Lax–Friedrichs method; CFL condition; finite-volume methods.

*** We will use the LaxFriedrichs fluxes evaluated in WENO (Weighted Essentially Non-oscillatory) method**

The Lax-Friedrichs Sweeping Schemes in One Dimension

1 One-dimensional scalar problems

we consider the one-dimensional scalar steady state problem

$$f(u)_x = s(u,x), \quad x \in [a,b] \tag{1}$$

subject to an initial guess and appropriate boundary conditions.

To obtain a numerical scheme, the interval is first discretized uniformly into N cells, and the grid points are denoted by $\{x_j\}_{j=0}^{N}$, where $x_j = a + j\Delta x, j = 0\cdots, N$ and $\Delta x = (b-a)/N$. The midpoint of a cell is defined as $x_{j+\frac{1}{2}} = (x_j + x_{j+1})/2, j = 0, \cdots, N-1$. The numerical approximation of u on the grid points x_j are denoted by $u_j, j = 0, \cdots, N$. A conservative finite difference type

discretization of Eq. (1) can be written as

$$\frac{\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}}{\Delta x} = s(u_j, x_j), \tag{2}$$

in which $\hat{f}_{j\pm\frac{1}{2}}$ represent numerical fluxes approximating the fluxes at $x_{j\pm\frac{1}{2}}$. The order of the numerical scheme thus depends on the order that $\frac{\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}}{\Delta x}$ approximates $f_x(u_j)$.

Our goal is to design a Gauss-Seidel type iterative scheme, based on Eq. (2), which can accelerate the speed to compute steady state solutions of hyperbolic equations.

1.1 A framework: a first order Lax-Friedrichs sweeping method

We substitute the numerical fluxes in Eq. (2) by the first order Lax-Friedrichs fluxes

$$\hat{f}_{j+\frac{1}{2}} = \frac{1}{2}(f(u_j) + f(u_{j+1})) - \frac{\sigma}{2}(u_{j+1} - u_j) \tag{3}$$

where $\sigma = \max_j\{|f'(u_j)|\}, j = 1, \cdots, N$. This leads to the discretization formula

$$\frac{1}{2}(f(u_{j+1}) - f(u_{j-1})) - \frac{\sigma}{2}(u_{j+1} - 2u_j + u_{j-1}) - \Delta x s(u_j, x_j) = 0,$$

and equivalently,

$$u_j = \frac{1}{2}(u_{j-1} + u_{j+1}) + \frac{1}{\sigma}(\Delta x s(u_j, x_j) - \frac{1}{2}(f(u_{j+1}) - f(u_{j-1}))).$$

If we denote the iteration step by n, with $n = 0$ corresponding to the initial guess, a simple update formula would be

$$u_j^{n+1} = \frac{1}{2}(u_{j-1} + u_{j+1}) + \frac{1}{\sigma}(\Delta x s(u_j^n, x_j) - \frac{1}{2}(f(u_{j+1}) - f(u_{j-1}))). \tag{4}$$

The superscripts of u_{j+1} and u_{j-1} on the right-hand-side of Eq. (4) are not specified because they depend on whether the numerical solutions are updated from left to right or from right to left, which we call the sweeping direction.

If the sweeping direction is from left to right, then we take $u_{j-1} = u_{j-1}^{n+1}$ and $u_{j+1} = u_{j+1}^n$. The formula becomes

$$u_j^{n+1} = \frac{1}{2}(u_{j-1}^{n+1} + u_{j+1}^n) + \frac{1}{\sigma}(\Delta x s(u_j^n, x_j) - \frac{1}{2}(f(u_{j+1}^n) - f(u_{j-1}^{n+1}))).$$

If the sweeping direction is from right to left, then $u_{j-1} = u_{j-1}^n$ and $u_{j+1} = u_{j+1}^{n+1}$ will be used, and

$$u_j^{n+1} = \frac{1}{2}(u_{j-1}^n + u_{j+1}^{n+1}) + \frac{1}{\sigma}(\Delta x s(u_j^n, x_j) - \frac{1}{2}(f(u_{j+1}^{n+1}) - f(u_{j-1}^n))).$$

These updating formulas are essentially Gauss-Seidel iterations because the point values are computed using newly updated neighboring values. Our sweeping method is simply to update point values by Eq. (4), with alternating sweeping directions to achieve a faster convergence rate.

1.2 A review of construction of high order finite difference WENO fluxes

To develop a higher order sweeping method, one needs high order numerical fluxes in Eq. (2). In this section, we briefly review the construction of numerical fluxes for high order finite difference WENO schemes, and the same construction will be used in our scheme, presented in the next section.

Consider a scalar hyperbolic conservation law in one dimension

$$u_t + f(u)_x = 0$$

with a positive wind direction $f'(u) \geq 0$. The spatial derivative is approximated by a conservative flux difference

$$f(u)_x|_{x=x_j} \approx \frac{1}{\Delta x}\left(\hat{f}_{j+\frac{1}{2}} - \hat{f}_{j-\frac{1}{2}}\right)$$

when a uniform discretization with mesh size Δx is used. The numerical flux $\hat{f}_{j+\frac{1}{2}}$ is computed through the neighboring point values $f_i = f(u_i)$. For a $(2k-1)$th order WENO scheme, we first compute k numerical fluxes

$$\hat{f}^{(r)}_{j+\frac{1}{2}} = \sum_{i=0}^{k-1} c_{ri} f_{j-r+i}, \quad r = 0, \cdots, k-1,$$

corresponding to k different candidate stencils $S_r(j) = \{x_{j-r}, \cdots, x_{j-r+k-1}\}$, $r = 0, \cdots, k-1$. Each of these numerical fluxes is kth order accurate. For example, when $k = 2$ (third order WENO scheme), the two second order accurate numerical fluxes are given by

$$\begin{aligned} \hat{f}^{(0)}_{j+\frac{1}{2}} &= \frac{1}{2} f_j + \frac{1}{2} f_{j+1}, \\ \hat{f}^{(1)}_{j+\frac{1}{2}} &= -\frac{1}{2} f_{j-1} + \frac{3}{2} f_j. \end{aligned}$$

The $(2k-1)$th order WENO flux is a convex combination of all these k numerical fluxes

$$\hat{f}_{j+\frac{1}{2}} = \sum_{r=0}^{k-1} w_r \hat{f}^{(r)}_{j+\frac{1}{2}}.$$

The nonlinear weights w_r satisfy $w_r \geq 0$, $\sum_{r=0}^{k-1} w_r = 1$, and are defined in the following way

$$w_r = \frac{\alpha_r}{\sum_{s=0}^{k-1} \alpha_s}, \quad \alpha_r = \frac{d_r}{(\epsilon + \beta_r)^2}.$$

Here d_r are the linear weights which yield $(2k-1)$th order accuracy, β_r are the so-called "smoothness indicators" of the stencils $S_r(j)$, which measure the smoothness of the function $f(u(x))$ in the stencils. The constant ϵ is a small number used to avoid the division by zero and is typically taken as 10^{-6}. For example, when $k = 2$ (third order WENO scheme), the linear weights are given

by

$$d_0 = \frac{2}{3}, \quad d_1 = \frac{1}{3},$$

and the smoothness indicators are given by

$$\beta_0 = (f_{j+1} - f_j)^2, \quad \beta_1 = (f_j - f_{j-1})^2.$$

The procedure for the case with $f'(u) \leq 0$ is mirror symmetric with respect to $j + \frac{1}{2}$.

The Lax-Friedrichs numerical flux is constructed through splitting the fluxes into positive and negative fluxes

$$f^{\pm}(u) = \frac{1}{2}(f(u) \pm \alpha u),$$

where α is taken as $\alpha = \max_u |f'(u)|$. The WENO procedure is applied to $f^{\pm}$ individually with upwind biased stencils. Depending on whether the maximum is taken globally or locally, such schemes are referred to as the Lax-Friedrichs WENO scheme or the local Lax-Friedrichs WENO scheme.

1.3 High order Lax-Friedrichs WENO sweeping method

As introduced in Section 1.2, high order numerical fluxes $\hat{f}_{j\pm\frac{1}{2}}$ can be constructed through WENO procedure, and here we use Lax-Friedrichs WENO fluxes to obtain the high order sweeping method.

First, we define

$$\hat{\hat{f}}_{j+\frac{1}{2}} = \hat{f}_{j+\frac{1}{2}} + \frac{\sigma}{2}(u_{j+1} - u_j), \quad j = 0, \cdots, N-1 \tag{5}$$

where $\hat{f}_{j+\frac{1}{2}}$ is the original high order Lax-Friedrichs flux. The newly defined flux retains the form of Eq. (3) by subtracting the diffusion term explicitly,

which allows us to formulate the iterative scheme. The discretization formula then could be written, in terms of $\hat{\hat{f}}_{j+\frac{1}{2}}$, as

$$\frac{(\hat{\hat{f}}_{j+\frac{1}{2}} - \frac{\sigma}{2}(u_{j+1} - u_j)) - (\hat{\hat{f}}_{j-\frac{1}{2}} - \frac{\sigma}{2}(u_j - u_{j-1}))}{\Delta x} = s(u_j, x_j). \tag{6}$$

Thus, we obtain the iterative scheme

$$u_j^{n+1} = \frac{1}{2}(u_{j-1} + u_{j+1}) + \frac{1}{\sigma}\left(\Delta x s(u_j^n, x_j) - (\hat{\hat{f}}_{j+\frac{1}{2}} - \hat{\hat{f}}_{j-\frac{1}{2}})\right). \tag{7}$$

As in the first order sweeping method, the iterations will take alternating directions: If we sweep from left to right, then $u_{j-1} = u_{j-1}^{n+1}$ and $u_{j+1} = u_{j+1}^{n}$ are used; if we sweep from right to left, $u_{j-1} = u_{j-1}^{n}$ and $u_{j+1} = u_{j+1}^{n+1}$ are used.

Upon the convergence of the iterations, high order accuracy will be achieved due to the use of high order numerical fluxes. The WENO construction of the numerical fluxes guarantees essentially non-oscillatory properties of the numerical solutions around discontinuities.

1.4 Convergence of the Lax-Friedrichs fast sweeping method

Generally, it is very difficult to study the convergence properties of the high order sweeping methods proposed in Section 1.3 due to its nonlinear nature. The analysis of the first order linear Lax-Friedrichs sweeping method for nonlinear hyperbolic equations is also limited because the numerical operators are also fully nonlinear. To gain some insights of the convergence properties with respect to different choices of σ, we perform the analysis on the simplest linear steady state equation:

$$u_x = 1, \quad x \in [0, 1], \tag{8}$$

with the boundary condition $u(0) = 0$. Although at $x = 1$, one does not need a boundary condition, numerical boundary conditions are still required, and here we use linear extrapolation at the right boundary.

Let us denote by superscript $+$ the sweep from left to right, and $-$ the sweep from right to left. We also denote the solution by a vector notation $\mathbf{u}^n = (u_1^n, \cdots, u_N^n)^T$ with superscript n representing the number of iterations. If we further denote $\frac{1}{2} - \frac{1}{2\sigma}$ by a and $\frac{1}{2} + \frac{1}{2\sigma}$ by b, then after two alternating iterations the sweeping method can be formulated by

$$A_+(\mathbf{u}^{n+1})^+ = B_+(\mathbf{u}^n)^- + C, \tag{9}$$

and

$$A_-(\mathbf{u}^{n+2})^- = B_-(\mathbf{u}^{n+1})^+ + C, \tag{10}$$

where

$$A_+ = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ -b & 1 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & -b & 1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & -b & 1 & 0 & \cdot \\ \cdot & \cdot & \cdot & 0 & -b & 1 & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & a-b & 1-2a \end{bmatrix},$$

$$B_+ = \begin{bmatrix} 0 & a & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & a & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 0 & a & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 0 & a & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & 0 & a \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & 0 \end{bmatrix},$$

$$A_- = \begin{bmatrix} 1 & -a & \cdot & \cdot & \cdot & \cdot & 0 \\ 0 & 1 & -a & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 1 & -a & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & 1 & -a & 0 \\ \cdot & \cdot & \cdot & \cdot & 0 & 1 & -a \\ 0 & \cdot & \cdot & \cdot & 0 & 0 & 1-2a \end{bmatrix},$$

$$B_- = \begin{bmatrix} 0 & 0 & \cdot & \cdot & \cdot & \cdot & 0 \\ b & 0 & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & b & 0 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & b & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & \cdot & b & 0 & 0 \\ 0 & \cdot & \cdot & \cdot & \cdot & b-a & 0 \end{bmatrix},$$

and $C = (\frac{\Delta x}{\sigma}, \frac{\Delta x}{\sigma}, \cdots, \frac{\Delta x}{\sigma})^T$.

Thus, the update formula becomes

$$(\mathbf{u}^{n+1})^+ = (A_+)^{-1} B_+ (\mathbf{u}^n)^- + (A_+)^{-1} C \tag{11}$$

and

$$(\mathbf{u}^{n+2})^- = (A_-)^{-1} B_- (\mathbf{u}^{n+1})^+ + (A_-)^{-1} C, \tag{12}$$

Combining the above two formulas, one obtains after one complete sweeping:

$$(\mathbf{u}^{n+2})^- = (A_-)^{-1} B_- (A_+)^{-1} B_+ (\mathbf{u}^n)^- + (A_-)^{-1} B_- (A_+)^{-1} C + (A_-)^{-1} C. \tag{13}$$

Convergence of the iterative scheme (13) is guaranteed for any fixed N if the spectral radius $\rho\left((A_-)^{-1} B_- (A_+)^{-1} B_+\right)$ is strictly less than 1. Since it is difficult to analyze theoretically the spectral radius of the iteration matrix , we provide the numerical evaluation of the spectral radius for $N = 100$ and $N = 200$, with the value of σ varied from 1 to 20 with increment 0.1. It can be seen from Fig. 1 that the spectral radius is always less than 1, but increasingly approaches 1 as σ becomes larger, and it decreases rapidly as σ is close to 1. Notice that in the scheme we implemented numerically the optimal choice $\sigma = 1$ as showed in Eq.(3) to have rapid convergence. Choice of larger σ may result in more iterations.

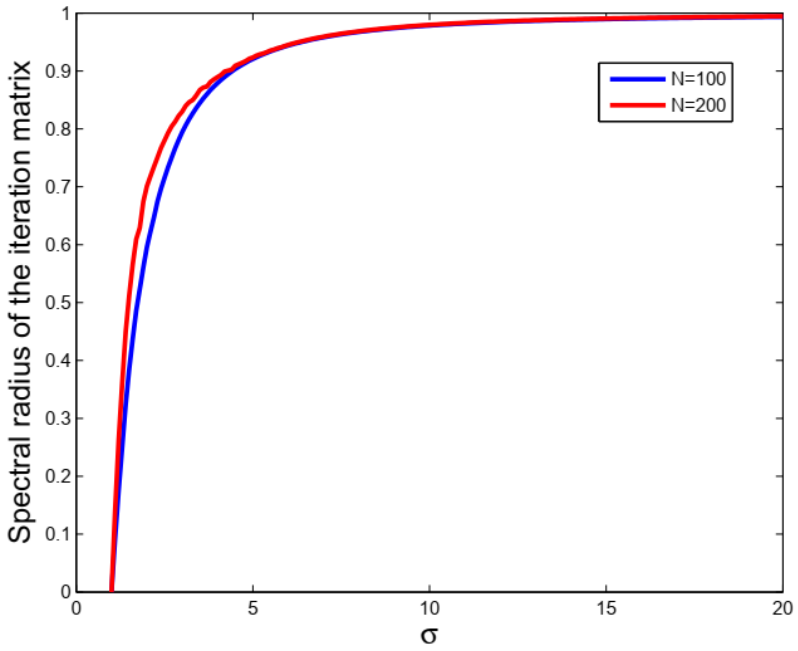

Figure 1: Spectral radius of the iteration matrix in Eq. (13), with σ varied from 1 to 20 and incremented by 0.1.

2 One-dimensional system problems

For one-dimensional systems

$$\mathbf{f}(\mathbf{u})_x = \mathbf{s}(\mathbf{u}, x), \quad x \in [a, b]$$

where $\mathbf{f}, \mathbf{s}$ and $\mathbf{u}$ are vector-valued functions in R^m, we use the same iterative formula but perform local characteristic decomposition for numerical flux construction, which is more robust than the component-by-component evaluation. The flux reconstruction procedure is described as follows. First we compute an average state $\mathbf{u}_{j+\frac{1}{2}}$ between $\mathbf{u}_j$ and $\mathbf{u}_{j+1}$, using either the simple arithmetic mean or a Roe's average . The right eigenvectors $\mathbf{r}_m$ and the left eigenvectors $\mathbf{l}_m$ of the Jacobian $\mathbf{f}'(\mathbf{u}_{j+\frac{1}{2}})$ are needed for the local characteristic decomposition. The WENO procedure is used on

$$\mathbf{v}_i^{\pm} = \mathbf{R}^{-1}\mathbf{f}_i^{\pm}, \quad \text{for } i \text{ in a neighborhood of } j,$$

where $\mathbf{R} = (\mathbf{r}_1, \cdots, \mathbf{r}_n)$ is the matrix whose columns are the right eigenvectors of $\mathbf{f}'(\mathbf{u}_{j+\frac{1}{2}})$. The numerical fluxes $\hat{\mathbf{v}}^{\pm}_{j+\frac{1}{2}}$ thus computed are then projected back

into the physical space by left multiplying with $\mathbf{R}$, yielding finally the numerical fluxes $\hat{\mathbf{f}}_{j\pm\frac{1}{2}}$ in the physical space. Finally, the grid values $\mathbf{u}_j$ are updated component-wisely by formula (7).

Courant–Friedrichs–Lewy condition (CFL Condition)

In mathematics, the **convergence condition by Courant–Friedrichs–Lewy** is a necessary condition for convergence while solving certain partial differential equations (usually hyperbolic PDEs) numerically. It arises in the numerical analysis of explicit time integration schemes, when these are used for the numerical solution. As a consequence, the time step must be less than a certain time in many explicit time-marching computer simulations, otherwise the simulation produces incorrect results. The condition is named after Richard Courant, Kurt Friedrichs, and Hans Lewy who described it in their 1928 paper.

Heuristic description

The principle behind the condition is that, for example, if a wave is moving across a discrete spatial grid and we want to compute its amplitude at discrete time steps of equal duration, then this duration must be less than the time for the wave to travel to adjacent grid points. As a corollary, when the grid point separation is reduced, the upper limit for the time step also decreases. In essence, the numerical domain of dependence of any point in space and time (as determined by initial conditions and the parameters of the approximation scheme) must include the analytical domain of dependence (wherein the initial conditions have an effect on the exact value of the solution at that point) to assure that the scheme can access the information required to form the solution.

Statement

To make a reasonably formally precise statement of the condition, it is necessary to define the following quantities:

- *Spatial coordinate*: one of the coordinates of the physical space in which the problem is posed
- *Spatial dimension of the problem*: the number of spatial dimensions, i.e., the number of spatial coordinates of the physical space where the problem is posed. Typical values are n=1, n=2 and n=3
- *Time*: the coordinate, acting as a parameter, which describes the evolution of the system, distinct from the spatial coordinates

The spatial coordinates and the time are discrete-valued independent variables, which are placed at regular distances called the *interval length* and the *time step*, respectively. Using these names, the CFL condition relates the length of the time step to a function of the interval lengths of each spatial coordinate and of the maximum speed that information can travel in the physical space. Operatively, the CFL condition is commonly prescribed for those terms of the finite-difference approximation of general partial differential equations that model the advection phenomenon.

The one-dimensional case

- For the one-dimensional case, the continuous-time model equation (that is usually solved for w) is

$$\frac{\partial w}{\partial t} = u\frac{\partial w}{\partial x}.$$

The CFL condition then has the following form:

$$C = \frac{u\,\Delta t}{\Delta x} \leq C_{\max}$$

- where the dimensionless number C is called the courant number, u is the magnitude of the velocity (whose dimension is length/time) Δt is the time step (whose dimension is time) Δx is the length interval (whose dimension is length).

The value of $C_{\max}$ changes with the method used to solve the discretised equation, especially depending on whether the method is explicit or implicit. If an explicit (time-marching) solver is used then typically $C_{\max}$ =1. Implicit (matrix) solvers are usually less sensitive to numerical instability and so larger values of $C_{\max}$ may be tolerated.

The two and general *n*-dimensional case

In the two-dimensional case, the CFL condition becomes

$$C = \frac{u_x\,\Delta t}{\Delta x} + \frac{u_y\,\Delta t}{\Delta y} \leq C_{\max}$$

with the obvious meanings of the symbols involved. By analogy with the two-dimensional case, the general CFL condition for the n dimensional case is the following one:

$$C = \Delta t \left(\sum_{i=1}^{n} \frac{u_{x_i}}{\Delta x_i}\right) \leq C_{\max}.$$

The interval length is not required to be the same for each spatial variable $\Delta x_i, i = 1, \ldots, n.$ This "degree of freedom" can be used to somewhat optimize the value of the time step for a particular problem, by varying the values of the different interval to keep it not too small.

Finite volume method

The **finite volume method** (**FVM**) is a method for representing and evaluating partial differential equations in the form of algebraic equations. In the finite volume method, volume integrals in a partial differential equation that contain a divergence term are converted to

surface integrals, using the divergence theorem. These terms are then evaluated as fluxes at the surfaces of each finite volume. Because the flux entering a given volume is identical to that leaving the adjacent volume, these methods are conservative. Another advantage of the finite volume method is that it is easily formulated to allow for unstructured meshes. The method is used in many computational fluid dynamics packages. "Finite volume" refers to the small volume surrounding each node point on a mesh.

Finite volume methods can be compared and contrasted with the finite difference methods, which approximate derivatives using nodal values, or finite element methods, which create local approximations of a solution using local data, and construct a global approximation by stitching them together. In contrast a finite volume method evaluates exact expressions for the *average* value of the solution over some volume, and uses this data to construct approximations of the solution within cells.

Example

Consider a simple 1D advection problem:

$$\frac{\partial \rho}{\partial t} + \frac{\partial f}{\partial x} = 0, \quad t \geq 0. \tag{1}$$

Here, $\rho = \rho(x,t)$ represents the state variable and $f = f(\rho(x,t))$ represents the flux or flow of ρ. Conventionally, positive f represents flow to the right while negative f represents flow to the left. If we assume that equation (1) represents a flowing medium of constant area, we can sub-divide the spatial domain, x, into *finite volumes* or *cells* with cell centers indexed as i.

For a particular cell, i, we can define the *volume average* value of $\rho_i(t) = \rho(x,t)$ at time $t = t_1$ and $x \in \left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$, as

$$\bar{\rho}_i(t_1) = \frac{1}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \rho(x,t_1)\, dx, \tag{2}$$

and at time $t = t_2$ as,

$$\bar{\rho}_i(t_2) = \frac{1}{x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \rho(x,t_2)\, dx, \tag{3}$$

where $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$ represent locations of the upstream and downstream faces or edges respectively of the i^{th} cell.

Integrating equation (1) in time, we have:

$$\rho(x,t_2) = \rho(x,t_1) - \int_{t_1}^{t_2} f_x(x,t)\, dt, \tag{4}$$

where $f_x = \frac{\partial f}{\partial x}$.

To obtain the volume average of $\rho(x,t)$ at time $t = t_2$, we integrate $\rho(x,t_2)$ over the cell volume, $\left[x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right]$ and divide the result by $\Delta x_i = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$, i.e.

$$\bar{\rho}_i(t_2) = \frac{1}{\Delta x_i} \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left\{ \rho(x,t_1) - \int_{t_1}^{t_2} f_x(x,t)\, dt \right\} dx. \tag{5}$$

We assume that f is well behaved and that we can reverse the order of integration. Also, recall that flow is normal to the unit area of the cell. Now, since in one dimension $f_x \triangleq \nabla \cdot f$, we can apply the divergence theorem, i.e. $\oint_v \nabla \cdot f dv = \oint_S f \, dS$, and substitute for the volume integral of the divergence with the values of $f(x)$ evaluated at the cell surface (edges $x_{i-\frac{1}{2}}$ and $x_{i+\frac{1}{2}}$) of the finite volume as follows:

$$\bar{\rho}_i\left(t_2\right) = \bar{\rho}_i\left(t_1\right) - \frac{1}{\Delta x_i}\left(\int_{t_1}^{t_2} f_{i+\frac{1}{2}}\, dt - \int_{t_1}^{t_2} f_{i-\frac{1}{2}}\, dt\right). \tag{6}$$

where $f_{i\pm\frac{1}{2}} = f\left(x_{i\pm\frac{1}{2}}, t\right)$.

We can therefore derive a *semi-discrete* numerical scheme for the above problem with cell centers indexed as i, and with cell edge fluxes indexed as $i \pm \frac{1}{2}$, by differentiating (6) with respect to time to obtain:

$$\frac{d\bar{\rho}_i}{dt} + \frac{1}{\Delta x_i}\left[f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}\right] = 0, \tag{7}$$

where values for the edge fluxes, $f_{i\pm\frac{1}{2}}$, can be reconstructed by interpolation or extrapolation of the cell averages. Equation (7) is *exact* for the volume averages; i.e., no approximations have been made during its derivation.

This method can also be applied to a 2D situation by considering the north and south faces along with the east and west faces around a node.

General conservation law

We can also consider the general conservation law problem, represented by the following PDE,

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot \mathbf{f}\left(\mathbf{u}\right) = \mathbf{0}. \tag{8}$$

Here, $\mathbf{u}$ represents a vector of states and $\mathbf{f}$ represents the corresponding flux tensor. Again we can sub-divide the spatial domain into finite volumes or cells. For a particular cell, i, we take the volume integral over the total volume of the cell, v_i, which gives,

$$\int_{v_i} \frac{\partial \mathbf{u}}{\partial t}\, dv + \int_{v_i} \nabla \cdot \mathbf{f}\left(\mathbf{u}\right)\, dv = \mathbf{0}. \tag{9}$$

On integrating the first term to get the *volume average* and applying the *divergence theorem* to the second, this yields

$$v_i \frac{d\bar{\mathbf{u}}_i}{dt} + \oint_{S_i} \mathbf{f}\left(\mathbf{u}\right) \cdot \mathbf{n}\, dS = \mathbf{0}, \tag{10}$$

where S_i represents the total surface area of the cell and $\mathbf{n}$ is a unit vector normal to the surface and pointing outward. So, finally, we are able to present the general result equivalent to (8), i.e.

$$\frac{d\bar{\mathbf{u}}_i}{dt} + \frac{1}{v_i}\oint_{S_i} \mathbf{f}\left(\mathbf{u}\right) \cdot \mathbf{n}\, dS = \mathbf{0}. \tag{11}$$

Again, values for the edge fluxes can be reconstructed by interpolation or extrapolation of the cell averages. The actual numerical scheme will depend upon problem geometry and mesh construction. MUSCL reconstruction is often used in high resolution schemes where shocks or discontinuities are present in the solution.

Finite volume schemes are conservative as cell averages change through the edge fluxes. In other words, *one cell's loss is another cell's gain*!

References and further readings

1 A. R. Conn, N. I. M. Gould and Ph. L. Toint, Trust-Region Methods, SIAM 2000.

2. J. Dennis and R. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear equations, (republished by) SIAM, 1996.

3. R. Fletcher, Practical Methods of Optimization, 2nd edition Wiley, 1987 (republished in 2000).

4. P. Gill, W. Murray and M. H. Wright, Practical Optimization, Academic Press, 1981.

5. N. I. M. Gould, An Introduction to Algorithms for Continuous Optimization, 2006. Available for download at http://www.numerical.rl.ac.uk/nimg/course/lectures/paper/paper.pdf.

6. J. Nocedal and S. J. Wright, Numerical Optimization, Springer Verlag, 1999 (1st edition) or 2006 (2nd edition).

7. Mathematical Modeling and Computation of Real-Time Problems: An Interdisciplinary Approach (Mathematical Engineering, Manufacturing, and Management Sciences) 1st Edition by Rakhee Kulshrestha, Chandra Shekhar, Madhu Jain, Srinivas R. Chakravarthy