AST560 2025: The Advection-Diffusion Equation

Ammar H. Hakim

March 4, 2025

Contents

1 Properties of the Advection-Diffusion Equation

The advection-diffusion equation is a fundamental equation that describes the transport of a scalar field, $f(\mathbf{x},t)$ in a given flow field. For scalar diffusion we can write this equation as

$$\frac{\partial f}{\partial t} + \nabla\cdot(\mathbf{u}f) = \nabla\cdot(\alpha\nabla f). \tag{1}$$

Here $\mathbf{u}(\mathbf{x},t)$ is a *given* time-dependent flow field, and $\alpha(\mathbf{x},t)$ is a *given* time-dependent diffusion coefficient. If we take the flow and diffusion as given, this is a *linear equation* for the scalar $f(\mathbf{x},t)$. Despite the apparent simplicity of this equation it is fundamental to understanding the physics of mixing processes in fluids and plasmas (which can be chaotic), and forms a good model and a starting point for more complicated equations. For example, the Vlasov-Maxwell equation and gyrokinetic equations are both advection-diffusion equations in *phase-space* and though nonlinear, can be solved with schemes similar to those we will develop for this linear equation.

Before we develop numerical methods to solve this equation we will derive some important properties of the continuous system. There are several reasons to do this analysis. First, it allows us to design schemes that also mimic these properties in the discrete limit. Not all continuous properties are inherited by the discrete scheme, and, in fact, it may be possible that some properties may be *harmful* to mimic. Second, as we will see, a careful study of the proofs of the continuous properties give us hints on how to construct proofs for the discrete properties.

We will begin by deriving a *weak-form* of Eq. (1). Let $w(\mathbf{x},t)$ be a smooth function[1]. Then, multiply Eq. (1) by w and integrate over an arbitrary volume Ω to get

$$\int_\Omega w\frac{\partial f}{\partial t}\, d^3\mathbf{x} + \oint_{\partial\Omega} w(\mathbf{u}f - \alpha\nabla f)\cdot\mathbf{n}\, ds - \int_\Omega \nabla w\cdot(\mathbf{u}f - \alpha\nabla f)\, d^3\mathbf{x} = 0. \tag{2}$$

In this expression $\partial\Omega$ is the surface bounding the volume Ω and $\mathbf{n}$ is the outward unit normal to $\partial\Omega$. We have rearranged the terms and also used integration by parts.

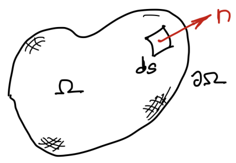

Figure 1: Arbitrary volume Ω, with closed bounding surface $\partial\Omega$ and outward-pointing surface normal $\mathbf{n}$.

This weak-form, in a sense, is more general than the PDE Eq. (1): as the derivatives are no longer on the solution variables, f can be *discontinuous*, and hence the weak-form allows a broader class of solutions than the PDE. Later we will use this weak-form to directly derive discrete schemes in the *discontinuous Galerkin* family of algorithms. For now, we will use the weak-form to prove a couple of important properties of the advection-diffusion equation.

Proposition 1. *The advection-diffusion equation conserves the total amount of scalar quantity. That is*

$$\frac{d}{dt}\int_\Omega f\, d^3\mathbf{x} + \oint_{\partial\Omega} (\mathbf{u}f - \alpha\nabla f)\cdot\mathbf{n}\, ds = 0. \tag{3}$$

Proof. This proof is trivial: just set $w = 1$ in the weak-form Eq. (2). □

[1] A smooth function is one that is continuous and has as many continuous derivatives as we need.

This proof shows something that recurs in other conservation laws, and so is important to understand. The proposition says that the amount of "stuff" in Ω is *conserved*, that is, it can only change due to the flow into and out of the volume Ω through its surface $\partial\Omega$. The quantity $(\mathbf{u}f - \alpha\nabla f)$ is called the *flux*. All conservation laws have this structure: some quantity an arbitrary volume can only change due to things flowing in and out of that volume.

Proposition 2. *The advection-diffusion equation monotonically decays the L_2-norm of f if the flow is incompressible, $\nabla \cdot \mathbf{u} = 0$, and $\alpha > 0$. The L_2-norm is conserved if $\nabla \cdot \mathbf{u} = 0$ and $\alpha = 0$.*

Proof. Choose $w = f$ in the weak-form Eq. (2) to get

$$\frac{d}{dt}\int_\Omega \frac{1}{2}f^2\, d^3\mathbf{x} + \oint_{\partial\Omega} (\mathbf{u}f^2 - \alpha\nabla\frac{1}{2}f^2)\cdot\mathbf{n}\, ds - \int_\Omega \nabla f \cdot (\mathbf{u}f - \alpha\nabla f)\, d^3\mathbf{x} = 0.$$

Now, if $\nabla \cdot \mathbf{u} = 0$ we can write $\nabla f \cdot \mathbf{u}f$ as $\nabla \cdot (\mathbf{u}f^2/2)$ to get

$$\frac{d}{dt}\int_\Omega \frac{1}{2}f^2\, d^3\mathbf{x} + \oint_{\partial\Omega} (\mathbf{u}f^2 - \alpha\nabla\frac{1}{2}f^2)\cdot\mathbf{n}\, ds - \int_\Omega \left(\nabla\cdot\frac{1}{2}\mathbf{u}f^2 - \alpha\|\nabla f\|^2\right) d^3\mathbf{x} = 0.$$

Doing an integration by parts on the first volume term we finally get

$$\frac{d}{dt}\int_\Omega \frac{1}{2}f^2\, d^3\mathbf{x} + \oint_{\partial\Omega} (\mathbf{u}\frac{1}{2}f^2 - \alpha\nabla\frac{1}{2}f^2)\cdot\mathbf{n}\, ds = -\int_\Omega \alpha\|\nabla f\|^2\, d^3\mathbf{x}. \tag{4}$$

If $\alpha > 0$ then this shows that the L_2-norm in Ω decays monotonically, and remains conserved if $\alpha = 0$. $\square$

Note that in this proof it was crucial that the flow is incompressible, $\nabla\cdot\mathbf{u} = 0$. Without this we can't show that the L_2-norm decays. In fact, in a homework problem you will be asked to construct a counter-example that explicitly shows that L_2-norm can *increase or decrease* for compressible flows.

The monotonic decay of the L_2-norm is an important property to ensure in a discrete scheme. In fact, even when $\alpha = 0$, we usually have to ensure that the *discrete scheme decays the L_2-norm*, even though the continuous equation conserves it. This is an example in which the discrete scheme must *not* mimic the continuous properties as otherwise the numerical algorithm is no longer *stable*.

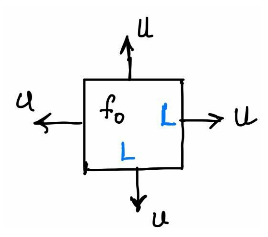

Figure 2: A small, cubical volume with outflow from all sides. If $f_0 > 0$ then then f can only decay towards zero, showing solution always remains positive if it is initially so.

We prove that solution remains positive if it is initially so. This is probably the hardest property to ensure in a discrete scheme.

Proposition 3. *If the solution initially positive everywhere, then it remains so for all times. That is, if $f(\mathbf{x}, 0) > 0$, then $f(\mathbf{x}, t) > 0$, for $t > 0$.*

Proof. Consider a small, cubical volume with sides L in which $f = f_0$ initially. As an extreme case, let the flow be outward on each side and with magnitude U. We can ignore diffusion in this box. Then the weak-form shows that we must have

$$\frac{df}{dt} = -\frac{6U}{L} f. \tag{5}$$

If f_0 and as $U > 0$, we must have that $f \sim e^{-6Ut/L}$. This means that f in the box can only decay towards zero, but never become negative. Any other flow configuration will add f to the box, hence independent of $\mathbf{u}$ the solution will always remain positive. □

The properties above hold for *arbitrary* volumes, and not just a *specific* volume. Such local properties are far more powerful (and restrictive) than *global* properties. For example, if we were to extend the volume for density conservation to the whole domain, then all one could say is that the total amount of f in the whole domain remains unchanged. However, this would not preclude f disappearing at some point in the domain and instantaneously reappearing somewhere else again. *Local* conservation laws will disallow this. As the volume is arbitrary, the only way f can decrease in this volume is by flux out of the surface.

2 Steps Needed in Constructing a Solver

Now that we have studied a few properties of the equation system we wish to discretize we can switch to constructing the discrete scheme. Notice that we have a *time-dependent* equations with first-order time-derivatives, and first and second order spatial derivatives. Hence, we must

- Choose a *computational grid or mesh* that is appropriate for the problem we wish to solve. For now we will assume simple, rectangular domains and so will use a uniform, rectangular mesh.

- Once we choose the mesh, we use a *discrete approximation* to the spatial derivative terms that appear in the equation. For this, we need to carefully consider where on the mesh to compute the various quantities that appear in our equation, and from this construct a discrete derivative operator of sufficient accuracy. The discrete derivative operators will be different, for example, if we use a curvilinear mesh or a triangular mesh. We will consider such meshes later in the course.
- We then need to decide how to advance the solution in time: we will replace the time derivative with a discrete approximation and *advance* or *march* the solution forward in time using a ODE solver. We will of course need to prescribed initial condition and apply the appropriate *boundary conditions*.

Each of these steps, mesh generation, constructing spatial discrete operators, and choosing a time-stepper are complicated and impact the quality of the numerical solution we will obtain. Once we have made all these choices we can use the *discrete* scheme to prove various properties. Not all properties of the continuous equations are inherited by the discrete scheme. Depending on the set of problems we wish to study, we may want to ensure some key properties of the continuous scheme are inherited by the discrete scheme.

In general, we will also find that there is a tradeoff between accuracy of the scheme and *robustness*. For example, a highly accurate scheme could violate positivity, or will develop spurious high-k modes when the spatial scales of the solution approach the grid spacing. In general, some form of *regularization*, for example, *limiters* or extra diffusion, may be needed to obtain a stable and useful solver.

Of course, implementing the scheme in computer code, specially to create a production solver is highly non-trivial: one needs to have good programming skills and follow good software engineering discipline. The appearance of modern parallel processors and GPUs has made the process even more complicated as, in general, one needs to have a deep understanding of the underlying hardware to obtain best performance from the solver.

Finally, I remark that production solvers applied to "real" physics or engineering problems will invariably encounter issue that will not show up in simple test cases. In fact, if anything *can* go wrong, it *will* go wrong. Eventually. Hence, it is important to have a deep understanding of *both* the physics of the equations *and* the properties of the discrete system. It is seldom fruitful or productive to treat a numerical method as a black-box. The deeper one understand the numerics, the better one can be confident in obtaining useful results from them.

3 Constructing and Analyzing Finite-Difference Formulas

In the advection-diffusion equation we have first-order and second-order derivative operators. We will use *finite-differences* to compute these, however, accounting for some form of *upwinding* for the advection term that we explain below.

3.1 Lagrange interpolating polynomials

Finite-difference (FD) approximations can be systematically constructed by constructing *interpolating polynomials* given values of a function at a set of discrete point. The key approach here is to construct an *interpolating* polynomial in the following way. Let x_i, $i = 1, \ldots, N$ be a set of locations on a 1D grid and $f = f(x_i)$ the corresponding values of f at those points. The first step in constructing FD scheme is to use these to construct a continuous polynomial $f_h(x)$ that satisfies the property that

$$f_h(x_i) = f_i \tag{6}$$

for $i = 1, \ldots, N$. One way to determine this polynomial is to assume a form

$$f_h(x) = a_0 + a_1 x + \ldots + a_{N-1} x^{N-1} \tag{7}$$

where a_j, $j = 0, \ldots, N-1$ are N coefficients. The condition $f_h(x_i) = f_i$ will give us N linear equations to determine the N unknowns a_j, hence determining the polynomial $f_h(x)$.

In reality we do not actually need to construct or solve any linear system. Instead, we can use *Lagrange interpolating* polynomials to construct $f_h(x)$ directly. To illustrate, let $N = 3$. Then the interpolating polynomial is simply

$$f_h(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f_1 + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f_2 + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f_3. \tag{8}$$

Note the construction of this polynomial: when $x = x_1$, for example, the second and third terms vanish, and we get $f_h(x_1) = f_1$, as required. Hence, this quadratic polynomial is the interpolation polynomial we are seeking. Of course, the generalization to more points is obvious.

Interpolation on uniformly spaced points can often give highly inaccurate results. For example, consider the function

$$f(x) = \frac{1}{1 + 10x^2} \tag{9}$$

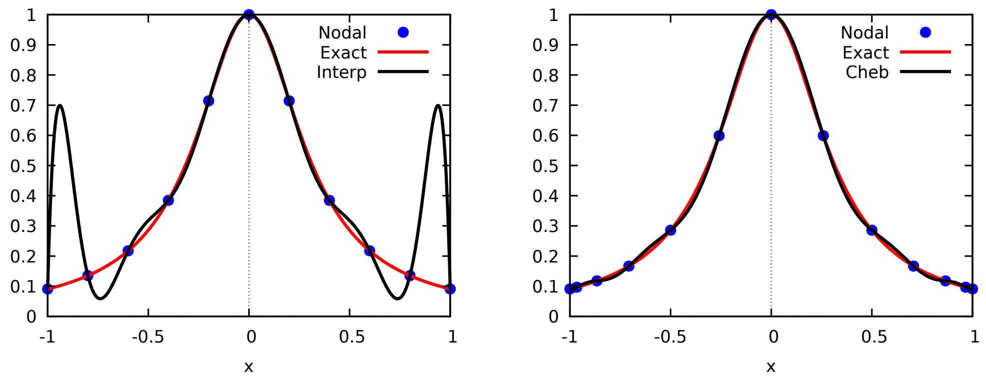

Figure 3: Left: Interpolation of a smooth function on uniformly spaced set of nodes. Black line is the interpolation polynomial that shows severe oscillations, despite the function (red line) being smooth. Right: The same, except using Chebyshev interpolation points. Using these significantly improves the quality of the interpolation.

on $x \in [-1, 1]$, This function is smooth but when an interpolating polynomial is constructed using uniformly spaced points it behaves in a pathological manner. One way around this "ringing" is to use a *non-uniformly* spaced sets of points, for example, Legendre or Chebyshev points. See Fig. 3.

3.2 Constructing difference approximations

Once we have the interpolation polynomial we can use it to approximate the values and derivatives we need at various places in the algorithm. For example, at some arbitrary point x we can approximate $f(x) = f_h(x)$, or

$$\frac{df(x)}{dx} \approx \frac{df_h(x)}{dx}. \tag{10}$$

Evaluation of the interpolation polynomial and its derivative is often made much easier by use of a compute algebra system (CAS). I highly recommend you use some CAS like Maple, Mathematica, or (my personal choice) Maxima. The use a CAS greatly simplified the various algebraic manipulations needed and also reduces the possibility of errors.

As example, consider we have three cells in a uniform, 1D grid with cell spacing Δx. We will choose to locate the f at *cell-centers*. Then we can use the interpolation formula above to construct a polynomial $f_h(x)$ in the middle cell, $-\Delta x/2 \le x \le \Delta x/2$, assuming, without loss of generality, that $x_0 = 0$. Then, for example, we can compute as approximation to the second derivative at the point $x_0 = 0$ by computing $d^2 f_h/dx^2$ and evaluating the result at $x = 0$. We get the expected central difference formulas

$$\frac{\partial f_h}{\partial x} = \frac{f_R - f_L}{2\Delta x} \tag{11}$$
$$\frac{\partial^2 f_h}{\partial x^2} = \frac{f_R - 2f_0 + f_L}{\Delta x^2}. \tag{12}$$

This, of course, should not be surprising to you but illustrates the general procedure for computing approximations from the interpolating polynomials.

As we will see below, we will also need to compute the values f_R and f_L marked in Fig. 4. Again, we simply evaluate the Lagrange interpolating polynomial at $x = \pm\Delta x/2$ to get

$$f^+ = \frac{1}{8}(3f_R + 6f_0 - f_L) \tag{13a}$$
$$f^- = \frac{1}{8}(3f_L + 6f_0 - f_R). \tag{13b}$$

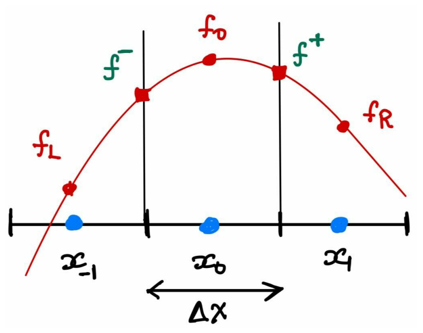

Figure 4: Three cells of a uniform 1D grid. The red curve is a quadratic that fits the cell-center values f_L, f_0 and f_R and is used to construct finite-difference schemes.

3.3 Taylor-series analysis

For a smooth $f(x)$ the accuracy of the interpolating polynomial at some point x will depend on the size of the *stencil*, that is, the number of points (cell-centers in this case) that are used in constructing it. To determine the accuracy and convergence of the interpolating polynomial we use a Taylor expansion of $f(x)$ around $x = 0$:

$$f(x) = \sum_{n=0} f^{(n)} \frac{x^n}{n!} \tag{14}$$

where $f^{(n)}$ is the n-th derivative of $f(x)$ evaluated at $x = 0$. We can now use this expansion in the finite-difference expressions and compare with the exact derivatives. For example we can compute

$$f_R = f(\Delta x) = f_0 + \Delta x f^{(1)} + \frac{\Delta x^2}{2} f^{(2)} + \frac{\Delta x^3}{6} f^{(3)} + \ldots \tag{15}$$
$$f_L = f(-\Delta x) = f_0 - \Delta x f^{(1)} + \frac{\Delta x^2}{2} f^{(2)} - \frac{\Delta x^3}{6} f^{(3)} + \ldots. \tag{16}$$

Using this in the finite-difference formula for the first derivative we get

$$\frac{\partial f_h}{\partial x} = \frac{f_R - f_L}{2\Delta x} = f^{(1)} + \frac{\Delta x^2}{6} f^{(3)} + \ldots \tag{17}$$

This shows that with the 3-point stencil, the first derivative computed using the finite-difference formula is *second-order accurate*. In the same way we can compute

$$\frac{\partial^2 f_h}{\partial x^2} = \frac{f_R - 2f_0 + f_L}{\Delta x^2} = f^{(2)} + \frac{\Delta x^2}{12} f^{(4)} + \ldots \tag{18}$$

again showing that the second derivative is also *second-order accurate*.

We can also compute the accuracy of the finite-difference formulas for computing the edge values $f^{\pm}$. Using the Taylor expansions in Eq. (13) we get

$$f^{\pm} - f(\pm\Delta x/2) = \pm\frac{\Delta x^3}{16} f^{(3)} + \ldots \tag{19}$$

showing that the edge values are computed to *third-order* accuracy.

The validity of the above Taylor series analysis, of course, depends crucially on the fact that the solution $f(x)$ is sufficiently smooth that the needed derivatives exist. This may not always be true, for example, in the presence of shocks or sharp gradients in the solution. It is also clear that the degree of smoothness we need will depend on the stencil size (and the order of the derivative we are computing): hence, wider stencils (more accurate derivatives) will be increasingly *less* robust, with potentially large errors in regions of sharp gradients. In general, this means that unless great care is taken, high order methods (third or higher order) are far *less robust* than lower order methods (second-order). This is one of the key reasons that many commercial computational physics packages, especially for fluid mechanics[2], typically use low-order schemes, sometimes even effectively just first-order schemes.

3.4 Fourier analysis

Besides Taylor series analysis, we can also do a Fourier analysis to understand how a single mode is represented by the finite-difference formula for the derivative operators. To do this we will take a single Fourier mode

$$f(x) = e^{ikx} \tag{20}$$

[2]Solid mechanics and heat transfer problems can be solved with higher order methods, as the solutions are typically much smoother that fluid problems. In the latter, specially in invicid or low viscosity fluids, the flow tends to cascade to shorter wavelengths (high-k) modes.

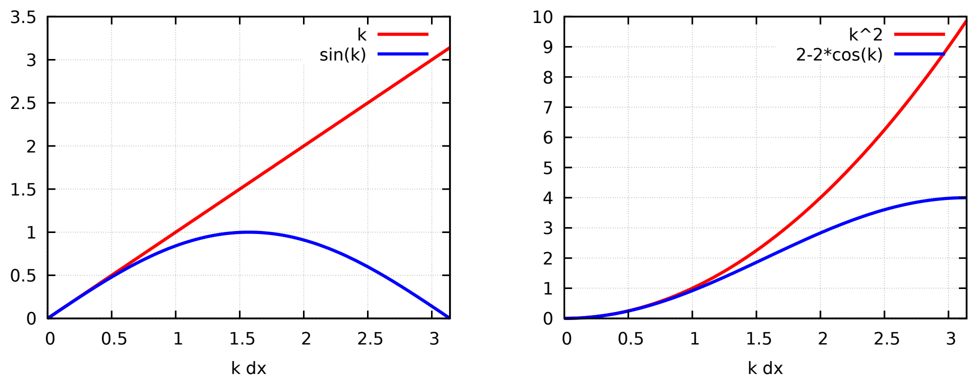

Figure 5: Left: Numerical dispersion relation (blue) plotted as a function of wave-number for central difference scheme for the constant advection. Significant dispersion and aliasing of modes is seen. Right: Damping of wave-number k for the constant-coefficient diffusion equation. High-k modes are *under-damped*. Red curves are the exact dispersion relations.

and substitute this in the finite-difference formula. For, example, consider the linear advection equation

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0. \tag{21}$$

with its semi-discrete form

$$\frac{\partial f_h}{\partial t} + u\frac{f_R - f_L}{2\Delta x} = 0. \tag{22}$$

Using the Fourier expansion $f_h(x) = e^{-i\omega t}e^{ikx}$ we get

$$-i\omega + u\frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2\Delta x} = 0 \tag{23}$$

This gives a numerical dispersion relation

$$\omega\Delta x = u\bar{k}\Delta x = u\sin(k\Delta x). \tag{24}$$

Hence, the discrete scheme "sees" a *modified wave-number* $\bar{k}$ rather than the exact wave-number k. In general, the modified wave-number will have both a real and an imaginary part. However, due to the symmetry of the central-difference formula, $\bar{k}$ only has a real-part. This means that the mode propagates *without any damping*.

The left panel of Fig. 5 compares the numerical dispersion with the exact dispersion. Notice there is strong deviation from the exact dispersion for even modest waves numbers.

This figure also shows strong *aliasing* of modes: each long wavelength mode is *aliased* to a corresponding short wavelength mode. Further, this figure also shows that if we were to use the central-difference formula for advection, modes with different wave-numbers would propagate with different phase- and group velocities, causing *dispersion* of waves.

Now consider a constant-coefficient diffusion equation

$$\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial x^2} \tag{25}$$

with its semi-discrete form

$$\frac{\partial f_h}{\partial t} = \alpha \frac{f_R - 2f_0 + f_L}{\Delta x^2}. \tag{26}$$

In a similar manner as above we can do a Fourier analysis assuming a central-difference formula for the second derivative. Again using $f_h(x) = e^{-i\omega t}e^{ikx}$ we will get the dispersion relation

$$-i\omega = \alpha \frac{e^{ik\Delta x} - 2 + e^{-ik\Delta x}}{\Delta x^2}. \tag{27}$$

This dispersion relation shows that a mode with wave-number k will be damped as $e^{-\alpha \bar{k}^2 t}$ where

$$\bar{k}^2 \Delta x^2 = 2 - 2\cos(k\Delta x). \tag{28}$$

Recall that the actual damping should be $e^{-\alpha k^2 t}$. Hence, as we see in the right-panel of Fig. 5 the high-k modes will be *under-damped* by this central difference scheme.

3.5 Discrete L_2 conservation, monotonicity and Godunov's Theorem

The Fourier analysis of the semi-discrete 1D diffusion equation with central differences shows that there is no damping of modes. This indicates that the central difference scheme for advection (without any diffusion) *conserves* the L_2 norm. To see this explicitly, consider the semi-discrete scheme in cell i:

$$\frac{df_i}{dt} + \frac{f_{i+1} - f_{i-1}}{2\Delta x} = 0 \tag{29}$$

where we assumed $u = 1$ without any loss of generality. Here, let $i = 1, \ldots, N$. Now, multiply by f_i and sum over all cells to get

$$\frac{d}{dt}\sum_{i=1}^{N} \frac{1}{2} f_i^2 + \frac{1}{2\Delta x}\sum_{i=1}^{N} f_i(f_{i+1} - f_{i-1}) = 0. \tag{30}$$

To proceed, we will perform a *discrete summation-by parts*, or *index-shifting*. This is analogous to integration by parts. Write

$$\sum_{i=1}^{N} f_i f_{i+1} = \sum_{j=2}^{N+1} f_{j-1} f_j = \sum_{j=1}^{N} f_{j-1} f_j + \underbrace{(f_N f_{N+1} - f_0 f_1)}_{\text{Boundary}-\text{terms}}. \tag{31}$$

Here we replaced the dummy index i by $j = i + 1$, and introduced the *boundary terms*. Hence, substituting this in the semi-discrete scheme, and replacing the dummy index j by i, we get

$$\frac{d}{dt} \sum_{i=1}^{N} \frac{1}{2} f_i^2 + \frac{f_N f_{N+1} - f_0 f_1}{2\Delta x} = 0. \tag{32}$$

Notice that the discrete spatial term has cancelled, except for the boundary terms. Hence, we see that the L_2 norm of the solution only changes due to boundary terms, and hence L_2 is *conserved*[3].

Notice that we have not said anything about the time-derivative term in this analysis: this proof of L_2 conservation only holds in the *time-continuous* limit of the scheme. Depending on the time integrator we choose the L_2 for the *fully discrete* scheme may actually *decay* (for example, with strong-stability preserving Runge-Kutta schemes described below).

It may appear that it is good news that the central difference scheme conserves the L_2 norm. However, it is not so. We can again use the dispersion relation for the linear advection equation we derived above (see Eq. (24)) to see this, even without writing any code. Take an initial condition $f(x, 0)$ and Fourier decompose it. Propagate each Fourier mode in time using the dispersion relation. Take the inverse Fourier transform. If we do this for a simple Gaussian $f(x, 0) = e^{-a^2x^2}$ we see that for $u = 1$ as the Guassian propogates it creates large regions of spurious oscillations as well as negativity in its wake. Despite appearances, the L_2 norm of the red and black curves in the figure are identical.

As you will show in homework problems, a first-order *upwind* scheme, though monotonically decaying the L_2 norm, does not create such spurious oscillations. However, first-order upwind scheme has a large amount of spurious *numerical diffusion*.

We can reduce the numerical diffusion by using a high-order upwind scheme. This will improve the solution quality, still decaying L_2 norm. However, the monotonicity problems will persist. In fact, as we will show below, there is simply speaking *no* **linear**

[3]If we used periodic boundary conditions, then $f_0 = f_N$ and $f_{N+1} = f_1$. With this, the boundary terms cancel.

scheme higher than first-order that preserves monotonicity of the solution. This highly depressing statement goes by the name of Godunov's theorem and has caused untold amount of problems in contructing "good schemes" for advective equations (which includes all hyperbolic PDEs). Godunov's theorem forces us to look for *non-linear* schemes, leading to the concept of *limiters*, that are of central importance in our ability to simulate flows with shocks and sharp features.

Theorem 1 (Godunov's Theorem)**.** *There are no linear, higher than first-order upwind schemes that preserve monotonicity of the solution to the linear advection equation, and hence, for the solution of any hyperbolic PDE.*

Proof. Consider a fully-discrete, general scheme for advection equation

$$f_j^{n+1} = \sum_k c_k f_{j+k}^n.$$

The discrete slope then is

$$f_{j+1}^{n+1} - f_j^{n+1} = \sum_k c_k \left(f_{j+k+1}^n - f_{j+k}^n \right).$$

Assume that all $f_{j+1}^n - f_j^n > 0$. To maintain monotonicity at next time-step hence one must have all $c_k \geq 0$. The *only* scheme that satifies this condition is the first-order upwind scheme:

$$f_j^{n+1} = f_j^n - \frac{\Delta t}{\Delta x}(f_j^n - f_{j-1}^n) = \left(1 - \frac{\Delta t}{\Delta x}\right) f_j^n + f_{j-1}^n.$$

This satisfies monotonicity as long as $\Delta t/\Delta x \leq 1$. Any other scheme, including the second-order central difference scheme,

$$f_j^{n+1} = f_j^n - \frac{\Delta t}{2\Delta x}(f_{j+1}^n - f_{j-1}^n)$$

will have *at least one* $c_k < 0$ indepedent of the time-step, hence showing that they it not preserve monotonicity. □

4 Spatial Discretization of the Advection-Diffusion Equation

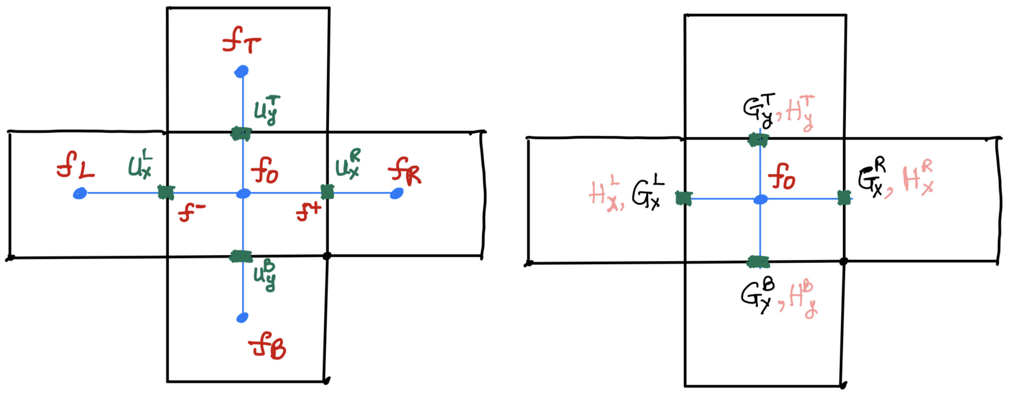

Figure 7: Stencils for spatial discretization of 2D advection-diffusion equations. We locate f at cell-centers, u_x on the middle of the vertical edges, and u_y at the middle of the horizontal edges. The diffusion coefficient α is located at the middle of each edge. The right-panel shows the spatial stencil written in terms of the advective and diffusive *fluxes* instead.

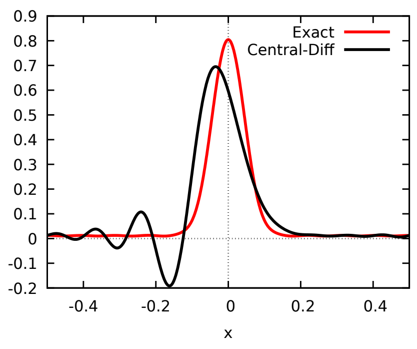

Figure 6: Advection of a Gaussian (red) with a central-difference scheme shows large, spurious oscillations (black), even though the L_2 norm is conserved.

We are now finally ready to write down the spatial discretization of the advection-diffusion equation. We will do this on a rectangular grid for the 2D equation

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}(u_x f) + \frac{\partial}{\partial y}(u_y f) = \frac{\partial^2}{\partial x^2}(\alpha f) + \frac{\partial^2}{\partial y^2}(\alpha f). \tag{33}$$

We will discretize the equation on the "stencil" shown in the figure: f is located at cell-centers, and u_x and u_y along the vertical and horizontal faces respectively. Note that this choice of location of the velocity ensures we know the *normal* component of the velocity at each face. The diffusion coefficient are located at the middle of each faces. See left-panel of Fig. 7.

We will write the scheme in *semi-discrete* form (as we will not discretize the time derivative yet), as follows

$$\frac{df_0}{dt} + \frac{G_x^R - G_x^L}{\Delta x} + \frac{G_y^T - G_y^B}{\Delta y} = \frac{H_x^R - H_x^L}{\Delta x} + \frac{H_y^T - H_y^B}{\Delta y} \tag{34}$$

where $G_{x,y}^{R,L}$ are *advective fluxes* and $H_{x,y}^{R,L}$ are *diffusive fluxes*. This stencil, written in terms of *fluxes* allows us to split the choice of fluxes into a seperate step, allowing flexibility in constructing different schemes by choosing different forms of the fluxes. See right-panel of Fig. 7.

A simple, but natural choice for the fluxes are as follows. For the advective flux we will use

$$G_x^L = \frac{u_x^L}{2}(f_0 + f_L) \tag{35}$$

$$G_y^B = \frac{u_y^B}{2}(f_0 + f_B) \tag{36}$$

with analogous formulas for the advective fluxes on the other two faces. This simple *averaging*, as is easy to see, is the second-order, central-difference scheme for the advection terms

For the diffusive flux we will use

$$H_x^L = \alpha_L \frac{f_0 - f_L}{\Delta x} \tag{37}$$

$$H_y^B = \alpha_B \frac{f_0 - f_B}{\Delta y} \tag{38}$$

This centered difference approximation the diffusive flux, as is easy to see, is the second-order, central-difference scheme for the diffusion terms.

We can make other choices for the advective and diffusive fluxes, leading to schemes with different properties. In particular, the use of centeral differences for the advection terms is particularly troublesome as we saw in the previous section. A better scheme would be to use an *upwind* or *upwind* biased scheme. To construct such a scheme, we write the advective flux as

$$G_x^L = u_x^L f^- \tag{39}$$

(see left-panel Fig. 7). Of course, we need to still choose f^-. One way to do this is to use *first-order upwinding*

$$f^- = f_L, \qquad u_x^L > 0. \tag{40a}$$

$$f^- = f_0, \qquad u_x^L \leq 0. \tag{40b}$$

Though this first-order upwind scheme will not produce spurious oscillations, it has a lot of numerical diffusion. We can, instead use a higher-order *upwind-biased* scheme by using the formulas, Eq. (13), for the edge-values we derived before. Using those we get

$$f^{-} = \frac{1}{8}(3f_0 + 6f_L - f_{LL}), \qquad u_x^L > 0. \tag{41a}$$

$$f^{-} = \frac{1}{8}(3f_L + 6f_0 - f_R), \qquad u_x^L < 0. \tag{41b}$$

Here the value f_{LL} is the value in the cell to the left to that of f_L. This higher-order scheme will still *not* prevent some spurious oscillations (as shown by Godunov's theorem), but will greatly reduce the numerical diffusion, and increase the accuracy of the solution.

5 Strong-Stability preserving Runge-Kutta time-steppers

To update the solution in time we will use Strong-Stability preserving Runge-Kutta (SSP-RK) time-steppers. I will not describe how these schemes are derived, but suffice it to say that these are *explicit* schemes constructed in a manner that if a *forward-Euler* step preserves, for example, monotonicity or positivity, then the SSP-RK scheme preserves it also.

Three steppers are described below: SSP-RK2 (second-order in time), SSP-RK3 and a four-stage SSP-RK3 (both third-order in time) that allows twice the CFL (for the cost of additional memory) as the other schemes. Here, the symbol $\mathcal{F}$ is used to indicate a first-order Euler update:

$$\mathcal{F}[f,t] = f + \Delta t \mathcal{L}[f,t] \tag{42}$$

where $\mathcal{L}[f]$ is the RHS operator from the spatial discretization.

SSP-RK2

$$f^{(1)} = \mathcal{F}[f^n, t^n] \tag{43}$$

$$f^{n+1} = \frac{1}{2}f^n + \frac{1}{2}\mathcal{F}[f^{(1)}, t^n + \Delta t] \tag{44}$$

with CFL ≤ 1.

Three-stage SSP-RK3

$$f^{(1)} = \mathcal{F}[f^n, t^n] \tag{45}$$
$$f^{(2)} = \frac{3}{4}f^n + \frac{1}{4}\mathcal{F}[f^{(1)}, t^n + \Delta t] \tag{46}$$
$$f^{n+1} = \frac{1}{3}f^n + \frac{2}{3}\mathcal{F}[f^{(2)}, t^n + \Delta t/2] \tag{47}$$

with $\mathrm{CFL} \leq 1$.

Four-stage SSP-RK3

$$f^{(1)} = \frac{1}{2}f^n + \frac{1}{2}\mathcal{F}[f^n, t^n] \tag{48}$$
$$f^{(2)} = \frac{1}{2}f^{(1)} + \frac{1}{2}\mathcal{F}[f^{(1)}, t^n + \Delta t/2] \tag{49}$$
$$f^{(3)} = \frac{2}{3}f^n + \frac{1}{6}f^{(2)} + \frac{1}{6}\mathcal{F}[f^{(2)}, t^n + \Delta t] \tag{50}$$
$$f^{n+1} = \frac{1}{2}f^{(3)} + \frac{1}{2}\mathcal{F}[f^{(3)}, t^n + \Delta t/2] \tag{51}$$

with $\mathrm{CFL} \leq 2$. Note that this scheme has four stages, but allows twice the time-step that SSP-RK2 and SSP-RK3, hence will result in a speed up of :math:‘1.5X‘ compared to the three-stage SSP-RK3 scheme.

5.1 Region of absolute stability

For any ODE solver, like the SSP-RK schemes presented above, one can compute *regions of stability*, that is, the bounds on time-step to ensure the solution does not “blow up”. To do this, we consider the simple first-order ODE

$$\dot{f} = (\lambda - i\omega)f. \tag{52}$$

Of course, the exact solution to this is simply a damped oscillation, $f = f_0 e^{\lambda t} e^{-i\omega t}$. Clearly, we must restrict $\lambda \leq 0$ to ensure the solutions are not growing.

Consider first the forward-Euler scheme:

$$f^{n+1} = \mathcal{F}[f^n, t] = [1 + (\lambda - i\omega)\Delta t]\, f^n. \tag{53}$$

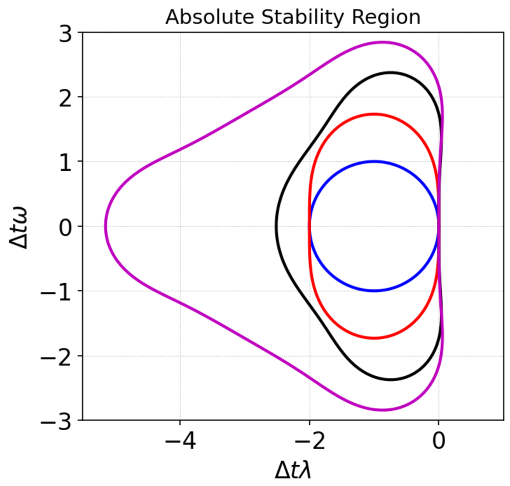

Figure 8: Absolute stability regions for a ODE $\dot{f} = (\lambda - i\omega)f$ for forward-Euler (blue), SSP-RK2 (red), SSP-RK3 (black) and four stage SSP-RK3 (magenta). When there is no diffusion ($\lambda = 0$) the SSP-RK2 scheme is slightly unstable as it has no intercept on the imaginary axis. Hence, the third order schemes should be preferred.

For the numerical solution to not grow we must ensure that

$$\left|\frac{f^{n+1}}{f^n}\right| \leq 1. \tag{54}$$

For the forward-Euler scheme we get

$$\left|\frac{f^{n+1}}{f^n}\right| = \sqrt{(1+\lambda\Delta t)^2 + \omega^2\Delta t^2} \leq 1. \tag{55}$$

Hence, the *region of stability* is a unit circle in the $(\omega\Delta t, \lambda\Delta t)$ plane centered around $\lambda\Delta t = -1$. In the special case $\lambda = 0$, that is, when we have no damping at all, this shows that for stability we need to ensure $\sqrt{1+\omega^2\Delta t^2} \leq 1$ which is impossible for any $\omega \neq 0$. Hence, the forward-Euler scheme is *unconditionally unstable*.

However, even though the forward-Euler scheme is unconditionally unstable, we can use it as a building block for *conditionally stable* schemes. In fact, each of the SSP-RK schemes above are built by combining forward-Euler steps and all conditionally stable. Figure 8 shows the region of stabilty for each of the SSP-RK schemes (and the forward-Euler scheme) listed above. For stability, we must ensure that, for a given ω and λ, we remain inside the region enclosed by the corresponding contour.

Notice from Fig. 8 that the region of stability increases as more stages are added to the scheme. Further, also notice that the region increases more rapidly along the $\lambda\Delta t$ axis, indicating that adding additional stages increases the maximum allows stable time-step for damping faster than it does for oscillations.

Consider a 1D advection-diffusion equation with constant advection speed and diffusion coefficient

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = \alpha\frac{\partial^2 f}{\partial x^2} \tag{56}$$

and look at a single mode $f \sim e^{ikx}$. Then this PDE turns into a ODE

$$\frac{\partial f}{\partial t} = (-iuk - \alpha k^2)f. \tag{57}$$

If we have used a semi-discrete scheme then if we use the ODE steppers listed above to advance the solution in time, we must ensure that the scheme remains stable with the highest-k in our grid. Let this be $k_{\max}$. In general, we will have $k_{\max} \sim 1/\Delta x$ with exact form dependent on the order of the spatial scheme we choose. Hence, we must have that $\omega \sim u/\Delta x$ and $\lambda \sim \alpha/\Delta x^2$.

What this analysis shows is that as we refine the grid, the stable time-step due to advection will decrease linearly with cell-size, but due to diffusion will decrease *quadratically* due to the diffusion. This quadratic dependence of time-step on diffusion can often be very restrictive, and make explicit schemes often prohibitively expensive. Hence, some form of *implicit* methods must be used. We will look at one such implicit method, called *super time-stepping* later in the course.