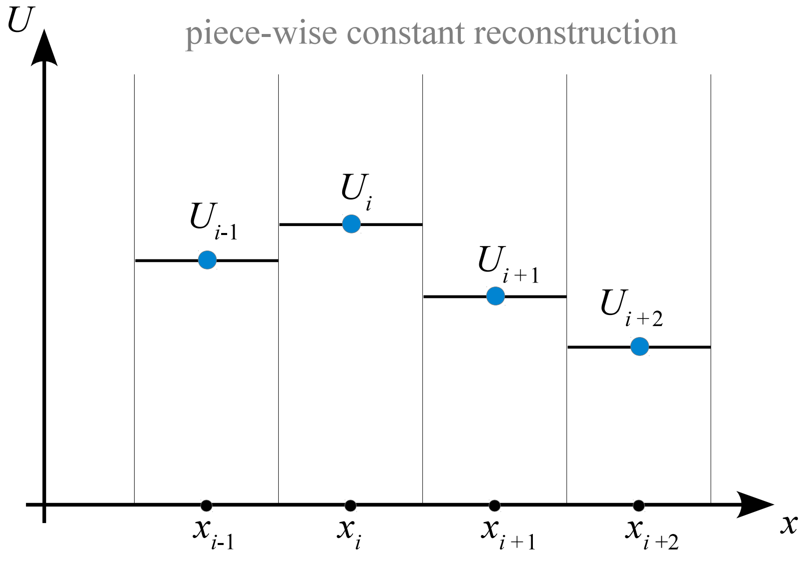
8 Multidimensional Fluid Solvers and Higher-order Schemes

8.1 Short repeat: Godunov's method and Riemann solvers

It is useful to introduce another interpretation of common finite-volume discretizations of fluid dynamics, so-called Reconstruct-Evolve-Average (REA) schemes. We use this here for a short repeat of Godunov's important method, and the way Riemann solvers come into play in it.



An REA update scheme of a hydrodynamical system discretized on a mesh can be viewed as a sequence of three steps:

1. **Reconstruct:** Using the cell-averaged quantities, this defines the run of these quantities everywhere in the cell. In the sketch, a piece-wise constant reconstruction is assumed, which is the simplest procedure one can use and leads to 1st order accuracy.

2. **Evolve:** The reconstructed state is then evolved forward in time by Δt. In Godunov's approach, this is done by treating each cell interface as a piece-wise constant initial value problem which is solved with the Riemann solver exactly

or approximately. This solution is valid for arbitrary times, but in practice one needs to limit the timestep Δt such that the waves emanating from the opposite sides of a cell do not yet start to interact (at this point, the solution constructed for an isolated interface would become invalid).

3. **Average:** The evolved solution of the previous step is averaged at time Δt to compute new average states $\mathbf{U}^{n+1}$ in each cell in a conservative fashion. Then the whole cycle repeats again.

Fortunately, the averaging step does not need to be done explicitly. Instead it can be carried out by accounting for the fluxes that enter or leave the control volume of the cell. Consider the following integral over the region $[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}] \times [t_n, t_{n+1}]$, corresponding to cell i and the course of the timestep $\Delta t = t_{n+1} - t_n$. We have

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathrm{d}x \int_{t_n}^{t_{n+1}} \mathrm{d}t \, (\partial_t \mathbf{U} + \partial_x \mathbf{F}) = 0 \tag{8.1}$$

because the solution we evolve in Godunov's approach is actually an exact solution of the hyperbolic equations. This then yields

$$\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}(x, t_{n+1})\mathrm{d}x - \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \mathbf{U}(x, t_n)\mathrm{d}x + \int_{t_n}^{t_{n+1}} \mathbf{F}(x_{i+\frac{1}{2}}, t)\mathrm{d}t - \int_{t_n}^{t_{n+1}} \mathbf{F}(x_{i-\frac{1}{2}}, t)\mathrm{d}t = 0. \tag{8.2}$$

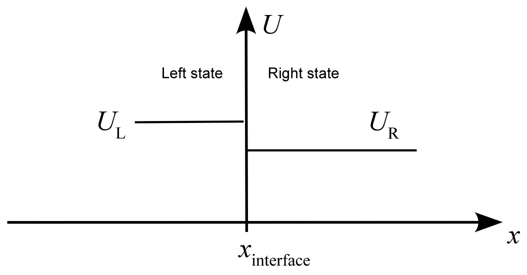
Notice that the first term is simply the average of $\mathbf{U}$ at the new time over the whole cell of length $\Delta x = x_{i+\frac{1}{2}} - x_{i+\frac{1}{2}}$, hence this is the cell average $\mathbf{U}_{n+1}$ we are looking for, multiplied by Δx. It may seem as if the fluxes at the cell interfaces are needed as a function of time during the timestep in order to do the time integration of the fluxes. However, in Godunov's method we estimate the fluxes through Riemann problems with piece-wise constant initial state. Here the solutions are scale-invariant in the variable x/t, and the fluxes at the initial interfaces are in fact constant in time. We can hence write

$$\mathbf{F}(x_{i+\frac{1}{2}}, t) = \mathbf{F}^{\star}_{i+\frac{1}{2}}, \tag{8.3}$$

where the asterisk is meant to indicate that the corresponding flux is the appropriate flux from the Riemann solver at the interface $x_{i+\frac{1}{2}}$. Equation (8.2) therefore gives:

$$\mathbf{U}_{n+1} = \mathbf{U}_n + \frac{\Delta t}{\Delta x} \left(\mathbf{F}^{\star}_{i-\frac{1}{2}} - \mathbf{F}^{\star}_{i+\frac{1}{2}} \right) \tag{8.4}$$

which is the update formula for the state of a cell in the REA schemes. What is hence needed is a prescription to either exactly or approximately solve the Riemann problem for a piece-wise linear left and right state that are brought into contact at time $t = t_n$.



Formally, this can be written as

$$\mathbf{F}^\star = \mathbf{F}_{\text{Riemann}}(\mathbf{U}_L, \mathbf{U}_R). \tag{8.5}$$

In practice, a variety of approximate Riemann solvers $\mathbf{F}_{\text{Riemann}}$ are commonly used in the literature, as discussed earlier in the lecture. For the ideal gas and for isothermal gas, it is also possible to solve the Riemann problem exactly, but not in closed form (i.e. the solution involves an iterative root finding of a non-linear equation).

There are now two main issues left:

- How can this be extended to multiple spatial dimensions?
- How can it be extended such that a higher order integration accuracy both in space and time is reached?

We'll discuss these issues next.

8.2 Extending the 1D problem to multiple dimensions

So far, we have considered *one-dimensional* hyperbolic conservation laws of the form

$$\partial_t \mathbf{U} + \partial_x \mathbf{F}(\mathbf{U}) = 0. \tag{8.6}$$

For example, for isothermal gas with soundspeed c_s, the state vector $\mathbf{U}$ and flux vector $\mathbf{F}(\mathbf{U})$ are given as

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + \rho c_s^2 \end{pmatrix}, \tag{8.7}$$

where u is the velocity in the x-direction.

In three dimensions, the PDEs describing a fluid become considerably more involved. For example, the Euler equations for an ideal gas are given in explicit form as

$$\partial_t \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix} + \partial_x \begin{pmatrix} \rho u \\ \rho u^2 + P \\ \rho u v \\ \rho u w \\ \rho u(\rho e + P) \end{pmatrix} + \partial_y \begin{pmatrix} \rho u \\ \rho u v \\ \rho v^2 + P \\ \rho v w \\ \rho v(\rho e + P) \end{pmatrix} + \partial_z \begin{pmatrix} \rho w \\ \rho u w \\ \rho v w \\ \rho w^2 + P \\ \rho w(\rho e + P) \end{pmatrix} = 0 \tag{8.8}$$

These equations are often written in the following notation

$$\partial_t \mathbf{U} + \partial_x \mathbf{F} + \partial_y \mathbf{G} + \partial_z \mathbf{H} = 0. \tag{8.9}$$

Here the functions $\mathbf{F}(\mathbf{U})$, $\mathbf{G}(\mathbf{U})$ and $\mathbf{H}(\mathbf{U})$ give the flux vectors in the x-, y- and z-direction, respectively.

8.2.1 Dimensional splitting

Let us know consider the three dimensionally split problems derived from equation (8.9):

$$(1) \quad \partial_t \mathbf{U} + \partial_x \mathbf{F} = 0, \tag{8.10}$$

$$(2) \quad \partial_t \mathbf{U} + \partial_y \mathbf{G} = 0, \tag{8.11}$$

$$(3) \quad \partial_t \mathbf{U} + \partial_z \mathbf{H} = 0, \tag{8.12}$$

Note that the vectors appearing here have still the same dimensionality as in the full equations. They are *augmented* one-dimensional problems, i.e. the transverse variables still appear but spatial differentiation happens only in one direction. Because of this, these additional transverse variables do not make the 1D problem more difficult compared to the 'pure' 1D problem considered earlier, but the fluxes appearing in them still need to be included.

Now let us assume that he have a method to solve/advance each of these one-dimensional problems. We can for example express this formally through time-evolution operators $\mathcal{X}(\Delta t)$, $\mathcal{Y}(\Delta t)$, and $\mathcal{Z}(\Delta t)$, which advance the solution by a timestep Δt. Then the full time advance of the system can for example be approximated by

$$\mathbf{U}^{n+1} \simeq \mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n \tag{8.13}$$

This is one possible dimensionally split update scheme. In fact, this is the exact solution if (8.10)-(8.11) are the linear advection problem, but for more general non-linear equations it only provides a first order approximation. However, higher-order dimensionally split update schemes can also be easily constructed. For example, in two-dimensions

$$\mathbf{U}^{n+1} = \frac{1}{2}[\mathcal{X}(\Delta t)\mathcal{Y}(\Delta t) + \mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n \tag{8.14}$$

and

$$\mathbf{U}^{n+1} = \mathcal{X}(\Delta t/2)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t/2)\mathbf{U}^n \tag{8.15}$$

are second-order accurate. Similarly, for three dimensions the schemes

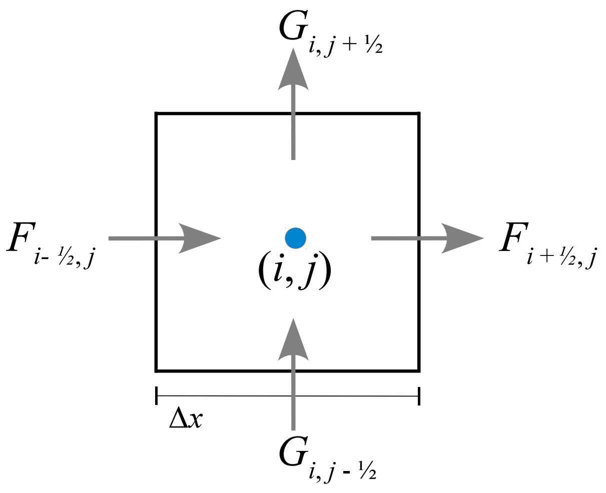
$$\mathbf{U}^{n+2} = \mathcal{X}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{Z}(\Delta t)\mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t)\mathcal{X}(\Delta t)\mathbf{U}^n \tag{8.16}$$

$$\mathbf{U}^{n+1} = \mathcal{X}(\Delta t/2)\mathcal{Y}(\Delta t/2)\mathcal{Z}(\Delta t)\mathcal{Y}(\Delta t/2)\mathcal{X}(\Delta t/2)\mathbf{U}^n \tag{8.17}$$

are second-order accurate. As a general rule of thumb, the time evolution operators have to be applied alternatingly in reverse order to reach second-order accuracy. We see that the dimensionless splitting hence reduces the problem effectively to a sequence of one-dimensional solution operations which are applied to multi-dimensional domains. Note that each one-dimensional operator leads to an update of $\mathbf{U}$, and is a complete step for the corresponding augmented one-dimensional problem. Gradients, etc., needed for the next step have then to be recomputed before the next time-evolution operator is applied. In practical applications of mesh codes, these one-dimensional solves often called sweeps.

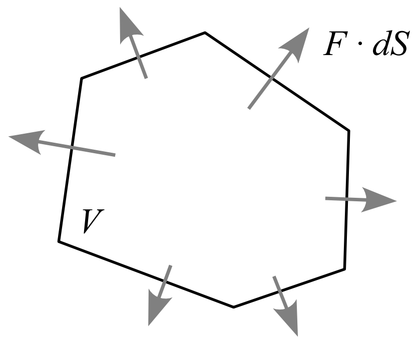
8.2.2 Unsplit schemes

In an unsplit approach, all flux updates of a cell are applied simultaneously to a cell, not sequentially. This is for example illustrated in this 2D situation.



The unsplit update of cell i, j is then given by

$$U_{i,j}^{n+1} = U_{i,j}^{n} + \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{i-\frac{1}{2},j} - \mathbf{F}_{i+\frac{1}{2},j}\right) + \frac{\Delta t}{\Delta y}\left(\mathbf{G}_{i,j-\frac{1}{2}} - \mathbf{G}_{i,j+\frac{1}{2}}\right) \tag{8.18}$$



Unsplit approaches can also be used for irregular shaped cells like those appearing in unstructured meshes. For example, integrating over a cell of volume V and denoting with $\mathbf{U}$ the cell average, we can write the cell update with the divergence theorem as

$$\mathbf{U}^{n+1} = \mathbf{U}^{n} - \frac{\Delta t}{V}\int \mathbf{F}\cdot \mathbf{dS}, \tag{8.19}$$

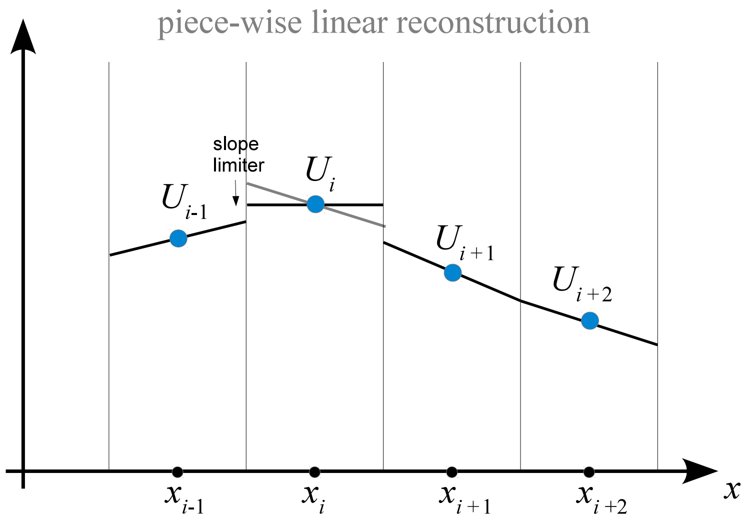
where the integration is over the whole cell surface, with outwards pointing face area vectors $\mathbf{dS}$.

8.3 Higher order schemes

We should first clarify what we mean with higher order schemes. Loosely speaking, this refers to the convergence rate of a scheme in smooth regions of a flow. For example, if we know the analytic solution $\rho(x)$ for some problem, and then obtain a numerical result ρ_i at a set of N points at locations x_i, we can ask what the typical error of the solution is. One possibility to quantify this would be through a L1 error norm. We can define

$$L1 = \frac{1}{N}\sum_i |\rho_i - \rho(x_i)|. \tag{8.20}$$

If we now measure this error quantitatively for different resolutions of the discretization, we want to find that L1 decreases with increasing N. In such a case our numerical scheme is converging, and provided we use sufficient numerical resources, we have a chance to get below any desired absolute error level. But the *rate of convergence* can be very different between different numerical schemes when applied to the same problem. If a method shows a $L1 \propto N^{-1}$ scaling, it is said to be first-order accurate; a doubling of the number of cells will then cut the error in half. A second-order method has $L1 \propto N^{-2}$, meaning that a doubling of the number of cells can actually reduce the error by a factor of 4. This much better convergence rate is of course highly desirable. It is also possible to construct schemes with still higher convergence rates, but they quickly become much more complex and computationally involved, so that one one often reaches a point of diminishing return quickly. But the extra effort one needs to make to go from first to second-order is often very small, sometimes trivially small, so that one basically should always strive to do at least this.



A first step in constructing a 2nd order extension of Godunov's method is to replace the piece-wise constant with a piece-wise linear reconstruction. This involves that one first estimates gradients for each cell (usually by a simple finite difference formula). These are then slope-limited if needed such that the linear extrapolations of the cell states to the cell interfaces do not introduce new extrema. This slope-limiting procedure is quite important; it needs to be done to avoid that real fluid discontinuities introduce large spurious oscillations into the fluid.

Given slope limited gradients, for example $\nabla\rho$ for the density, one can then estimate the left and right state adjacent to an interface $x_{i+\frac{1}{2}}$ by spatial extrapolation from the centers of the cells left and right from the interface:

$$\rho^L_{i+\frac{1}{2}} = \rho_i + (\nabla\rho)_i \frac{\Delta x}{2}, \tag{8.21}$$

$$\rho^R_{i+\frac{1}{2}} = \rho_{i+1} - (\nabla\rho)_{i+1} \frac{\Delta x}{2}. \tag{8.22}$$

The next step would in principle be to use these states in the Riemann solver. In doing this we will ignore the fact that our reconstruction has now a gradient over the cell; instead we still pretend that left and right of the interface, the fluid state can be taken as piece-wise constant as far as the Riemann solver is concerned. However, it turns out that the spatial extrapolation needs to be augmented also with a temporal extrapolation one half timestep into the future, such that the flux estimate is now effectively done in the middle of the timestep. This is necessary both to reach second-order accuracy in time and also for stability reasons.

Hence we really need to use

$$\rho^L_{i+\frac{1}{2}} = \rho_i + (\nabla\rho)_i \frac{\Delta x}{2} + \left(\frac{\partial\rho}{\partial t}\right)_i \frac{\Delta t}{2} \tag{8.23}$$

$$\rho^R_{i+\frac{1}{2}} = \rho_{i+1} - (\nabla\rho)_{i+1} \frac{\Delta x}{2} + \left(\frac{\partial\rho}{\partial t}\right)_{i+1} \frac{\Delta t}{2} \tag{8.24}$$

for extrapolating to the interfaces. More generally, this has to be done for the whole state vector of the system, i.e.

$$\mathbf{U}^L_{i+\frac{1}{2}} = \mathbf{U}_i + (\partial_x \mathbf{U})_i \, \frac{\Delta x}{2} + (\partial_t \mathbf{U})_i \, \frac{\Delta t}{2} \tag{8.25}$$

$$\mathbf{U}^R_{i+\frac{1}{2}} = \mathbf{U}_{i+1} - (\partial_x \mathbf{U})_{i+1} \, \frac{\Delta x}{2} + (\partial_t \mathbf{U})_{i+1} \, \frac{\Delta t}{2}. \tag{8.26}$$

Note that here the quantity $(\partial_x \mathbf{U})_i$ is a (slope-limited) *estimate* of the gradient in cell i, based on finite-differences plus a slope limiting procedure. Similarly, we somehow need to estimate the time derivative encoded in $(\partial_x \mathbf{U})_i$. How can this be done? One smart way to do this is to exploit the Jacobian matrix of the Euler equations. We can write the Euler equations as

$$\partial_t \mathbf{U} = -\partial_x \mathbf{F}(\mathbf{U}) = -\frac{\partial \mathbf{F}}{\partial \mathbf{U}} \, \partial_x \mathbf{U} = -\mathbf{A}(\mathbf{U}) \, \partial_x \mathbf{U}, \tag{8.27}$$

where $\mathbf{A}(\mathbf{U})$ is the Jacobian matrix. Using this, we can simply estimate the required time-derivative based on the spatial derivatives:

$$(\partial_t \mathbf{U})_i = -\mathbf{A}(\mathbf{U}_i) \, (\partial_x \mathbf{U})_i \tag{8.28}$$

Hence the extrapolation can be done as

$$\mathbf{U}^L_{i+\frac{1}{2}} = \mathbf{U}_i + \left[\frac{\Delta x}{2} - \frac{\Delta t}{2} \mathbf{A}(\mathbf{U}_i) \right] (\partial_x \mathbf{U})_i \tag{8.29}$$

$$\mathbf{U}^R_{i+\frac{1}{2}} = \mathbf{U}_{i+1} + \left[-\frac{\Delta x}{2} - \frac{\Delta t}{2} \mathbf{A}(\mathbf{U}_{i+1}) \right] (\partial_x \mathbf{U})_{i+1} \tag{8.30}$$

This procedure defines the so-called MUSCL-Hancock scheme, which is a 2nd-order accurate extension of Godunov's method.

Higher-order extensions such as the piece-wise parabolic method (PPM) start out with a higher order polynomial reconstruction. In the case of PPM, parabolic shapes are assumed in each cell instead of piece-wise linear states. The reconstruction is still guaranteed to be conservative, however, i.e. the integral underneath the reconstruction recovers the total values of the conserved variables individually in each cell. So-called ENO and WENO schemes use yet higher-order polynomials to reconstruct the state in a conservative fashion. Here many more cells in the environment need to be considered (i.e. the stencil of these methods is much larger) to robustly determine the coefficients of the reconstruction. This can for example involve a least-square fitting procedure.