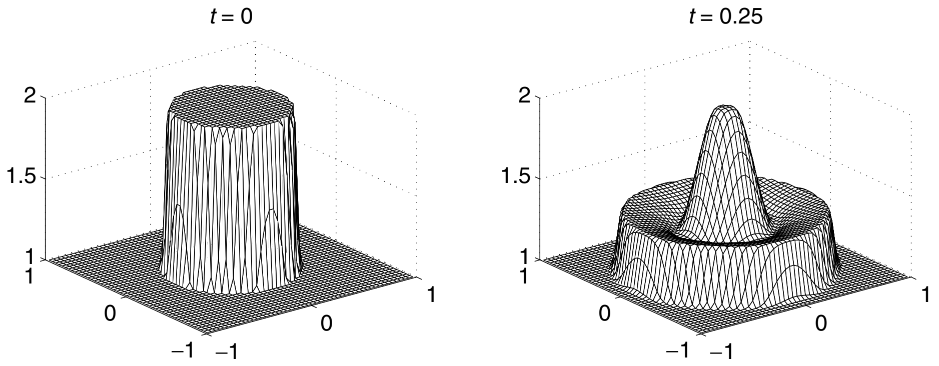

Fig. 21.3. Depth of water h for a radial dam-break problem, as computed on a 50 × 50 grid.

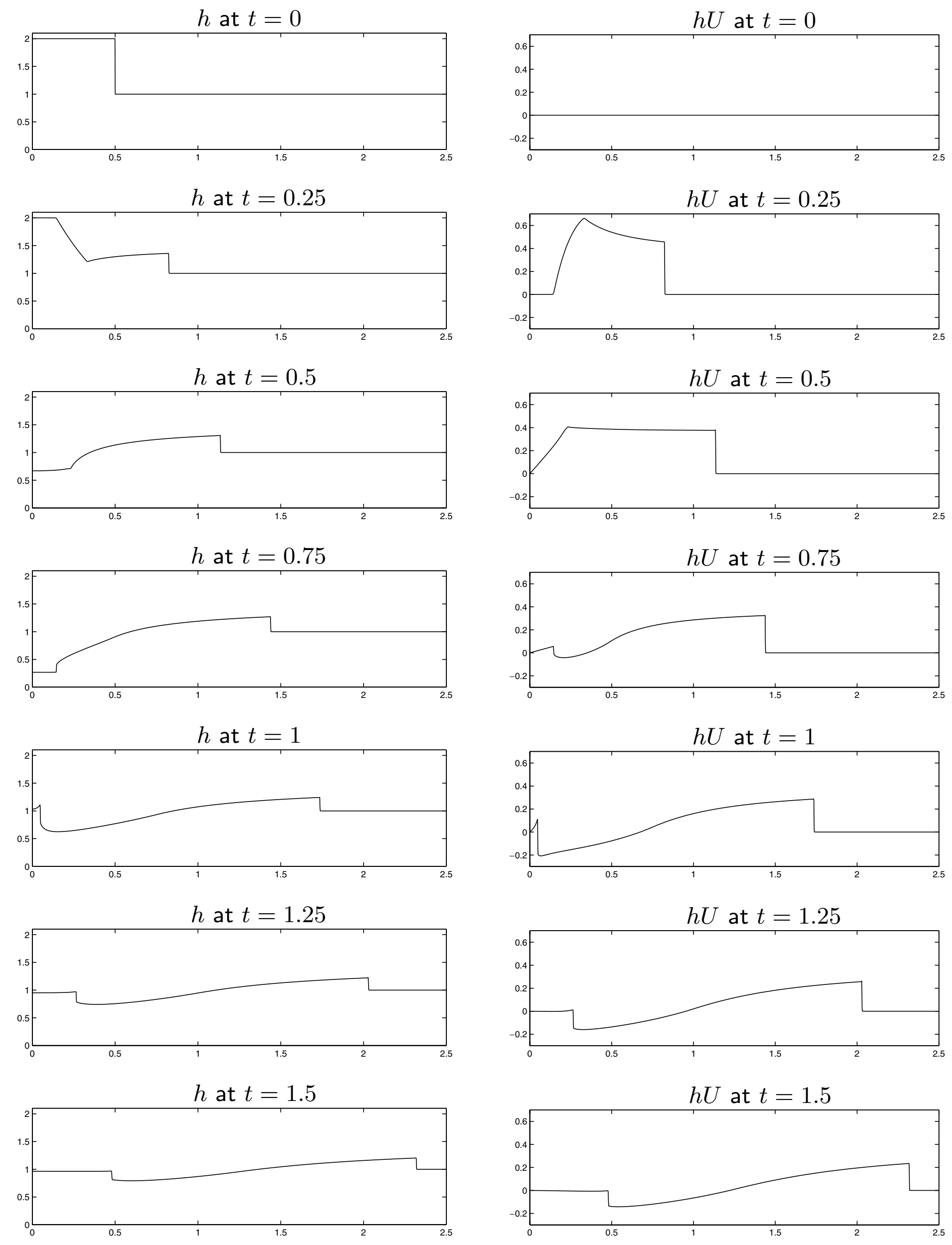

Fig. 21.4. Solution to the radial dam-break problem as a function of r. Left: depth h. Right: radial momentum hU. [claw/book/chap21/radialdam/1drad]

where $U(r, t)$ is the radial velocity. These follow from (18.52) with the hydrostatic pressure (18.39). Note that at time $t = 0.25$ the depth and momentum are no longer constant between the shock and rarefaction wave as in the one-dimensional Riemann problem. This is due to the source terms in (21.37), which physically arise from the fact that the fluid is spreading out and it is impossible to have constant depth and constant nonzero radial velocity.

Once the rarefaction wave hits the origin, all of the available fluid has been accelerated outwards. At this point the depth at the center begins to fall and ultimately falls below $h = 1$. At later times this depression in the water leads to inward acceleration of the fluid, filling this hole back in again. Note that after about $t = 0.75$ there is a region of negative radial momentum. As the fluid starts to flow inward, a second shock wave forms where the converging inward flow is decelerated back to zero velocity. This shock wave is already visible at time $t = 0.75$, and by $t = 1$ it has passed through the origin. At later times the structure has a basic N-wave form. The initially quiescent fluid is accelerated outwards through the leading shock, the velocity falls through a rarefaction wave to an inward velocity, and then the fluid is decelerated back to zero velocity through the second shock. These shocks weaken as they propagate outward, due to the radial spreading, and for large time will be essentially N-waves.

Figure 21.5 shows contour plots of numerical results computed using the unsplit method on a 125×125 grid over the domaim $[-2.5, 2.5] \times [-2.5 \times 2.5]$, along with scatterplots of the computed solution vs. distance from the origin. This is a fairly coarse grid for this problem, the same resolution $\Delta x = \Delta y = 0.04$ as shown in Figure 21.3 but on a larger

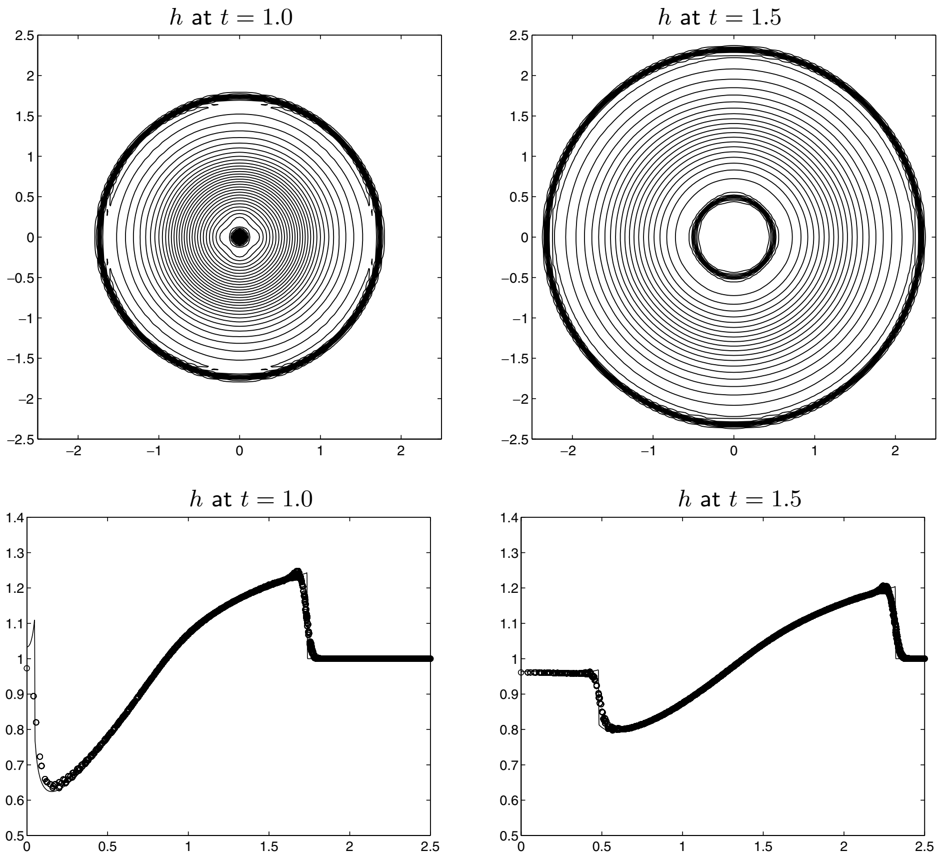

Fig. 21.5. Computed solutions for the radial dam-break problem. Top: contour plots of the depth at two times. Bottom: scatterplots of the depth vs. distance from the origin at the same two times. Contour levels are 0.61 : 0.02 : 1.31. `[claw/book/chap21/radialdam]`

domain. The depth is shown at two different times. At $t = 1.0$ the shock near the origin cannot be resolved on this grid. The structure looks correct elsewhere, however. At $t = 1.5$ the basic structure is captured well everywhere. Note in particular that the depth near the origin stabilizes at a value $h \approx 0.96$ that is well captured also in the two-dimensional results.

21.8 Boundary Conditions

Boundary conditions in two (or three) space dimensions can be handled in much the same way as in one dimension. The grid on the desired computational domain consists of *interior cells* that we will label by $i = 1, 2, \ldots, m_x$ and $j = 1, 2, \ldots, m_y$. This grid is extended by introducing a set of *ghost cells* on all sides, for $i = 1 - m_{\text{BC}}, \ldots, 0$ and $i = m_x + 1, \ldots, m_x + m_{\text{BC}}$, and for $j = 1 - m_{\text{BC}}, \ldots, 0$ and $j = m_y + 1, \ldots, m_y + m_{\text{BC}}$. Figure 21.6 shows a portion of such an extended grid for the case $m_{\text{BC}} = 2$. At the beginning of each time step the ghost-cell values are filled, based on data in the interior cells and the given boundary conditions, and then the algorithm of choice is applied over the extended domain. How many rows of ghost cells are needed depends on the stencil of the algorithm. The high-resolution algorithms presented in Chapters 20 and 21 generally require two rows of ghost cells as shown in the figure. This allows us to solve a Riemann problem at the the boundary of the original domain and also one additional Riemann problem outside the domain. The waves from this Riemann problem do not enter the domain and so do not affect the solution directly, but are used to limit the waves arising from the original boundary. In the discussion below we assume $m_{\text{BC}} = 2$, but it should be clear how to extend each of the boundary conditions for larger values of m_{BC}.

The manner in which ghost cells are filled depends on the nature of the given boundary conditions. Periodic boundary conditions, for example, are easy to apply (as in the one-dimensional case of Section 7.1) by simply copying data from the opposite side of the grid. Below we will consider other standard cases, extending the approaches that were developed

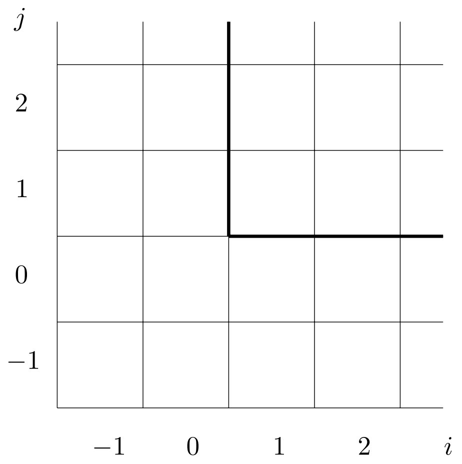

Fig. 21.6. The lower left corner of a typical computational domain is shown as the dark line. Interior grid cells are labeled with $i = 1, 2, \ldots$ and $j = 1, 2, \ldots$. The domain is extended with $m_{\text{BC}} = 2$ rows of ghost cells on each side.

in Chapter 7 to multidimensional problems. In particular, we consider solid-wall boundary conditions and the use of extrapolation at outflow boundaries. All of these are implemented in the default CLAWPACK routine `[claw/clawpack/2d/lib/bc2.f]`.

21.8.1 Dimensional Splitting

We will primarily concentrate on the techniques needed when unsplit methods of the form (19.10) or (19.19) are used, in which the data Q^n is used to compute all fluctuations and fluxes in both the x- and y-directions. If a dimensional-splitting method is used, as described in Section 19.5, then additional issues arise. If we sweep first in the x-direction to obtain Q^* from Q^n, and then sweep in the y-direction to obtain Q^{n+1} from Q^* (using the Godunov splitting), we will need to specify boundary conditions for Q^* that may differ from the physical boundary conditions originally given for q. Some discussion of this point was given in Section 17.9 in the context of fractional-step methods for source terms. For dimensional splitting, appropriate ghost-cell values for Q^* can often be obtained by first extending Q^n to all ghost cells and then sweeping over the rows of ghost cells along the top and bottom of the grid ($j = -1, 0$ and $j = m_y + 1, m_y + 2$) as well as over the rows of interior cells ($j = 1, 2, \ldots, m_y$). This modifies the ghost-cell values by solving the same one-dimensional equation as in the interior, and gives Q^*-values in these cells that can now be used as the ghost-cell values in the y-sweeps.

Note that if we wish to use the Strang splitting (19.29), then the ghost-cell values along the left and right edges ($i = -1, 0, m_x + 1, m_x + 2$) must also be updated to Q^* so that we can apply y-sweeps to these rows of cells as well as in the interior. Then we will have Q^{**}-values in these ghost cells, which are needed in taking the final x-sweep to obtain Q^{n+1} from Q^{**}. To do this requires that we have twice as many ghost cells (at least along the left and right boundaries), so that Q^* can be obtained in the ones where Q^{**} is ultimately needed.

21.8.2 Unsplit Wave-Propagation Algorithms

Even if unsplit algorithms of the form (19.19) are used, it may be necessary to sweep over the rows of ghost cells as well as the interior cells in order to properly implement the multidimensional algorithms. For example, the implementation of the wave-propagation algorithm discussed in Section 21.2 uses the solution to the Riemann problem between cells $\mathcal{C}_{i-1,j}$ and $\mathcal{C}_{ij}$ to update the fluxes $\tilde{G}_{i-1,j+1/2}$ and $\tilde{G}_{i,j+1/2}$. Referring to Figure 21.6, we see that when $j = 0$ we must solve the Riemann problems in this row of ghost cells in order to obtain proper values of the fluxes $\tilde{G}_{i,1/2}$. These in turn are used to update the interior values Q_{i1}.

21.8.3 Solid Walls

In many of the problems we have considered (e.g., acoustics, shallow water equations, gas dynamics), the solution variables include velocity or momentum components in each spatial dimension. A common boundary condition is that the velocity normal to a wall should be zero, corresponding to a solid wall that fluid cannot pass through. As in one dimension, we can implement this boundary condition by a suitable extension of the solution to the ghost cells. Consider the left edge of the domain, for example, where the x-component of the velocity should vanish. We first extrapolate all components in a symmetric manner, setting

$$Q_{0,j} = Q_{1,j}, \qquad Q_{-1,j} = Q_{2,j} \qquad \text{for } j = 1, 2, \ldots, m_y. \tag{21.38}$$

We then negate the component of Q_{ij} (for $i = -1, 0$) that corresponds to the x-component of velocity or momentum. We perform a similar extension at the right boundary. At the bottom boundary we extrapolate

$$Q_{i,0} = Q_{i,1}, \qquad Q_{i,-1} = Q_{i,2} \qquad \text{for } i = -1, 0, 1, \ldots, m_x + 1, m_x + 2, \tag{21.39}$$

and then negate the y-component of the velocity or momentum in these ghost cells. Note that we apply this procedure in the rows of ghost cells (e.g., $i = -1, 0$) as well as in the interior cells in order to insure that all the ghost cells shown in Figure 21.6 are filled, including the four corner cells where i and j both have values 0 or -1. One should always insure that these corner cells are properly filled, especially if combinations of different boundary conditions are used at the two adjacent sides.

Note that in the procedure just outlined, the tangential component of velocity is simply extrapolated from the interior. For the problems we have considered (acoustics, shallow water, gas dynamics), it really doesn't matter what value the tangential velocity has in the ghost cells, since any jump in this quantity will propagate with zero normal velocity and will not affect the solution in the interior cells. This is due to the symmetry of the extrapolated data, which results in the contact discontinuity in the Riemann solution having zero velocity, properly mimicking a stationary wall. Hence any tangential velocity is allowed by these boundary conditions.

21.8.4 No-Slip Boundary Condition

In many fluid dynamics problems there is another physical boundary condition we might wish to impose at a solid wall. The *no-slip boundary condition* states that the tangential velocity should also vanish at the wall along with the normal velocity, so that fluid adjacent to the wall is stationary. This is expected due to friction between the wall and fluid molecules, which keeps molecules from slipping freely along the wall. However, this friction is present only in viscous fluids, and hyperbolic equations only model inviscid fluids, so we are not able to impose the no-slip condition in these models. If fluid viscosity is introduced, we obtain a parabolic equation (e.g., the Navier–Stokes equations instead of the inviscid Euler equations) that allows (in fact, requires) more boundary conditions to be specified.

If the physical viscosity of the fluid is very small relative to the typical fluid velocity away from the wall (i.e., if the *Reynolds number* is large), then there will often be a thin *boundary layer* adjacent to the wall in which the tangential velocity rapidly approaches the zero velocity of the wall. The thickness of this layer depends on the magnitude of the viscosity ϵ and often vanishes as $\epsilon \to 0$. As in the case of shock waves, the inviscid hyperbolic equation attempts to model the $\epsilon = 0$ limit.

In some applications this is a suitable approximation. For example, in many aerodynamics problems the thickness of the physical boundary layer on the surface of a body is much smaller than a computational cell. For this reason the inviscid Euler equations are often used rather than the more expensive Navier–Stokes equations. However, caution must be used, since in some problems the viscous effects at the boundary do have a substantial influence on the global solution, even when the viscosity is very small. This is particularly true if the geometry is such that the boundary layer separates from the wall at some point (as must happen at the trailing edge of a wing, for example). Then the vorticity generated by a no-slip boundary will move away from the wall and perhaps lead to large-scale turbulence that persists even when ϵ is extremely small, but would not be seen in an ideal inviscid fluid. The numerical viscosity that is inherent in any "inviscid" algorithm can lead to similar effects, but may mimic flow at the wrong Reynolds number. A full discussion of these issues is beyond the scope of this book.

21.8.5 Extrapolation and Absorbing Boundary Conditions

For many problems we must use a computational domain that is smaller than the physical domain, particularly if we must cut off an essentially infinite domain at some point to obtain a finite computational domain, as already discussed in Section 7.3.1. We then wish to impose boundary conditions that allow us to compute on this smaller domain and obtain results that agree well with what would be computed on a larger domain. If the computational domain is large enough for the problem of interest, then we expect that there will only be outgoing waves at the boundary of this domain. There should not be substantial incoming waves unless they have a known form (as from some known external source) that can be imposed as part of the boundary conditions, as was done in Section 7.3.2 in one dimension. Here we will assume there should be no incoming waves, in which case our goal is to impose boundary conditions on the computational domain that are *nonreflecting*, or *absorbing*, and that allow any outgoing waves to disappear without generating spurious incoming waves.

It may be that the outgoing waves should interact outside the computational domain in such a way that incoming waves are generated, which should appear at the boundary at a later time. In general we cannot hope to model such processes via the boundary conditions, and this would be an indication that our computational domain is simply not large enough to capture the full problem.

In one space dimension, we saw in Section 7.3.1 that quite effective absorbing boundary conditions can be obtained simply by using zero-order extrapolation. This idea can be extended easily to more dimensions as well. For example, along the left and bottom edge we would set

$$Q_{0j} = Q_{1j}, \quad Q_{-1,j} = Q_{1j} \qquad \text{for } j = 1, 2, \ldots, m_y, \tag{21.40}$$

and then

$$Q_{i0} = Q_{i1}, \quad Q_{i,-1} = Q_{i1} \qquad \text{for } i = -1, 0, \ldots, m_x + 2. \tag{21.41}$$

Note that we have filled the corner ghost cells in Figure 21.6 as well as the edge ghost cells. The value obtained in all four of the corner cells is Q_{11}. The same value would be obtained if we reversed the order above and first extrapolated in y and then in x.

This simple approach to absorbing boundary conditions often works very well in multi-dimensional problems. As in one dimension, its success rests on the fact that the Riemann problem at the edge of the computational domain has the same data on either side, resulting in zero-strength waves and in particular no incoming waves.

While surprisingly effective, this approach is unfortunately not quite as effective as in one dimension, except in the special case of plane waves exiting normal to the boundary of the domain. An outgoing wave at some angle to the grid can be viewed as a superposition of various waves moving in the x- and y-directions. Some of these waves should be incoming from the perspective of the computational boundary. This is clear from the fact that solving a Riemann problem in x or y in the midst of such an oblique wave will result in nontrivial waves moving with both positive and negative speeds. Using zero-order extrapolation will result in the loss of some of this information. The fact that there are no incoming waves normal to the boundary results in an incorrect representation of the outgoing oblique wave, which appears computationally as an incoming reflected wave. The strength of this reflection

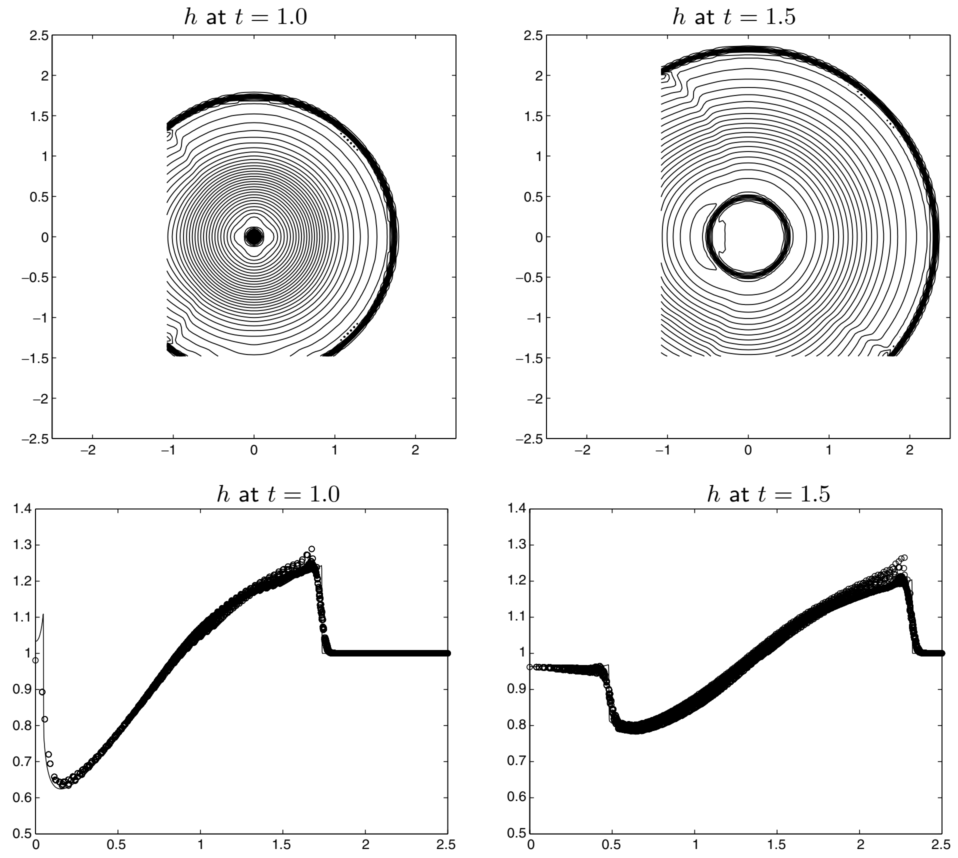

Fig. 21.7. Computed solutions for the radial dam-break problem on a reduced domain $[-1.1, 2.5] \times [-1.5, 2.5]$ with zero-order extrapolation boundary conditions. Top: contour plots of the depth at two times. Bottom: scatterplots of the depth vs. distance from the origin at the same two times. Compare Figure 21.5, where the same resolution has been used on a larger domain. Contour levels are 0.61 : 0.02 : 1.31. `[claw/book/chap21/radialdamabc]`

generally depends on the angle of the wave to the boundary and is typically worst in corners. This is illustrated in Example 21.2 below, which shows the effects of this error. Still, these boundary conditions are fairly effective considering their simplicity, and are often good enough in practice.

There is an extensive literature on more sophisticated approaches to specifying absorbing boundary conditions for wave-propagation problems. See, for example, [1], [21], [28], [117], [123], [176], [197], [230].

Example 21.2. As an example, consider the radial dam-break problem for the shallow water equations described in Section 21.7.1. We now solve this same problem on a 90×100 grid in a smaller domain $[-1.1, 2.5] \times [-1.5, 2.5]$ rather than the full domain $[-2.5, 2.5] \times [-2.5, 2.5]$ used to compute the results shown in Figure 21.5. The same mesh size $\Delta x = \Delta y = 0.04$ is used. As seen in the contour plots of Figure 21.7, small-amplitude reflected waves are generated as the shock wave leaves the computational domain. The effect of these errors is also observed in the scatterplots of depth vs. distance from the origin.