Notes on Grids

1 Discretization

The physical systems we model are described by continuous mathematical functions, $f(x,t)$ and their derivatives in space and time. To represent this continuous system on a computer we must discretize it—convert the continuous function into a discrete number of points in space at one or more discrete instances in time.

There are many different discretization methods used throughout the physical sciences, engineering, and applied mathematics fields, each with their own strenghts and weaknesses. Broadly speaking, we can divide these methods into grid-based and gridless methods.

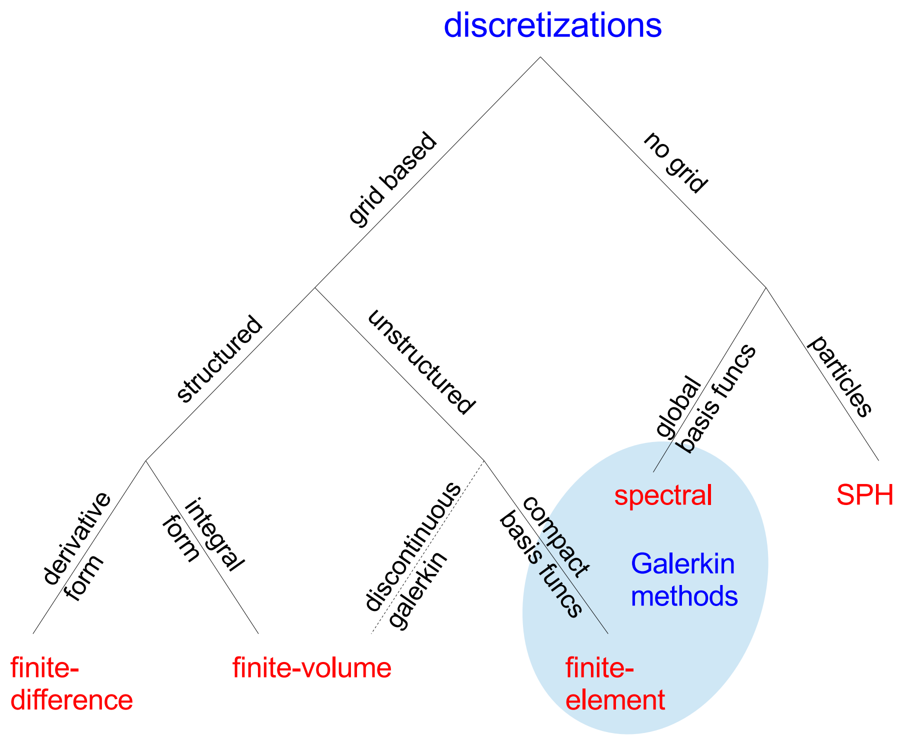

Figure 1: Taxonomy of different discretizations.

Gridless methods include those which represent the function as a superposition of continuous basis functions (e.g. sines and cosines). This is the fundamental idea behind *spectral methods*. A different class of methods are those that use discrete particles to represent the mass distribution and produce continuous functions by integrating over these particles with a suitable kernel—this is the basis of *smoothed particle hydrodynamics* (SPH). SPH is a very popular method in astrophysics.

Grid-based methods include the finite-difference, finite-volume, and finite-element methods. Finite element methods are widely used in engineering together with unstructure grids. A type of finite-element method called a *galerkin method* can be formed by integrating the continuous function agains a compact basis set represented by tetrahedra. These methods work will with the irregular geometries that arise in engineering. Finite-difference and finite-volume methods can both be applied to *structured grids* (each zone can be referenced simply by an integer index for each spatial dimension). The main difference between these methods is that the finite-difference methods build from the differential form of PDEs while the finite-volume methods build from the integral form of the PDEs. Both of these methods find wide use in astrophysics.

In these notes, we will focus on grid-based methods.

2 Grid basics

The grid is the fundamental object for representing continous functions in a discretized fashion, making them amenable to computation. In astrophysics, we typically use structured grids—these are logically Cartesian, meaning that the position of a quantity on the grid can be specified by a single integer index in each dimension.

We represent derivatives numerically by discretizing the domain into a finite number of zones (a numerical grid). This converts a continuous derivative into a difference of discrete data. Different approximations have different levels of accuracy.

There are two main types of structured grids used in astrophysics: *finite-difference* and *finite-volume*. These differ in way the data is represented. On a finite-difference grid, the discrete data is associated with a specific point in space. On a finite-volume grid, the discrete data is represented by averages over a control volume. Nevertheless, these methods can often lead to very similar looking discrete equations.

Consider the set of grids show in Figure 2. On the top is a class finite-difference grid. The discrete data, f_i, are stored as points regularly spaced in x. With this discretization, the spatial locations of the points are simply $x_i = i\Delta x$, where $i = 0, \ldots, N-1$. Note that for a finite-sized domain, we would put a grid point precisely on the physical boundary at each end.

The middle grid is also finite-difference, but now we imagine first dividing the domain into N cells or zones, and we store the discrete data, f_i, at the center of the zone. This is often called a *cell-centered finite-difference* grid. The physical coordinate of the zone centers (where the data lives) are: $x_i = (i + 1/2)\Delta x$, where $i = 0, \ldots N-1$. Note that now for a finite-sized domain, the left edge of the first cell will be on the boundary and the first data value will be associated at a point $\Delta x/2$ inside the boundary. A similar situation arises at the right physical boundary.

Finally, the bottom grid is a finite-volume grid. The layout looks identical to the cell-centered finite difference grid, except now instead of the discrete data being associated at a single point in space, we draw a line to show that it is taken as the average in a zone. We label it as $\langle f \rangle_i$ to indicate the average. We label the left and right edges of a zone with half-integer indices $i - 1/2$ and $i + 1/2$. The physical coordinate of the center of the zone is the same as in the cell-centered finite-difference case.

In all cases, for a *regular* structured grid, we take Δx to be constant. For the finite-difference grids, the discrete value at each point is obtains from the continuous function $f(x)$ as:

$$f_i = f(x_i) \tag{1}$$

For the finite-volume discretization, we compute the average value of the function $f(x)$ in each cell as:

$$\langle f \rangle_i = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} f(x) dx \tag{2}$$

3 Differences and order of accuracy

We can express the first derivative of a quantity a at the cell-center i as:

$$\left.\frac{\partial a}{\partial x}\right|_i \approx \frac{a_i - a_{i-1}}{\Delta x} \tag{3}$$

or

$$\left.\frac{\partial a}{\partial x}\right|_i \approx \frac{a_{i+1} - a_i}{\Delta x} \tag{4}$$

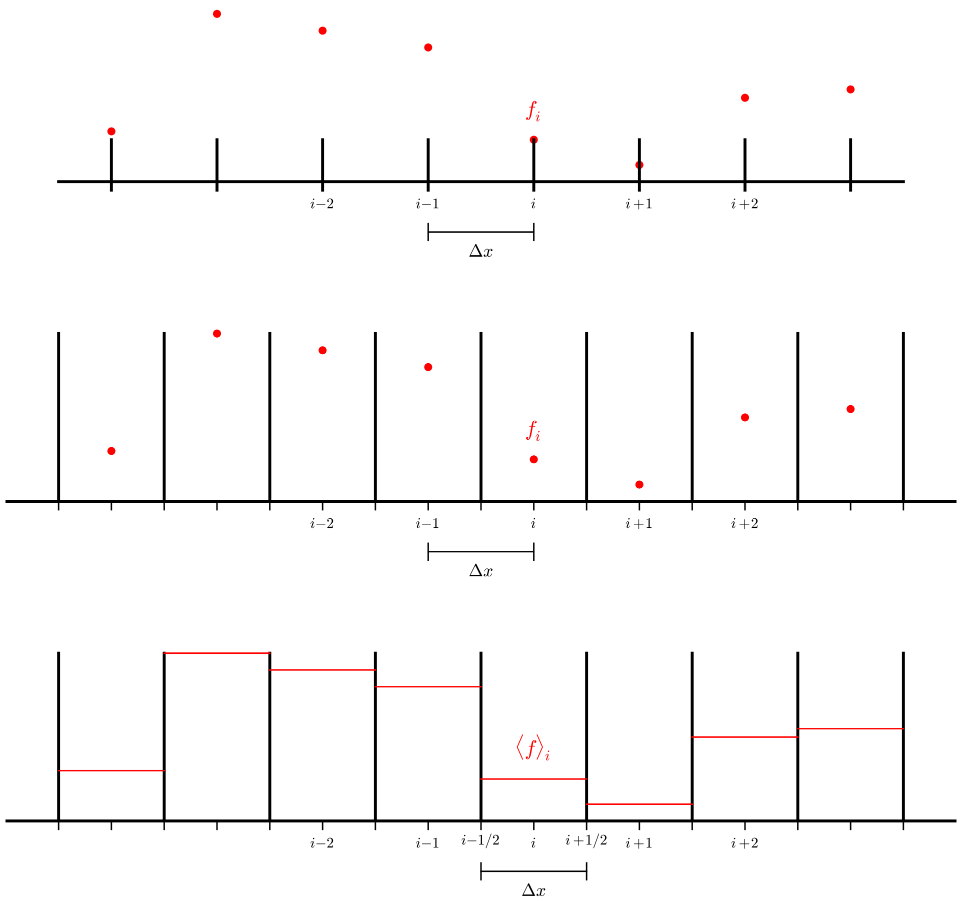

Figure 2: Different types of structured grids. Top: a finite-difference grid—the discrete data are associated with a specific point in space. Middle: a cell-centered finite-difference grid—again the data is at a specific point, but now we imagine the domain divided into zone with the data living at the center of each zone. Bottom: a finite-volume grid—here the domain is divided into zones and we store the average value of the function within each zone.

(Indeed, as $\Delta x \rightarrow 0$, this is the definition of a derivative from calculus.) Both of these are *one-sided differences*. By Taylor expanding the data about x_i, we see

$$a_{i+1} = a_i + \Delta x \left.\frac{\partial a}{\partial x}\right|_i + \frac{1}{2}\Delta x^2 \left.\frac{\partial^2 a}{\partial x^2}\right|_i + \ldots \tag{5}$$

Solving for $\partial a/\partial x|_i$, we see

$$\left.\frac{\partial a}{\partial x}\right|_i = \frac{a_i - a_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \tag{6}$$

where $\mathcal{O}(\Delta x)$ indicates that the order of accuracy of this approximation is $\sim \Delta x$. We say that this is *first order accurate*. This means that we are neglecting terms that scale as Δx. This is our *truncation error*—the error that arises because our numerical approximation throws away higher order terms. The approximation $\partial a/\partial x|_i = (a_{i+1} - a_i)/\Delta x$ has the same order of accuracy.

Exercise 1: Show that a centered difference, $\partial a/\partial x|_i = (a_{i+1} - a_{i-1})/(2\Delta x)$, is second order accurate, i.e. its truncation error is $\mathcal{O}(\Delta x^2)$.

Generally speaking, higher-order methods have lower numerical error associated with them.

Notice that the righthand side of all these approximations involve some linear combinations of a_i's. We call this the *stencil*. The *width* of the stencil is a measure of how many zones on either side of zone i we reach when computing our approximation.

4 Finite-volume grids

In the finite-volume discretization, a_i represents the average of $a(x,t)$ over the interval $x_{i-1/2}$ to $x_{i+1/2}$, where the half-integer indexes denote the zone edges (i.e. $x_{i-1/2} = x_i - \Delta x/2$):

$$a_i = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} a(x) dx \tag{7}$$

The figure below shows the grid, with the half-integer indices. Here we've drawn a_i as a horizontal line

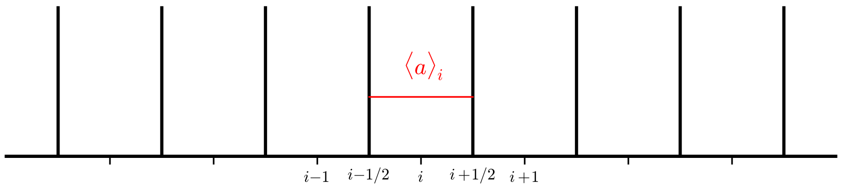

Figure 3: A simple 1-d finite-volume grid with the zone-centered and zone-edge indexes. Here we show only the data associated with zone i.

spanning the entire zone—this is to represent that it is an average within the volume defined by the zone edges. To second-order accuracy,

$$a_i = \frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} a(x) dx \sim a(x_i) \tag{8}$$

This means that we can treat the average of a over a zone as simply $a(x)$ evaluated at the zone center if we only want second-order accuracy.

Exercise 2: Show that Eq. 8 is true to $O(\Delta x^2)$ by expanding $a(x)$ as a Taylor series in the integral.

4.1 Conservation

The finite-volume grid is useful when dealing with conservation laws. Consider the following system:

$$\frac{\partial U}{\partial t} + \nabla \cdot F(U) = 0 \tag{9}$$

where U is a vector of conserved quantities and $F(U)$ is the flux of each quantity. This type of system arises, for example, in fluid flow, where the system will represent conservation of mass, momentum, and energy.

Consider one-dimension. Integrating this system over a zone, and normalizing by Δx, we have:

$$\frac{\partial U_i}{\partial t} = -\frac{1}{\Delta x} \int_{x_{i-1/2}}^{x_{i+1/2}} \frac{\partial F}{\partial x} dx = -\frac{1}{\Delta x} \left\{ F(U)|_{x_{i+1/2}} - F(U)|_{x_{i-1/2}} \right\} \tag{10}$$

Independent of how we discretize in time, notice that the righthand side is simply a difference of fluxes through the interfaces of the zone. For the $i+1$ zone, the update would be:

$$\frac{\partial U_{i+1}}{\partial t} = -\frac{1}{\Delta x} \left\{ F(U)|_{x_{i+3/2}} - F(U)|_{x_{i+1/2}} \right\} \tag{11}$$

Notice that this shares the flux at the $x_{i+1/2}$ interface, but with the opposite sign. When all of the updates are done, the flux through each boundary adds to one zone and subtracts from its neighbor, exactly conserving (to round-off error) the quantity U. This cancellation of the sums is an example of a *telescoping property*, and is the main reason why finite-volume methods are attractive—conserved quantities are conserved.

4.2 Boundary conditions with finite-volume grids

Imagine that we wish to compute the derivative in every zone using:

$$\left.\frac{\partial a}{\partial x}\right|_i = \frac{a_i - a_{i-1}}{\Delta x} \quad . \tag{12}$$

If we denote the index corresponding to the leftmost zone in our domain as 'lo', then when we try to compute $\partial a/\partial x|_{\mathrm{lo}}$, we will "fall-off" the grid, i.e., we need a value of a for zone $\mathrm{lo} - 1$, which is outside the domain. This is where boundary conditions for our grid come into play.

We implement boundary conditions by extending the computational grid beyond the physical domain. These additional zones are called *ghost cells*. The exist solely to handle the boundary conditions and allow us to use the same update equation (e.g. Eq. 12) for all zones in the domain.

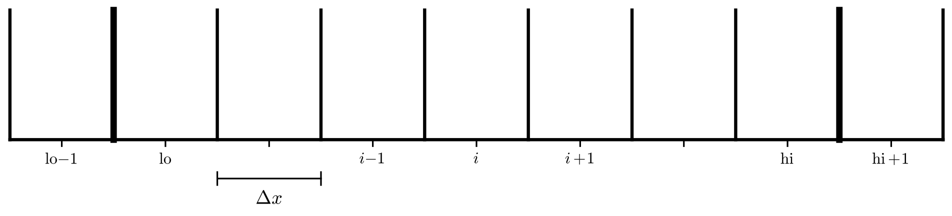

Figure 4: A simple 1-d finite-volume grid with a single ghost cell at each end of the domain. The domain boundaries are indicated by the heavy vertical lines. Here there are $\mathrm{hi} - \mathrm{lo} + 1$ zones used in the discretization of the domain, with the first zone in the domain labelled lo and the last in the domain labelled hi.

The number of ghostcells needed for the grid depends on how wide the stencils used in the computation are. The wider the stencil, the more ghostcells are needed.

Periodic boundary conditions would be implemented as:

$$a_{\mathrm{hi}+1} = a_{\mathrm{lo}} \tag{13}$$
$$a_{\mathrm{lo}-1} = a_{\mathrm{hi}} \tag{14}$$

5 Going further

- *Domain decomposition*: when running on a parallel computer, the work is divided up across processor using domain decomposition. Here, we break the computational domain into smaller sub-domains, and put one (or more) sub-domains on each processor. Each sub-domain has its own perimeter of ghost cells. These are now filled by copying information from the neighboring sub-domains or using the physical boundary conditions for the full domain, depending on where the ghost cells lie. Figure 5 shows a simple decomposition of a domain into 6 sub-domains.

- *AMR for structured grids*: adaptive mesh refinement uses a hierarchy of grids to focus resolution in regions of complex flow. For finite-volume codes, the standard reference for AMR is Berger &

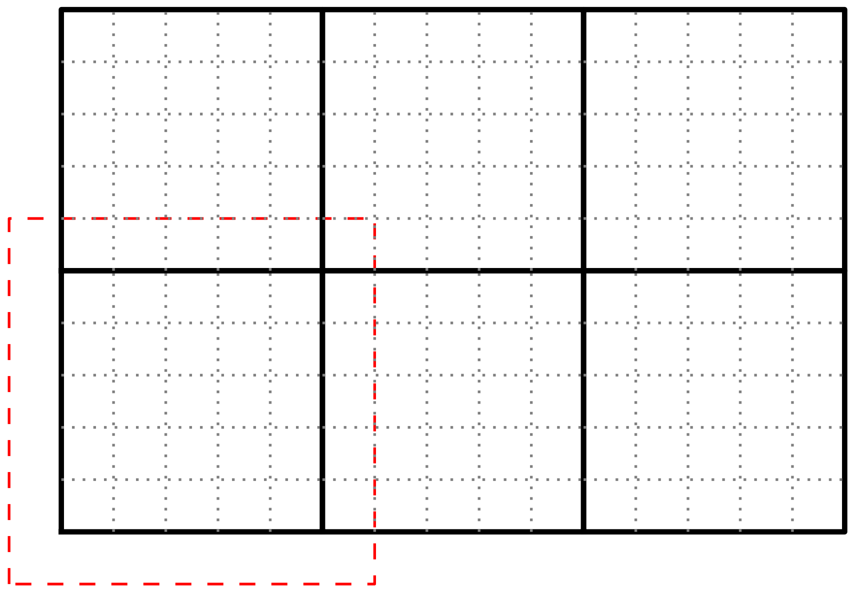

Figure 5: Domain decomposition of the computational domain into 6 separate sub-domains. Each sub-domain here has 5×5 zones. For one of the sub-domains, the perimeter of ghost cells is illustrated as the red boundary.

Colella [1]. Each level is an even integer multiple finer in resolution, and the grid cells line up with one another (i.e. in two-dimensions, four fine cells will be completely enclosed by a single coarse cell, when using a jump in resolution of $2\times$.) This provides a natural way to enforce conservation. At coarse-fine interfaces, corrections are done to ensure consistency.

References

[1] M. J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. 82(1):64–84, May 1989.