CS205b/CME306

Lecture 9

1 Convection

Supplementary Reading: Osher and Fedkiw, Sections 3.3 and 3.5; Leveque, Sections 6.7, 8.3, 10.2, 10.4. For a reference on Newton polynomial interpolation via divided difference tables, see Heath, *Scientific Computing*, Section 7.3.3.

Reference: Osher and Fedkiw, Section 3.1

There are two different approaches we can take in describing fluid motion.

Lagrangian In the Lagrangian formulation, we follow the motion of individual particles as they are advected by the flow. Assuming that a velocity $\mathbf{u}(\mathbf{x}) = \langle u, v, w \rangle$ is known for every point $\mathbf{x}$ we can compute the particle motion by solving the ordinary differential equation (ODE)

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}).$$

This approach is simple, and the equation is easy to solve numerically. However, it is difficult to apply it to nonlinear phenomena such as shock waves. In general, Lagrangian formulations are most often used in solid mechanics problems.

Eulerian Recall that in the Eulerian formulation the simple convection (or advection) equation is

$$\phi_t + \mathbf{u} \cdot \nabla\phi = 0, \tag{1}$$

where ϕ is the quantity being advected. Recall that ∇ is the gradient operator, and that $\mathbf{u} \cdot \nabla\phi = u\phi_x + v\phi_y + w\phi_z$. The Eulerian approach can be thought of as sitting still at a point $\mathbf{x}$ and observing the changes in ϕ at that point due to the flow. Eulerian formulations are most often used in fluid dynamics problems.

1.1 Upwind Differencing

Reference: Osher and Fedkiw, Section 3.2

In the notes that follow we will always denote the time step in the superscript, and the spatial indices in the subscripts. Thus, ϕ_i^n is the value of ϕ at grid location i at time n.

1.1.1 Temporal Discretization

We look at some possible discretizations for the time derivatives in the convection equation. The equations below are called *semi-discrete* because we have discretized only the time derivative.

- *Forward Euler*

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \phi^n = 0 \tag{2}$$

- *Backward Euler*

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \phi^{n+1} = 0 \tag{3}$$

We call forward Euler is an explicit scheme, since it is often written with the quantity ϕ^{n+1} to be determined explicitly written in terms of known quantities. We call backward Euler an implicit scheme, since the quantity ϕ^{n+1} to be determined is expressed implicitly in terms of itself and must be solved. Both forward and backward Euler are first order accurate, meaning that the error in both discretizations is $O(\Delta t)$.

1.1.2 Spatial Discretization

We begin by writing equation 2 in expanded form

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + u^n \phi_x^n + v^n \phi_y^n + w^n \phi_z^n = 0 \tag{4}$$

and address the evaluation of the $u^n \phi_x^n$ term first. The techniques used to approximate this term can then be applied independently to the $v^n \phi_y^n$ and $w^n \phi_z^n$ terms in a dimension by dimension manner. For simplicity, consider the one dimensional version of equation 4

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + u^n \phi_x^n = 0$$

where the sign of u^n indicates whether the values of ϕ are moving to the right or to the left. Since u^n can be spatially varying, we focus on a specific grid point x_i where we write

$$\frac{\phi_i^{n+1} - \phi_i^n}{\Delta t} + u_i^n (\phi_x)_i^n = 0.$$

We use the forward Euler discretization in time. This means that we can solve the equations for each grid point independently, and we will not have to solve a linear system at each time step as in the case of backward Euler.

We first introduce the difference operators D^o, D^+ and D^-.

- D^o, *central difference* operator (second-order accurate)

$$(D^o \phi)_i = \frac{\phi_{i+1} - \phi_{i-1}}{2\Delta x}$$

- D^+, *forward difference* operator (first-order accurate)

$$(D^+\phi)_i = \frac{\phi_{i+1} - \phi_i}{\Delta x}$$

- D^-, *backward difference* operator (first-order accurate)

$$(D^-\phi)_i = \frac{\phi_i - \phi_{i-1}}{\Delta x}$$

For hyperbolic equations, information is propagated in specific directions (along the characteristic curves). Therefore central differencing, which uses information from both directions, is not very useful for numerically solving hyperbolic problems. Instead we prefer to use *upwinding*. In upwinding the idea is to choose the spatial discretization based on the direction that information is propagating. For the one-dimensional case,

if $u_i > 0$, we use $D^-\phi$
if $u_i < 0$, we use $D^+\phi$
if $u_i = 0$, then $u_i\phi_x = 0$, so we do not need to approximate ϕ_x

Upwinding is first-order accurate. If we instead choose the difference operators so that data is taken from the direction opposite the one from which it is propagating, then we are using *downwinding*, which is unstable.

The numerical errors resulting from upwind differencing cause *dissipation* in the numerical solution. The numerical errors resulting from central differencing cause *dispersion* in the numerical solution (for more information see Strikwerda, Chapter 5).

In any numerical method for solving PDEs, we must ultimately be concerned with *convergence*. The means that as we refine our grid in time and space, the numerical solution converges to an analytical solution. To this end we examine the notions of *consistency* and *stability*. For precise definitions of convergence, consistency, and stability, see Strikwerda, Chapter 1. If a scheme is consistent, the errors in approximating the differential operator vanish as Δt, $\Delta x \to 0$. This is typically proved by assuming a sufficiently smooth solution and using a Taylor series expansion. If the scheme is stable, the solution does not blow up in a finite time. Stability may be unconditional or conditional. In conditional stability, we have restrictions on the value of Δt.

It is possible to meet one but not both of these conditions. For example, it can be shown that discretizing the scalar, linear, constant-coefficient convection equation given above using central differencing in space and forward Euler in time is consistent, but unstable.

The *Lax-Richtmyer Equivalence Theorem* states that, given a consistent finite difference approximation to a linear partial differential equation, the approximation is convergent if and only if it is stable.

A necessary (but not sufficient) condition for convergence for all PDEs is the *Courant-Friedreichs-Lewy* condition (CFL condition), which asserts that the numerical domain of dependence[1] for each point must contain the physical domain of dependence. In the case of the advection equation, this implies that numerical waves must propagate at least as fast as physical waves, i.e. the numerical

[1]The *domain of dependence* for a point is the set of points that have an effect on the value of the solution of that point

wave speed $\Delta x/\Delta t$ must be at least as fast as the physical wave speed $|u|$. This leads us to the CFL time step restriction of

$$\Delta t < \frac{\Delta x}{\max\{|u|\}}$$

where $\max\{|u|\}$ is chosen to be the largest value of $|u|$ over the entire Cartesian grid. In practice this is usually enforced by choosing a CFL number α with

$$\Delta t\left(\frac{\max\{|u|\}}{\Delta x}\right) = \alpha$$

and $0 < \alpha < 1$. A common near optimal choice is $\alpha = .9$, and a common conservative choice is $\alpha = .5$. A multidimensional CFL condition can be written as

$$\Delta t \max\left\{\frac{|u|}{\Delta x} + \frac{|v|}{\Delta y} + \frac{|w|}{\Delta z}\right\} = \alpha$$

although

$$\Delta t\left(\frac{\max\{|\mathbf{u}|\}}{\min\{\Delta x, \Delta y, \Delta z\}}\right) = \alpha$$

is also frequently used in practice, despite the fact that it is incorrect.

Instead of upwinding, the spatial derivatives in equation 1 could be approximated with the more accurate central differencing method. Unfortunately, simple central differencing is unstable with forward Euler time discretization and the usual CFL conditions with $\Delta t \sim \Delta x$. We look at three possible ways of achieving stability while using central differencing.

1. Stability can be achieved by using a much more restrictive CFL condition with $\Delta t \sim (\Delta x)^2$. However, this is usually too computationally costly.

2. Stability can also be achieved by using a different temporal discretization, e.g. the third order accurate Runge-Kutta method.

3. A third way of achieving stability consists of adding some artificial viscosity to the right hand side of equation 1 to obtain

   $$\phi_t + \mathbf{u} \cdot \nabla\phi = \mu\Delta\phi$$

   where the viscosity coefficient μ is chosen proportional to Δx, i.e. $\mu \sim \Delta x$, so that the artificial viscosity vanishes as $\Delta x \to 0$ enforcing consistency for this method.

While all three of these approaches stabilize central differencing, we instead prefer to use upwind methods.

1.2 Higher Order Accuracy

In the previous sections we started with a PDE and discretized it to construct a numerical method. For convergence, we require that the numerical method be both consistent and stable, possibly imposing conditions on the time step (e.g., $\Delta t < \alpha\Delta x$) to achieve stability. Now we look at some higher order accurate discretizations. Our goal in using higher order accurate methods is to improve upon consistency. The order of accuracy of the method refers to the order of the local truncation error resulting from the discretization. That is, a method with $O(\Delta x^n)$ is considered to be have spatial order of accuracy n. Note too that the local truncation error may also contain mixed terms like $O(\Delta t\Delta x)$.

1.2.1 TVD Runge-Kutta

To achieve higher order accuracy in the temporal discretization, one can use *Total Variation Diminishing* (TVD) Runge-Kutta (RK) methods. These methods guarantee that the total variation of the solution does not increase, so that no new extrema are generated. For example, if our solution represents a temperature profile, spurious oscillations caused by the numerical method may trigger a chemical reaction in our simulation; using a TVD method would guarantee that such oscillations do not occur. A related concept is that of *Total Variation Bounded* (TVB) methods, where the growth of the total variation is bounded. Both concepts are related to stability. W consider 1st order, 2nd order, and 3rd order TVD RK.

- **1st order**

  The 1st order TVD RK method is identical to forward Euler and 1st order RK. It is given by

$$\frac{\phi^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \phi^n = 0.$$

- **2nd order**

  The 2nd order TVD RK method is also known as 2nd order RK, the midpoint rule, modified Euler, and Heun's predictor-corrector method.

  First, an Euler step is taken to advance the solution to time t^{n+1}

$$\frac{\hat{\phi}^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \phi^n = 0.$$

  This is followed by a second Euler step to advance the solution to time t^{n+2}

$$\frac{\hat{\phi}^{n+2} - \hat{\phi}^{n+1}}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \hat{\phi}^{n+1} = 0.$$

  Finally, a convex combination of the original and produced values is taken

$$\phi^{n+1} = \frac{\phi^n + \hat{\phi}^{n+2}}{2}$$

  to produce the second order accurate TVD approximation to ϕ at time t^{n+1}.

- **3rd order**

  First, an Euler step is taken to advance the solution to time t^{n+1}

$$\frac{\hat{\phi}^{n+1} - \phi^n}{\Delta t} + \mathbf{u}^n \cdot \nabla \phi^n = 0.$$

  This is followed by a second Euler step to advance the solution to time t^{n+2}

$$\frac{\hat{\phi}^{n+2} - \hat{\phi}^{n+1}}{\Delta t} + \mathbf{u}^{n+1} \cdot \nabla \hat{\phi}^{n+1} = 0.$$

This is followed by an averaging step

$$\hat{\phi}^{n+\frac{1}{2}} = \frac{3}{4}\phi^n + \frac{1}{4}\hat{\phi}^{n+2}$$

that produces an approximation to ϕ at time $t^n + \frac{1}{2}\Delta t$. Then another Euler step is taken to advance the solution to time $t^n + \frac{3}{2}\Delta t$

$$\frac{\hat{\phi}^{n+\frac{3}{2}} - \hat{\phi}^{n+\frac{1}{2}}}{\Delta t} + \mathbf{u}^{n+\frac{1}{2}} \cdot \nabla \hat{\phi}^{n+\frac{1}{2}} = 0$$

and is followed by a second averaging step

$$\phi^{n+1} = \frac{1}{3}\phi^n + \frac{2}{3}\hat{\phi}^{n+\frac{3}{2}}$$

that produces a third order accurate approximation to ϕ at time t^{n+1}. This third order accurate TVD RK method has a stability region that includes part of the imaginary axis. Thus, a stable (though ill-advised) numerical method results from combining third order accurate TVD RK with central differencing.

1.2.2 Hamilton-Jacobi ENO

Recall that in the first order accurate upwind differencing, we approximate the spatial derivative as follows:

If $u_i > 0$, use $\phi_x^- \approx D^-\phi = \frac{\phi_i - \phi_{i-1}}{\Delta x}$

If $u_i < 0$, use $\phi_x^+ \approx D^+\phi = \frac{\phi_{i+1} - \phi_i}{\Delta x}$

If $u_i = 0$, then $u_i\phi_x = 0$, so we do not need an approximation.

This scheme can be improved upon by using a more accurate approximation for ϕ_x^- and ϕ_x^+. The simplest way to approximate the spatial derivative is to assume that the function is piecewise linear. This is what the approximations D^+ and D^- do. However, we could also pass a parabola or higher order polynomial through the data points. We can then differentiate the interpolated polynomial to get the derivative.

The general idea behind *essentially non-oscillatory* (ENO) schemes is to use a higher order accurate polynomial to reconstruct ϕ and then differentiate it to get an approximation to ϕ_x. A different such polynomial is constructed at each grid point. The key to the algorithm is to choose the neighboring points for the interpolation so that we are not interpolating across steep gradients.

We describe the implementation of the HJ ENO method, in which we perform Newton polynomial interpolation via a divided difference table. The Newton polynomial interpolating a given set of n data points, $(x_1, \phi_1), (x_2, \phi_2), \ldots, (x_n, \phi_n)$ has the form

$$P_{n-1}(x) = \alpha_0 + \alpha_1(x - x_1) + \cdots + \alpha_{n-1}(x - x_1)\cdots(x - x_{n-1}).$$

The coefficients, α_j, are the entries in the divided difference table. We use the notation, D_i^k to represent the k^{th} divided difference at grid point i. The first few levels of the divided difference

table are constructed as follows:

$$\begin{aligned} D_i^0\phi &= \phi_i \\ D_{i+\frac{1}{2}}^1\phi &= \frac{D_{i+1}^0\phi - D_i^0\phi}{\Delta x} \\ D_i^2\phi &= \frac{D_{i+\frac{1}{2}}^1\phi - D_{i-\frac{1}{2}}^1\phi}{2\Delta x} \\ D_{i+\frac{1}{2}}^3\phi &= \frac{D_{i+1}^2\phi - D_i^2\phi}{3\Delta x} \\ \vdots &\quad \vdots \quad \vdots \end{aligned}$$

Note that the 0^{th} level is defined at grid nodes, the first level is midway between grid nodes, the third level at grid nodes, and so on. Also from the above table we can see that $D_{i+\frac{1}{2}}^1\phi = (D^+\phi)_i$ and $D_{i-\frac{1}{2}}^1\phi = (D^-\phi)_i$.

The divided differences are used to reconstruct a polynomial of the form

$$\phi(x) = Q_0(x) + Q_1(x) + Q_2(x) + Q_3(x)$$

that can be differentiated and evaluated at x_i to find $(\phi_x^-)_i$ and $(\phi_x^+)_i$. That is, we use

$$\phi_x(x_i) = Q_1'(x_i) + Q_2'(x_i) + Q_3'(x_i)$$

to define $(\phi_x^-)_i$ and $(\phi_x^+)_i$. Note that the constant $Q_0(x)$ term vanishes upon differentiation.

To find ϕ_x^- we start with $k = i-1$, and to find ϕ_x^+ we start with $k = i$. Then we define

$$Q_1(x) = (D_{k+1/2}^1\phi)(x - x_i)$$

so that

$$Q_1'(x_i) = D_{k+\frac{1}{2}}^1\phi$$

implying that the contribution from $Q_1'(x_i)$ in equation 1.2.2 is the backward difference in the case of ϕ_x^- and the forward difference in the case of ϕ_x^+. In other words, first order accurate polynomial interpolation is exactly first order upwinding. Improvements are obtained by including the $Q_2'(x_i)$ and $Q_3'(x_i)$ terms in equation 1.2.2 leading to second and third order accuracy, respectively.

Looking at the divided difference table and noting that $D_{k+1/2}^1\phi$ was chosen for first order accuracy, we have two choices for the second order accurate correction. We could include the next point to the left and use $D_k^2\phi$, or we could include the next point to the right and use $D_{k+1}^2\phi$. The key observation is that smooth, slowly varying data tends to produce small numbers in divided difference tables, whereas discontinuous or quickly varying data tends to produce large numbers in divided difference tables. This is obvious in the sense that the differences measure variation in the data. We would like to avoid interpolating near large variations such as discontinuities or steep gradients, since they cause overshoots in the interpolating function leading to numerical errors in the approximation of the derivative. Thus, if $|D_k^2\phi| \leq |D_{k+1}^2\phi|$ we set $c = D_k^2\phi$ and $k^\star = k-1$, otherwise we set $c = D_{k+1}^2\phi$ and $k^\star = k$. Then we define

$$Q_2(x) = c(x - x_k)(x - x_{k+1})$$

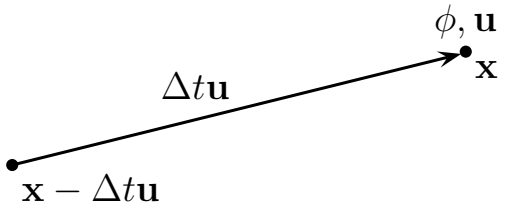

Figure 1: Data movement for semi-Lagrangian advection.

so that

$$Q_2'(x_i) = c(2(i-k)-1)\Delta x$$

is the second order accurate correction to the approximation of ϕ_x in equation 1.2.2. If we stop here, i.e. omitting the Q_3 term, we have a second order accurate method for approximating ϕ_x^- and ϕ_x^+. Note that we have not yet used $k^\star$. This is because it is only necessary when calculating the third order accurate correction.

Similar to the second order accurate correction, the third order accurate correction is obtained by comparing $|D^3_{k^\star+1/2}\phi|$ and $|D^3_{k^\star+3/2}\phi|$. If $|D^3_{k^\star+1/2}\phi| \le |D^3_{k^\star+3/2}\phi|$ we set $c^\star = D^3_{k^\star+1/2}\phi$, otherwise we set $c^\star = D^3_{k^\star+3/2}\phi$. Then we define

$$Q_3(x) = c^\star(x - x_{k^\star})(x - x_{k^\star+1})(x - x_{k^\star+2})$$

so that

$$Q_3'(x_i) = c^\star\big(3(i-k^\star)^2 - 6(i-k^\star) + 2\big)\Delta x^2$$

is the third order accurate correction to the approximation of ϕ_x in equation 1.2.2.

1.3 Semi-Lagrangian Advection

Previously we discretized the equation

$$\phi_t + \mathbf{u}\cdot\nabla\phi = 0$$

using e.g. 3rd order TVD RK for the temporal derivative and 3rd order ENO in a dimension by dimension approach for the spatial derivative. Here we consider an alternative which is lower order, but not dimension by dimension. This is called the *method of characteristics*.

To determine a new value for ϕ at time step $n+1$, we look back in the direction $\mathbf{u}$ a distance $\Delta t\|\mathbf{u}\|$ as shown in Figure 1. The new value of ϕ is given by

$$\phi^{n+1}(\mathbf{x}) = \phi^n(\mathbf{x} - \Delta t\mathbf{u})$$

The point $\mathbf{x} - \Delta t\mathbf{u}$ is generally not a grid point. To get a value for ϕ at that point, we use linear interpolation in $1D$ (bilinear interpolation in $2D$, or trilinear interpolation in $3D$). Let us look at the interpolation in detail in $1D$. Assume that the point $x_j - u\Delta t$ lies between grid points x_i and x_{i+1}, and that its distance from grid point x_i is s. Therefore its distance from grid point x_{i+1} is $\Delta x - s$. Then linear interpolation gives

$$\phi_j^{n+1} = \Big(1 - \frac{s}{\Delta x}\Big)\phi_i^n + \frac{s}{\Delta x}\phi_{i+1}^n$$

Since $0 \leq s \leq \Delta x$, we have $1 - \frac{s}{\Delta x} \geq 0$ and $\frac{s}{\Delta x} \geq 0$. Then

$$\phi_j^{n+1} = \Big(1 - \frac{s}{\Delta x}\Big)\phi_i^n + \frac{s}{\Delta x}\phi_{i+1}^n \leq \Big(1 - \frac{s}{\Delta x}\Big)\max\left(\phi_i^n, \phi_{i+1}^n\right) + \frac{s}{\Delta x}\max\left(\phi_i^n, \phi_{i+1}^n\right) = \max\left(\phi_i^n, \phi_{i+1}^n\right)$$

and

$$\phi_j^{n+1} = \Big(1 - \frac{s}{\Delta x}\Big)\phi_i^n + \frac{s}{\Delta x}\phi_{i+1}^n \geq \Big(1 - \frac{s}{\Delta x}\Big)\min\left(\phi_i^n, \phi_{i+1}^n\right) + \frac{s}{\Delta x}\min\left(\phi_i^n, \phi_{i+1}^n\right) = \min\left(\phi_i^n, \phi_{i+1}^n\right).$$

No new extrema are introduced, since

$$\min\left(\phi_i^n, \phi_{i+1}^n\right) \leq \phi_j^{n+1} \leq \max\left(\phi_i^n, \phi_{i+1}^n\right)$$

This fact is important for unconditional stability. For example, it immediately implies that the method is unconditionally stable in the max norm. This stability is a simple consequence of obtaining new values by copying interpolated values. Since linear, bilinear, and trilinear interpolation never create new extrema, semi-Lagrangian advection also does not. Note, however, that higher order interpolation can create new extrema. We may obtain higher order be using a more accurate estimate for the characteristic and higher order interpolation that is clamped to prevent new extrema.