AOS180 – Numerical Methods in Atmospheric Sciences
Notes for Spring 2023

Marcelo Chamecki

June 12, 2023

Chapter 1

Introduction

The principles that govern fluid motion in the atmosphere and in the ocean are fairly simple: conservation of mass, conservation of energy, and Newton's second law of motion. However, when applied to moving fluids, these simple principles lead to a very complex set of partial differential equations. Because we cannot obtain exact solutions for these equations under conditions of practical interest, we typically resort to numerical solutions obtained on a computer. In general, numerical analysis is the study of methods to obtain approximate numerical solutions to a mathematical problem. Computational fluid dynamics (CFD) is the use of numerical analysis to solve problems that involve fluid flows. The focus of this course is on numerical solution of partial differential equations (PDEs) with emphasis on the equations that describe conservation laws in the atmosphere and the ocean.

1.1 A quick review of the primitive equations

Even though we will not use these equations until much later in the course, we will start by quickly reviewing them so that we are aware of their mathematical nature. Then we will proceed to develop numerical methods that are designed to address each the different (physical and mathematical) components of these equations.

1.1.1 Conservation of mass

The conservation of mass is written as

$$\frac{D\rho}{Dt} + \rho \nabla \cdot \mathbf{u} = 0, \tag{1.1}$$

where ρ is the fluid density and $\mathbf{u} = (u, v, w)$ is the velocity vector. The first term is the material derivative of the density, which represents the changes in density following a fluid

parcel and can be written as

$$\frac{D\rho}{Dt} = \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho. \tag{1.2}$$

Here the two terms on the right-hand side are the local (i.e., fixed in space) change in density (also called density tendency in meteorology jargon) and the advection of density by the flow. The second term in equation (1.1) represents the fractional change in volume of a fluid parcel

$$\nabla \cdot \mathbf{u} = \frac{\delta V}{V}, \tag{1.3}$$

where V is the volume of the fluid parcel. Thus, equation (1.1) simply states that any change in density of a fluid parcel must be accompanied by a change in volume, in such a way that the total mass is conserved.

1.1.2 Newton's second law

If we apply Newton's second law to a moving fluid, it results in an equation usually referred to as the Navier-Stokes equations, or the momentum equations:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} - 2\boldsymbol{\Omega} \times \mathbf{u} + \nu \nabla^2 \mathbf{u}. \tag{1.4}$$

Here, p is the pressure, $\mathbf{g} = (0, 0, -g)$ is the gravitational acceleration, $\boldsymbol{\Omega}$ is the angular velocity of the planet, and ν is the kinematic viscosity of the fluid. The left-hand side represents the acceleration of a fluid parcel, and the right-hand side contains all the forces (per unit mass) relevant here: pressure-gradient force, gravitational force (lumped together with centrifugal force), Coriolis force, and viscous force, respectively. This is Newton's second law written as $\mathbf{a} = (1/m)\mathbf{F}$. Note that in this case the material derivative can be expanded as

$$\frac{D\mathbf{u}}{Dt} = \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}. \tag{1.5}$$

Equation (1.4) is a vector equation, which can be written as 3 separate equations for each component of acceleration. As an example, the zonal component is usually written as

$$\frac{Du}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + 2|\boldsymbol{\Omega}| \sin(\phi) v - 2|\boldsymbol{\Omega}| \cos(\phi) w + \nu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} \right). \tag{1.6}$$

We usually define the Coriolis parameter $f = 2|\boldsymbol{\Omega}| \sin(\phi)$ and neglect the term $-2|\boldsymbol{\Omega}| \cos(\phi) w$.

1.1.3 First law of thermodynamics

We usually write the conservation of energy as

$$\frac{dU}{dt} + p\frac{dV}{dt} = \dot{Q}. \tag{1.7}$$

Here, U is the internal energy, the second term on the left-hand side is the work done by pressure, and $\dot{Q}$ is the diabatic heating rate (in the atmosphere this comes mostly from radiative effects and sensible and latent heating). If we work with an ideal gas (such as air), this equation can be written as an equation for the gas temperature (T):

$$c_p\frac{DT}{Dt} - \frac{1}{\rho}\frac{Dp}{Dt} = \dot{Q} + \nu_T c_p \nabla^2 T. \tag{1.8}$$

Here, c_p is the specific heat at constant pressure for air and the product ν_T is the molecular diffusivity for heat. We further relate pressure, temperature, and density by introducing the ideal gas law

$$p = \rho R_D T, \tag{1.9}$$

where R_D is the gas constant for dry air. Note that for oceanic flows the equation of state is more complicated as both salinity and temperature impact density. Nevertheless, this does not change the mathematical character of the resulting equations.

1.1.4 Primitive equations

The final set of primitive equations governing atmospheric flows can be written as

$$\frac{\partial \rho}{\partial t} + \mathbf{u}\cdot\nabla\rho + \rho\nabla\cdot\mathbf{u} = 0, \tag{1.10}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{g} - 2\boldsymbol{\Omega}\times\mathbf{u} + \nu\nabla^2\mathbf{u}, \tag{1.11}$$

$$\frac{\partial T}{\partial t} + \mathbf{u}\cdot\nabla T - \frac{1}{\rho c_p}\frac{Dp}{Dt} = \frac{1}{c_p}\dot{Q} + \nu_T\nabla^2\theta, \tag{1.12}$$

$$p = \rho R_D T. \tag{1.13}$$

These 6 equations completely determine the time evolution of 6 variables of interest (ρ, u, v, w, T, and p). However, they consist of a complex system of coupled, nonlinear, partial differential equations whose solution is not trivial. Note that they involve only first-order derivatives in time, and first- and second-order derivatives in spacial coordinates. In principle, with an appropriate set of initial and boundary conditions, we can find the solution.

The challenge is that this set of equations is not suitable for solution on a computer, as computers are good with discrete mathematics and our system involves variables that are

continuous functions of space and time (resulting in partial differential equations). To make efficient use of computational resources, we must approximate the system of equations by a system of algebraic equations that relates the values of these variables in a finite set of points in space-time. We illustrate this idea in the next section.

1.2 Basic idea of discretization

Here we will use a very simple example to illustrate the main idea of discretization. Suppose we are interested in calculating the vertical position z_p of an object that is falling in a vacuum. The object starts from rest, and only its vertical velocity and position are of interest here. In this case, the system is governed by Newton's second law given by

$$\frac{d^2 z_p}{dt^2} = -g. \tag{1.14}$$

It is usually better to split the second-order ODE (1.14) into two first-order ODEs by introducing the vertical velocity of the object w_p:

$$\frac{dw_p}{dt} = -g \tag{1.15}$$

$$\frac{dz_p}{dt} = w_p. \tag{1.16}$$

The initial condition here is given by setting the initial position and velocity of the object to zero:

$$z_p(0) = 0 \qquad\qquad w_p(0) = 0. \tag{1.17}$$

In this case, it is very easy to obtain the exact solution. Integration of (1.15) using the second initial condition yields

$$w_p(t) = -gt, \tag{1.18}$$

which can be replaced into (1.16) to yields

$$\frac{dz_p}{dt} = -gt. \tag{1.19}$$

Integration of (1.19) with the first initial condition yields

$$z_p(t) = -\frac{gt^2}{2}. \tag{1.20}$$

A numerical solution to the same problem would consist of solving (1.14) or (1.15) and (1.16) numerically. However, we will take a simpler approach just to illustrate the idea of

discretization: instead we want to solve (1.19) with the first initial condition numerically. First we need to "discretize time". This means we will only find the solution at certain times and not a continuous solution that is valid at all times. For simplicity, we can choose these discrete times to be equally spaced by an interval Δt, which we will call the time step. So our solution will only exist at the times $t = t^0, t^1, t^2, t^3, t^4, ...$, where each time is generically given by

$$t^n = n\Delta t, \quad \text{for } n = 0, 1, 2, ... \tag{1.21}$$

Here, $n = 0, 1, ...$ is an index for the discrete points in time. The time discretization is illustrated in Figure 1.1. We can introduce a notation to represent the solution at these discrete times

$$z_p^n = z_p(t^n). \tag{1.22}$$

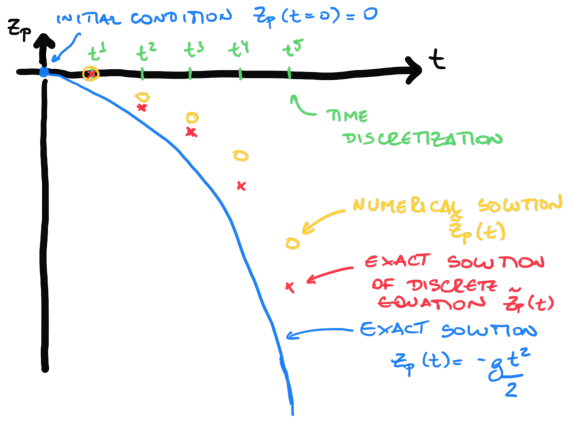

Figure 1.1: Schematic illustrating time discretization and the 3 solutions for the body in free fall.

Now we need to approximate the time derivative in (1.19), so that the solution at a given discrete time can be related to the solution at other discrete times. Recall that the derivative is defined from the limit

$$\frac{dz_p}{dt} \equiv \lim_{\Delta t \to 0} \frac{z_p(t + \Delta t) - z_p(t)}{\Delta t}. \tag{1.23}$$

Suppose that based on the definition above, we try the intuitive approximation for the time derivative with a finite Δt

$$\frac{dz_p}{dt} \approx \frac{z_p(t + \Delta t) - z_p(t)}{\Delta t}. \tag{1.24}$$

Now we can approximate (1.19) as

$$\frac{\tilde{z}_p(t + \Delta t) - \tilde{z}_p(t)}{\Delta t} = -gt \tag{1.25}$$

Here $\tilde{z}_p(t)$ is the solution to equation (1.25). We introduced the tilde to make it clear that $\tilde{z}_p(t)$ is not the same as $z_p(t)$, because we introduced an error in the equation when we

"discretized" the time derivative. This error is called *truncation error*.

In order to solve the problem on a computer, we can now write (1.25) using the discrete notation we introduced:

$$\tilde{z}_p^{n+1} = \tilde{z}_p^n - (g\Delta t)t^n. \tag{1.26}$$

Note that the initial condition in this notation is $z_p^0 = 0$, so using $n = 0$ we can write

$$\tilde{z}_p^1 = \tilde{z}_p^0 - (g\Delta t)t^0. \tag{1.27}$$

So using only the initial condition we can calculate the solution $\tilde{z}_p^1$. We can now repeat the process for $n = 1$, and calculate $\tilde{z}_p^2$ from $\tilde{z}_p^1$. We can continue repeating the process for increasingly large n, advancing the solution in time. This process is called *marching in time*.

There is one more problem we need to consider: when we calculate the solution using marching in time on a computer, we introduce additional convergence and roundoff errors, and the result is an approximate solution of the approximate system $\tilde{\tilde{z}}_p(t)$. The challenge is that when we march the solution in time, errors can accumulate and the solution becomes less accurate as t increases (see Figure 1.1). The critical question we need to answer is the following: how close is $\tilde{\tilde{z}}_p(t)$ to $z_p(t)$? The difference must small for the approximate solution to be useful. In most cases, we do not know $z_p(t)$, so we cannot determine how good the approximate solution is. But if we design "good" discrete approximations to the original ODE, we can at least guarantee that the error is bounded and that it can be reduced by decreasing the time step Δt.

The entire process is represented in the schematic shown in Figure 1.2, where the process of obtaining a numerical solution for an ODE is illustrated. Note that there are two stages for the approximation, and each stage introduces different types of errors. The goal of a well designed numerical method is to ensure that all the errors are small, so that $\tilde{\tilde{z}}_p(t)$ is close enough to $z_p(t)$.

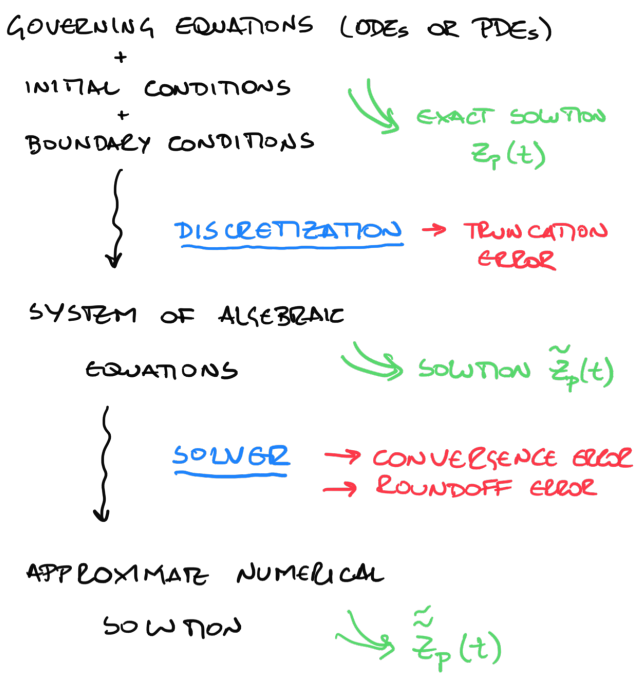

Figure 1.2: Schematic illustrating time discretization and the 3 solutions for the body in free fall.

Chapter 2

Time discretization

Suppose we are interested in the general first-order ODE

$$\frac{du}{dt} = q(u,t) \tag{2.1}$$

defined on $t \geq 0$ with initial condition $u(t=0) = u_0$. Here u is a generic variable (not necessarily the zonal wind speed), and $q(u,t)$ is a generic forcing function. For the simple example of equation (1.19), $q(u,t) = -gt$. Formally, equation (2.1) is an initial value problem, because we only specify conditions at $t = 0$. This is usually the case for time, but can also be the case for space variables. As mentioned in Chapter 1, these problems are usually approached using a marching scheme. While this is quite straightforward in principle, the approach creates a potential problem: arbitrarily small errors (such as roundoff errors) can grow very quickly as the solution is advanced in time, leading to numerical stability problems. When we are developing discretization schemes for initial value problems, we need to ensure that the discrete system does not amplify errors, i.e. that the scheme is *stable*. We will get back to this later.

We already know that the truncation error originates in the approximation of the derivative. Our goal here is to proceed more formally, so that we can characterize this error. Suppose we are marching in time, and given the solution at time t_0, we need to advance the solution to time t_1. We can relate the solution at the two times via Taylor series expansion:

$$\begin{aligned} u(t_1) =& u(t_0) + (t_1 - t_0)\left(\frac{du}{dt}\right)_0 + \frac{(t_1-t_0)^2}{2!}\left(\frac{d^2u}{dt^2}\right)_0 + \frac{(t_1-t_0)^3}{3!}\left(\frac{d^3u}{dt^3}\right)_0 + \\ &+ ... + \frac{(t_1-t_0)^n}{n!}\left(\frac{d^nu}{dt^n}\right)_0 + \mathcal{O}\left[(t_1-t_0)^{n+1}\right]. \end{aligned} \tag{2.2}$$

The symbol $\mathcal{O}$ means "of order" in the sense that $u = 10 + \mathcal{O}(1)$ implies that $u - 10$ results in a number of order 1. Note that this is an exact result, but we will use it develop approximations for the derivatives on the discretized time space.

2.1 The Euler-Forward scheme

For simplicity, we can introduce a time step Δt and for now adopt $t_0 = t$ and $t_1 = t + \Delta t$. In this case we have

$$u(t+\Delta t) = u(t) + \Delta t \left(\frac{du}{dt}\right)_t + \frac{\Delta t^2}{2!}\left(\frac{d^2u}{dt^2}\right)_t + \frac{\Delta t^3}{3!}\left(\frac{d^3u}{dt^3}\right)_t + \mathcal{O}\left(\Delta t^4\right). \tag{2.3}$$

We need an approximation for the first-order derivative, so we can solve for $(du/dt)_t$:

$$\left(\frac{du}{dt}\right)_t = \frac{u(t+\Delta t) - u(t)}{\Delta t} - \frac{\Delta t}{2}\left(\frac{d^2u}{dt^2}\right)_t - \frac{\Delta t^2}{6}\left(\frac{d^3u}{dt^3}\right)_t + \mathcal{O}\left(\Delta t^3\right). \tag{2.4}$$

If we plug this equality into our original ODE (2.1), we have

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} \underbrace{-\frac{\Delta t}{2}\left(\frac{d^2u}{dt^2}\right)_t - \frac{\Delta t^2}{6}\left(\frac{d^3u}{dt^3}\right)_t + \mathcal{O}\left(\Delta t^3\right)}_{\mathcal{E}_T = \mathcal{O}(\Delta t)} = q(u,t). \tag{2.5}$$

Therefore, if we adopt the approximation

$$\frac{u(t+\Delta t) - u(t)}{\Delta t} \approx q(u,t), \tag{2.6}$$

then the truncation error associated with this approximation is

$$\mathcal{E}_T = -\frac{\Delta t}{2}\left(\frac{d^2u}{dt^2}\right)_t - \frac{\Delta t^2}{6}\left(\frac{d^3u}{dt^3}\right)_t + \mathcal{O}\left(\Delta t^3\right) = \mathcal{O}\left(\Delta t\right). \tag{2.7}$$

In practice we write the approximate equation using the discrete scheme based on the time index n as

$$\frac{u^{n+1} - u^n}{\Delta t} = q(u^n, t^n), \tag{2.8}$$

which can be arranged into a recursive algebraic equation for marching in time

$$u^{n+1} = u^n + \Delta t q(u^n, t^n). \tag{2.9}$$

Note how this can be easily implemented into a loop that calculates the solution at each time based on the solution at the previous time. So given the initial condition u^0 we can calculate u^1, and then from u^1 we can calculate u^2, and so on.

The approximation adopted here

$$\left(\frac{du}{dt}\right)^n \approx \frac{u^{n+1} - u^n}{\Delta t} \tag{2.10}$$

is called the Euler-Forward (EF) scheme because the derivative at the time n is calculated using the solution at time n and the solution at a forward time $(n+1)$. This scheme is said to be first-order accurate in Δt because the smallest exponent exponent of Δt in the truncation error is one. This means that if we divide Δt by 2, we divide the error (approximately) by 2 as well as the other terms in the error are divided by 4, 8, etc. So the term with the smallest power of Δt is the one that decays slowest as we reduce the time step.

Note that a discretization scheme is termed consistent if

$$\lim_{\Delta t \to \infty} \mathcal{E}_T = 0. \tag{2.11}$$

As we can see, Euler-Forward is a consistent scheme. As we will see later, this is very important.

2.2 Other simple schemes

2.2.1 The Euler-Backward scheme

Note that an alternative would be to expand $u(t - \Delta t)$ in terms of $u(t)$. This is equivalent to adopting $t_0 = t$ and $t_1 = t - \Delta t$ in equation (2.2), and it results in the approximation

$$\left(\frac{du}{dt}\right)^n \approx \frac{u^n - u^{n-1}}{\Delta t}. \tag{2.12}$$

This scheme is called the Euler-Backward (EB) scheme because the derivative at time n is calculated using the backward time $(n-1)$. The truncation error is

$$\mathcal{E}_T = +\frac{\Delta t}{2}\left(\frac{d^2u}{dt^2}\right)_t - \frac{\Delta t^2}{6}\left(\frac{d^3u}{dt^3}\right)_t + \mathcal{O}\left(\Delta t^3\right) = \mathcal{O}\left(\Delta t\right). \tag{2.13}$$

This scheme is also first-order, and even though at first look it looks quite similar to the Euler-Forward scheme, the key difference is apparent when we write the approximate equation

$$\frac{u^n - u^{n-1}}{\Delta t} = q(u^n, t^n). \tag{2.14}$$

Note that we can replace n by $(n+1)$ and $(n-1)$ by n without changing the equation. In this case the recursive equation becomes

$$u^{n+1} = u^n + \Delta t q(u^{n+1}, t^{n+1}). \tag{2.15}$$

Note that u^{n+1} appears on the right-hand side of the equation, which complicates enormously the solution if the function $q(u,t)$ is not very simple. Schemes in which u^{n+1} appears on the

right-hand side are termed *implicit schemes*, while schemes such as the Euler-Forward scheme are called *explicit schemes*. Implicit schemes are harder to implement and computationally more expensive, but they also have much nicer properties (e.g. when it comes to numerical stability and amplification of small errors).

2.2.2 Leapfrog and Crank-Nicolson schemes

Two possible combinations of EF and EB schemes are also commonly used. The first is obtained by simply averaging (2.10) and (2.12). This is motivated by noting that the leading-order term in the truncation error has opposite signs in these two methods (see 2.7 and 2.13) and it results in

$$\left(\frac{du}{dt}\right)^n \approx \frac{1}{2}\left[\frac{u^{n+1}-u^n}{\Delta t}+\frac{u^n-u^{n-1}}{\Delta t}\right]=\frac{u^{n+1}-u^{n-1}}{2\Delta t}. \tag{2.16}$$

This is the Leapfrog scheme (also known as mid-point scheme) and it uses points on both side of u^n to estimate the derivative. The algebraic recursive equation becomes

$$u^{n+1}=u^{n-1}+2\Delta t q(u^n,t^n). \tag{2.17}$$

and the truncation error is

$$\mathcal{E}_T=-\frac{\Delta t^2}{6}\left(\frac{d^3u}{dt^3}\right)_t+\mathcal{O}\left(\Delta t^3\right)=\mathcal{O}\left(\Delta t^2\right). \tag{2.18}$$

So this is an explicit second-order scheme, but it does require us to store information from a previous time step. Even though this seems quite simple at first, for large simulations this storage requirement may be non-trivial.

The Crank-Nicolson scheme (also called trapezoidal rule) is instead given by

$$u^{n+1}=u^n+\frac{\Delta t}{2}\left[q(u^n,t^n)+q(u^{n+1},t^{n+1})\right] \tag{2.19}$$

This is also a second-order scheme, except that, unlike the leapfrog scheme, it is implicit.

2.3 Other approaches

There are two large groups of advanced approaches that are frequently used in practice. Multi-step methods use information from several time-steps to estimate the derivative, and the order of accuracy increases with the number of additional time-steps involved in the calculation. These include a family of explicit methods called Adams-Bashforth schemes and a family of implicit methods called Adams-Moulton schemes. Multi-stage methods (also called predictor-corrector methods) achieve higher order of accuracy by splitting one single

time step into several stages. These include the family of Runge-Kutta (RK) methods. We will explore at least a few additional methods later in the course.

2.4 Final comments

Note that for practical purposes all we need to do is to write a code that uses the appropriate recursive algebraic equations to march the solution in time. These recursive equations look very similar (see equations 2.9, 2.15, 2.17, and 2.19), but result in methods with very different properties. Implementing explicit schemes is very straightforward, but implicit schemes are much more difficult and the implementation actually depends on the specific form of $q(u,t)$.

Finally, to help illustrate the use of these methods, we will apply this to the simple ODE example from the previous chapter and given by equation (1.19). In this case, our variable is z_p instead of u, and the forcing function is $q(u,t) = -gt$ so that $q(u^n, t^n) = -gt^n = -gn\Delta t$ (note that it does not depend on the variable z_p, which simplifies the problem considerably). In this case, the recursive equation for Euler-Forward (2.9) becomes

$$z_p^{n+1} = z_p^n + \Delta t(-gn\Delta t). \tag{2.20}$$

Similarly, for Euler-Backward (2.15), Leapfrog (2.17), and Crank-Nicolson (2.19) we have, respectively,

$$z_p^{n+1} = z_p^n + \Delta t[-g(n+1)\Delta t], \tag{2.21}$$

$$z_p^{n+1} = z_p^{n-1} + 2\Delta t(-gn\Delta t), \tag{2.22}$$

and

$$z_p^{n+1} = z_p^n + \frac{\Delta t}{2}\left[-gn\Delta t - g(n+1)\Delta t\right]. \tag{2.23}$$

Chapter 3

PDEs and the finite-difference method

As we have discussed before, the primitive equations are too complicated to serve as a framework to develop and study numerical schemes, so we will focus on simple equations that contain important features of the primitive equations. Even though the primitive equations are nonlinear in nature, we will start by studying different types of linear equations.

3.1 Classification of linear second-order PDEs

As we will see, the choice of discretization scheme depends on the properties of the PDE. For this purpose, there are 3 main types of equations: (i) hyperbolic, (ii) parabolic, and (iii) elliptic equations. We will very briefly describe these types here.

3.1.1 Hyperbolic equations

The most common example here is the advection equation (also called one-way wave equation):

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0. \tag{3.1}$$

Here c is the advection or propagation velocity. Note that this equation requires one initial condition and one boundary condition, and it describes the advection or translation of the solution without change in amplitude or shape as illustrated in Figure 3.1.

Note that even though the advection equation is not a second-order PDE, it is closely related to the wave equation that describes bidirectional propagation of information:

$$\frac{\partial^2 u}{\partial t^2} - c^2\frac{\partial^2 u}{\partial x^2} = 0. \tag{3.2}$$

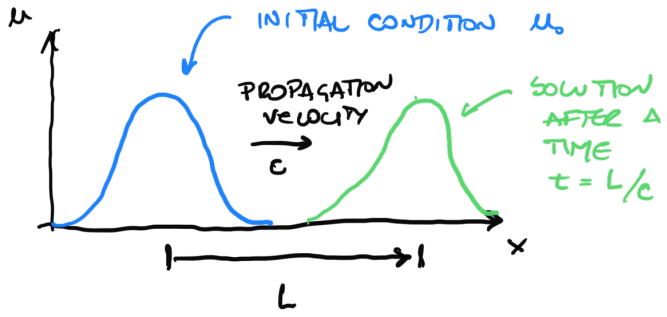

Figure 3.1: Schematic illustrating solution to the advection equation.

3.1.2 Parabolic equations

The most common example of a parabolic equation is the diffusion equation:

$$\frac{\partial u}{\partial t} = k\frac{\partial^2 u}{\partial x^2}. \tag{3.3}$$

Here k is the diffusivity. This equation requires one initial condition and two boundary conditions (one on each side of the domain), and it describes change in amplitude and shape without translation as illustrated in Figure 3.2.

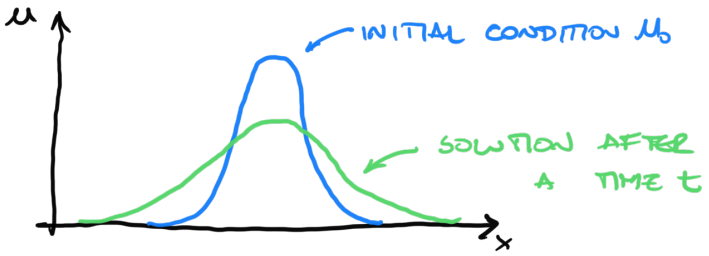

Figure 3.2: Schematic illustrating solution to the diffusion equation.

3.1.3 Elliptic equations

The classic example here is Poisson's equation in 2 spatial dimensions:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = F(x, y). \tag{3.4}$$

This equation requires two boundary conditions in x and 2 boundary conditions in y, specifying some property of the solution across the entire boundary of the domain. Poisson's equations are diagnostic equations (i.e., no time evolution) and usually describe some equilibrium configuration. One important characteristic of Poisson's equation is that the value of F at any given point influences the solution everywhere in the domain.

3.2 The finite-difference method

In the next few weeks, we will solve numerically all 3 types of equations above. For now, we start by discussing how we discretize the spatial coordinate x. Suppose we want to solve a PDE on a domain $0 \leq x \leq L_x$ with appropriate boundary conditions. We first divide the domain $[0; L_x]$ using N_x grid points, which will result in $(N_x - 1)$ equal segments of length

$$\Delta x = \frac{L_x}{N_x - 1}. \tag{3.5}$$

The length Δx is termed mesh size or grid size, and the resulting grid is illustrated in Figure 3.3.

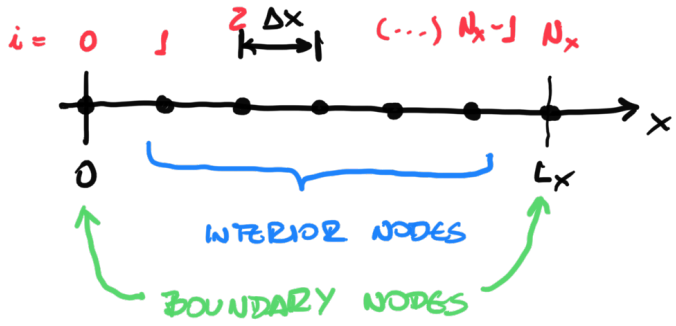

Figure 3.3: Schematic illustrating finite-difference discretization.

Just as we introduced the index n to indicate the discrete times in which the solution exists, we will use i for grid points in the x direction. Our choice of discretization results in grid points at the location given by

$$x_i = (i-1)\Delta x \quad \text{for } i = 1, ..., N_x. \tag{3.6}$$

We will use this numbering scheme for our lecture notes. Note however that when writing code using languages that start arrays on the index 0 (such as Python and C++), it may be more appropriate to use the equivalent notation

$$x_i = (i)\Delta x \quad \text{for } i = 0, ..., (N_x - 1). \tag{3.7}$$

As shown in Figure 3.3, the nodes $i = 1$ and $i = N_x$ are special nodes termed *boundary nodes*, as they are used to impose boundary conditions and usually require special treatment in the code. The other nodes are termed *interior nodes*.

Also note that for a given grid spacing of Δx, the smallest resolvable feature has a wavelength of $\lambda_{\min} = 2\Delta x$ as illustrated in the Figure 3.4. For our purposes here, we will always try to choose a mesh size small enough that all the features in the solution are well

represented. This is only possible if the PDEs of interest are linear, as we will see later.

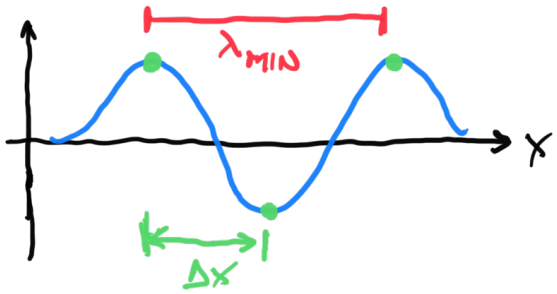

Figure 3.4: Schematic illustrating smallest resolvable wavelength in the finite-difference discretization.

Next we need to develop approximations for the spatial derivatives. As we have done for the time derivatives, the approach relies on Taylor series expansions of $u(x)$ in the vicinity of $x = x_0$:

$$u(x) = u(x_0) + (x - x_0)\left(\frac{du}{dx}\right)_0 + \frac{(x - x_0)^2}{2!}\left(\frac{d^2u}{dx^2}\right)_0 + \frac{(x - x_0)^3}{3!}\left(\frac{d^3u}{dx^3}\right)_0 + \mathcal{O}\left[(x - x_0)^4\right]. \tag{3.8}$$

3.2.1 First-order derivatives

Suppose we are writing the approximate equation at the nodal point with index i, and we need to discretize the first-order derivative at this nodal point (see Figure 3.6). For now, we will use u_i and u_{i+1} in the approximation, so we set $x_0 = x_i$, $x = x_{i+1}$, and $(x - x_0) = (x_{i+1} - x_i) = \Delta x$ in the Taylor series:

$$u_{i+1} = u_i + \Delta x\left(\frac{du}{dx}\right)_i + \frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i + \frac{\Delta x^3}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right). \tag{3.9}$$

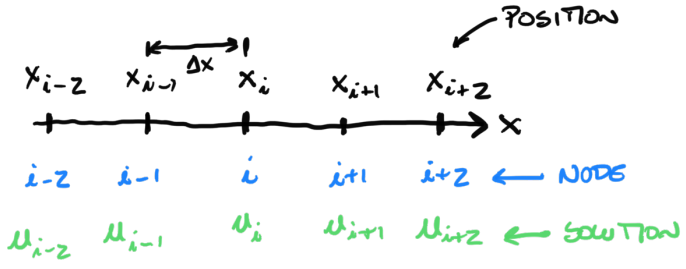

Figure 3.5: Finite-difference discretization for first-order derivative.

Now we solve for the first-order derivative:

$$\left(\frac{du}{dx}\right)_i = \frac{u_{i+1}-u_i}{\Delta x} \underbrace{-\frac{\Delta x}{2}\left(\frac{d^2u}{dx^2}\right)_i - \frac{\Delta x^2}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^3\right)}_{\mathcal{O}(\Delta x)}. \tag{3.10}$$

This results in the approximation

$$\left(\frac{du}{dx}\right)_i \approx \frac{u_{i+1}-u_i}{\Delta x}. \tag{3.11}$$

This is the forward-difference scheme for the first-order derivative (this is equivalent to Euler-Forward for the time discretization). Note that the truncation error involved in this approximation is given by

$$\mathcal{E}_T = \frac{\Delta x}{2}\left(\frac{d^2u}{dx^2}\right)_i + \frac{\Delta x^2}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^3\right) = \mathcal{O}\left(\Delta x\right). \tag{3.12}$$

Because the truncation error is $\mathcal{O}\left(\Delta x\right)$, the approximation is said to be first-order in Δx. Note that this approximation is also consistent, as clearly

$$\lim_{\Delta x \to 0} \mathcal{E}_T = 0. \tag{3.13}$$

Note that, if instead of using u_{i+1} we had chosen u_{i-1}, we would have set $x_0 = x_i$, $x = x_{i-1}$, and $(x - x_0) = (x_{i-1} - x_i) = -\Delta x$, resulting in

$$u_{i+1} = u_i - \Delta x\left(\frac{du}{dx}\right)_i + \frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i - \frac{\Delta x^3}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right), \tag{3.14}$$

which yields

$$\left(\frac{du}{dx}\right)_i \approx \frac{u_i - u_{i-1}}{\Delta x}. \tag{3.15}$$

This is the backward-difference scheme, which is also first-order in Δx.

At this point we are interested in obtaining schemes with better convergence rate than the two first-order schemes above. There is a simple general method to do so. We start by choosing the desired *stencil*, which is the set of points that will be used in the approximation of the derivative. Suppose we choose the points $(i-1)$, i, and $(i+1)$.

Now we write the derivative as a linear combination of the solution at these chosen points

$$\left(\frac{du}{dx}\right)_i = au_{i-1} + bu_i + cu_{i+1} + \mathcal{O}\left(\Delta x^m\right), \tag{3.16}$$

where a, b, c, and m are unknown. Next we replace u_{i-1} and u_{i+1} by Taylor series expansions,

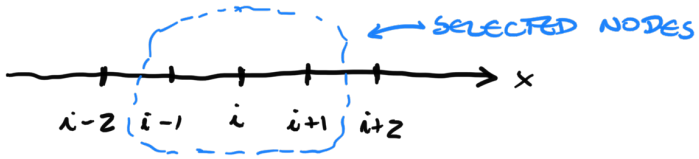

Figure 3.6: 1D stencil for first-order derivative.

so that the entire expression is written in terms of u_i:

$$\begin{aligned} LC =& au_{i-1} + bu_i + cu_{i+1} \\ =& a\left[u_i - \Delta x\left(\frac{du}{dx}\right)_i + \frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i - \frac{\Delta x^3}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right)\right] + bu_i + \\ &+ c\left[u_i + \Delta x\left(\frac{du}{dx}\right)_i + \frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i + \frac{\Delta x^3}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right)\right]. \end{aligned} \tag{3.17}$$

Rearranging this equation results in:

$$\begin{aligned} LC =& (a+b+c)u_i + (-a+c)\Delta x\left(\frac{du}{dx}\right)_i + (a+c)\frac{\Delta x^2}{2}\left(\frac{d^2u}{dx^2}\right)_i + \\ &+ (-a+c)\frac{\Delta x^3}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right). \end{aligned} \tag{3.18}$$

Clearly, in order to satisfy the consistency requirement

$$\lim_{\Delta x \to 0} LC = \left(\frac{du}{dx}\right)_i, \tag{3.19}$$

we need to force the first term on the right-hand side of (3.18) to be zero and the coefficient of the second term to be identically one. Thus, we impose

$$\begin{cases} (a+b+c) = 0 \\ (-a+c)\Delta x = 1 \end{cases}, \tag{3.20}$$

which leaves one more constraint to be freely chosen. The obvious choice is to eliminate the leading order error remaining in the approximation, which in this case is the term involving the second-order derivative. This requires $(a+c)\Delta x^2/2 = 0$. Thus, by solving the linear system we obtain:

$$\begin{cases} a+b+c=0 \\ -a+c = 1/\Delta x \\ a+c=0 \end{cases} \Rightarrow \begin{cases} a = -1/(2\Delta x) \\ b = 0 \\ c = 1/(2\Delta x) \end{cases}. \tag{3.21}$$

With this solution, the equation for the first-order derivative becomes

$$\left(\frac{du}{dx}\right)_i = \frac{1}{2\Delta x}\left(u_{i+1} - u_{i-1}\right) - \frac{\Delta x^2}{6}\left(\frac{d^3u}{dx^3}\right)_i + \mathcal{O}\left(\Delta x^4\right), \tag{3.22}$$

where we used the fact that the term $\mathcal{O}\left(\Delta x^3\right)$ (as well as any other odd power of Δx) will cancel out as well. The final approximation is

$$\left(\frac{du}{dx}\right)_i \approx \frac{1}{2\Delta x}\left(u_{i+1} - u_{i-1}\right). \tag{3.23}$$

This is the centered-difference scheme for the first-order derivative, which is second-order in Δx.

Note that just because this scheme is second-order in Δx does not necessarily imply that it is more accurate than a first-order scheme. The error of the centered-difference scheme may be larger than that of the forward- or backward-difference scheme for a given problem and a given Δx (specially for coarser grids). However, the error in a second-order scheme decreases faster as the grid size is reduced and if the solution converges to the exact solution, it does so faster than for a first-order scheme.

At this point, it should be clear that as we include more points in the stencil, we have more degrees of freedom, which at the end allow us to select additional high-order terms to be eliminated from the truncation error. Using this approach we can construct higher order schemes. Note that we can also construct asymmetric schemes such as

$$\left(\frac{du}{dx}\right)_i = au_i + bu_{i+1} + cu_{i+2} + \mathcal{O}\left(\Delta x^m\right), \tag{3.24}$$

which can be particularly useful near boundaries and in the discretization of Neumann boundary conditions.

3.2.2 Second-order derivatives

The best way to proceed here is to use the general method outlined in the previous section and write the second-order derivative as

$$\left(\frac{d^2u}{dx^2}\right)_i = au_{i-1} + bu_i + cu_{i+1} + \mathcal{O}\left(\Delta x^m\right). \tag{3.25}$$

Because we used the same points as before, Equation (3.18) is still the same. But now, the consistency requirements are

$$\begin{cases} a+b+c=0 \\ -a+c=0 \\ a+c=2/\Delta x^2 \end{cases} \Rightarrow \begin{cases} a=1/\Delta x^2 \\ b=-2/\Delta x^2 \\ c=1/\Delta x^2 \end{cases} . \tag{3.26}$$

The second order derivative becomes

$$\left(\frac{d^2u}{dx^2}\right)_i = \frac{1}{\Delta x^2}\left(u_{i-1}-2u_i+u_{i+1}\right) - \frac{\Delta x^2}{12}\left(\frac{d^4u}{dx^4}\right)_i + \mathcal{O}\left(\Delta x^4\right), \tag{3.27}$$

which yields the centered-difference scheme for the second-order derivative given by

$$\left(\frac{d^2u}{dx^2}\right)_i \approx \frac{1}{\Delta x^2}\left(u_{i-1}-2u_i+u_{i+1}\right). \tag{3.28}$$

Note that this scheme is already second-order in Δx.

3.2.3 Final comments

You can apply the approach outlined above to approximate derivatives of any order using any number of grid points you like, in a symmetric or asymmetric distribution around the point where the derivative is needed. If you do not want to do all the work multiple Taylor series expansions, check out this on-line calculator developed by Cameron Taylor at MIT that gives you the coefficients for any approximation: `https://web.media.mit.edu/~crtaylor/calculator.html`

Chapter 4

The advection equation

We will start with a simple one-dimensional linear advection equation

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0, \tag{4.1}$$

defined for $0 \leq x \leq L_x$ and $t \geq 0$. Here we will take $v > 0$ as the constant advection velocity and we will consider an initial condition $u(x, t = 0) = u_0(x)$. The boundary condition is already problematic, so we will consider the simplest case of a periodic boundary condition on the domain $[0; L_x]$, which essentially means $u(x = 0, t) = u(x = L_x, t)$. As we will see, the first-order derivative in x will cause trouble.

4.1 Discretization using forward in time and backward in space scheme (FTBS)

Because we need to combine spatial and time discretization, we will discretize $u(x, t)$ as u_i^n, where by definition the index i will refer to the spatial discretization and the index n to the time discretization. For clarity, we can write

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v\left(\frac{\partial u}{\partial x}\right)_i^n = 0, \tag{4.2}$$

to emphasize that we must approximate both derivatives at the same position and at the same instant in time. If we use Euler-Forward for the time derivative and backward in space for the spatial derivative we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + v\frac{u_i^n - u_{i-1}^n}{\Delta x} = 0. \tag{4.3}$$

This is the discrete equation, which can be written in recursive form as

$$u_i^{n+1} = u_i^n - \frac{v\Delta t}{\Delta x}(u_i^n - u_{i-1}^n). \tag{4.4}$$

Here it is useful to define the *Courant number*

$$\mathcal{C} = \frac{v\Delta t}{\Delta x} \tag{4.5}$$

and write the recursive equation as

$$u_i^{n+1} = (1 - \mathcal{C})u_i^n + \mathcal{C}u_{i-1}^n. \tag{4.6}$$

As we will see later, the Courant number plays a very important role in the numerical solution. The recursive equation (4.6) is the key result for numerical implementation in your code. However, before we implement this scheme to solve the advection equation, we need to check for consistency, accuracy, and other possible problems.

4.1.1 Consistency and accuracy

The formal treatment of accuracy (actually rate of convergence) follows the analysis using Taylor series expansions in the previous chapters. For simplicity, we will keep only the leading-order error term in each series. For the time discretization we have

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \frac{u_i^{n+1} - u_i^n}{\Delta t} - \frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n + \mathcal{O}\left(\Delta t^2\right), \tag{4.7}$$

and for the spatial discretization we have

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{u_i^n - u_{i-1}^n}{\Delta x} + \frac{\Delta x}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + \mathcal{O}\left(\Delta x^2\right). \tag{4.8}$$

If we replace these two approximations into the discrete equation (4.3), we obtain

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v\left(\frac{\partial u}{\partial x}\right)_i^n \underbrace{+\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n - v\frac{\Delta x}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + \mathcal{O}\left(\Delta t^2, \Delta x^2\right)}_{\mathcal{E}_T = \mathcal{O}(\Delta t, \Delta x)} = 0 \tag{4.9}$$

This equation is sometimes called the *modified equation*, and it is the true equation that corresponds to the discrete equation (4.3). Note that the first two terms are the original advection equation, and the other terms correspond to the truncation error introduced by

the discretization process. Note that clearly the scheme is consistent as clearly

$$\lim_{\Delta t, \Delta x \to 0} \mathcal{E}_T = 0. \tag{4.10}$$

Note also that the scheme is first-order in Δt and first-order in Δx.

It is difficult to interpret the truncation error because it combines partial derivatives in space and time. We can significantly improve our understanding by using the original PDE to eliminate the time derivative. Note that we can use the original PDE to write

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x}. \tag{4.11}$$

Thus, the second order derivative in time that appears in the truncation error can be written as

$$\frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial t}\left(\frac{\partial u}{\partial t}\right) = \frac{\partial}{\partial t}\left(-v\frac{\partial u}{\partial x}\right) = -v\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right) = -v\frac{\partial}{\partial x}\left(-v\frac{\partial u}{\partial x}\right) = v^2\frac{\partial^2 u}{\partial x^2}. \tag{4.12}$$

Therefore we have that

$$\frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_i^n = \frac{v^2\Delta t}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n \tag{4.13}$$

and we can re-write the modified equation (4.9) as

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{1}{2}(v\Delta x - v^2\Delta t)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + \mathcal{O}\left(\Delta t^2, \Delta x^2\right). \tag{4.14}$$

Note that the leading order error in our approximation introduces an artificial diffusive behavior. Our original equation was the advection equation, but our approximate finite-difference scheme is equivalent to an equation that includes both advection and diffusion. The *numerical diffusivity* introduced by the discretization is

$$K_{\text{eff}} = \frac{1}{2}(v\Delta x - v^2\Delta t) = \frac{1}{2}v\Delta x(1-\mathcal{C}). \tag{4.15}$$

This expression for K_{eff} delineates three possible scenarios

- If $\mathcal{C} < 1$ we have a diffusive scheme with numerical diffusion given by (4.15). We can anticipate a stable scheme with artificial diffusion, meaning that as our solution moves in space its amplitude will be reduced and its shape will change.

- If $\mathcal{C} = 1$ the diffusion term is identically zero, the leading order error is no longer diffusion (more on this later) and the scheme is $\mathcal{O}\left(\Delta t^2, \Delta x^2\right)$. So we gain accuracy if we can always use $\mathcal{C} = 1$.

- If $\mathcal{C} > 1$ we have negative diffusion that results in an unstable scheme (the amplitude

of the solution will artificially grow in time; more on this very soon).

If you look carefully at equations (4.14) and (4.15) you will notice that the positive numerical diffusion comes from the spatial discretization, and the negative diffusion comes from the time discretization. The criterion $\mathcal{C} \leq 1$ is simply requiring the diffusion introduced by the spatial discretization to be larger in magnitude than that of the time discretization.

As a side note, it is interesting to note what happens if we replace the backward in space by the forward in space scheme. This is the FTFS scheme, and in this case the truncation error is

$$\mathcal{E}_T = \frac{1}{2} v \Delta x (-1 - \mathcal{C}) \left(\frac{\partial^2 u}{\partial x^2} \right)_i^n + \mathcal{O}\left(\Delta t^2, \Delta x^2\right). \tag{4.16}$$

Now both temporal and spatial discretizations produce negative diffusion, and the scheme always has negative diffusion (it is always unstable). However, it is interesting to note that this scheme would work for the case $v < 0$, while our original FTBS scheme would always be unstable for $v < 0$. The conclusion is that if we use a one-sided approximation for the spatial derivative, this works when the points we choose for the stencil is on the direction the information is coming from, as illustrated in Fig. 4.1.

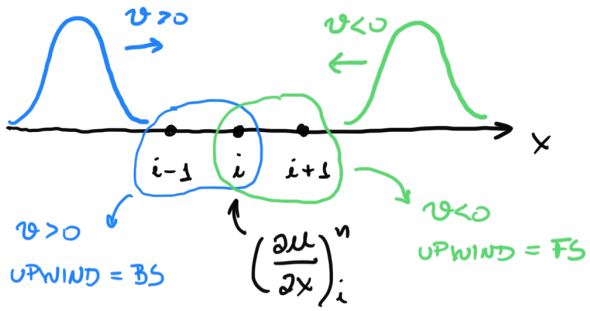

Figure 4.1: Schematic illustrating the concept of upwind schemes.

This is the source of the idea of *upwind schemes*. Upwind schemes are based on one-sided (asymmetric) differences in space taking into account the direction of propagation: backward in space for $v > 0$ and forward in space for $v < 0$. In other words, they use information from the direction information is coming from. These schemes are very popular, even when the advective velocity is part of the solution (i.e., it can change sign in space and time), as in the Navier-Stokes equations. We will discuss this later.

For now, our analysis of the modified equation suggests that the scheme is stable if

$$\mathcal{C} = \frac{v \Delta t}{\Delta x} \leq 1. \tag{4.17}$$

This is a fairly general condition for explicit schemes applied to hyperbolic PDEs (such as the advection and wave equations), called the Courant-Friedrichs-Lewy (CFL) condition. The physical interpretation of this condition is that the distance traveled by the solution in one time step (equal to $v\Delta t$) has to be smaller than the grid size (i.e., $v\Delta t \leq \Delta x$).

4.1.2 Stability analysis

By definition, a numerical method is stable if it does not amplify errors that appear in the course of the numerical solution. We have a rigorous approach to analyze stability of numerical methods when applied to linear equations called the von Neumann stability analysis. Here we will use a simplified version of this method.

The stability analysis starts from the basic idea that the solution to the linear PDE of interest can be represented using Fourier series, and so can the error in the solution itself. Because the PDE is linear, different wavenumbers do not interact with each other, so we can study the behavior of one generic Fourier mode, and then focus on the consequences of the discrete equation for the worst case scenario (i.e., the Fourier mode for which the error is amplified more quickly).

Suppose we stat by analyzing the behavior of one single mode of the Fourier series given by

$$u(x,t) = Ae^{\mathrm{i}(kx-\omega t)}, \tag{4.18}$$

where $\mathrm{i} = \sqrt{-1}$ is the imaginary number, A is the amplitude of the solution, k is the wavenumber of the Fourier mode, and ω is the angular frequency. In general, the angular frequency ω can be a complex number. Note that the relationship between ω and k, termed *dispersion relation*, is imposed by the PDE. If we replace (4.18) into the advection equation (4.2) we have

$$-\omega \mathrm{i} A e^{\mathrm{i}(kx-\omega t)} + vk\mathrm{i} A e^{\mathrm{i}(kx-\omega t)} = 0, \tag{4.19}$$

which yields the dispersion relation for the advection equation

$$\omega = vk. \tag{4.20}$$

Because v and k are real number, then in the case of the advection equation ω is also real. Note that the phase speed (which is the speed at which information is propagated in space) is given by

$$c_{ph} = \frac{\omega}{k} = v. \tag{4.21}$$

As expected, the mode (4.18) propagates with constant phase speed v.

However, for stability we are really interested in the changes in the solution between a generic time t and a later time $(t + n\Delta t)$. Note that based on the mode (4.18) we can write

the solution at $(t + n\Delta t)$ as

$$\begin{aligned} u(x, t + n\Delta t) &= Ae^{\mathrm{i}[kx-\omega(t+n\Delta t)]} \\ &= Ae^{\mathrm{i}(kx-\omega t)}e^{-\mathrm{i}\omega n\Delta t} \\ &= u(x, t)e^{-\mathrm{i}\omega n\Delta t} \end{aligned} \tag{4.22}$$

To characterize the change in the solution we define the *changing function* based on the changes in one single time step (i.e., for $n = 1$)

$$\mu = \frac{u(x, t + \Delta t)}{u(x, t)} = e^{-\mathrm{i}\omega\Delta t}, \tag{4.23}$$

so that

$$u(x, t + n\Delta t) = \mu^n u(x, t). \tag{4.24}$$

In general ω is a complex number that can be written in standard notation as $\omega = \omega_R + \mathrm{i}\omega_I$, where ω_R and ω_I are the real and imaginary parts. Thus, the general changing function can be written as

$$\mu = e^{\omega_I\Delta t}e^{-\mathrm{i}\omega_R\Delta t} = e^{\omega_I\Delta t}\left[\cos(\omega_R\Delta t) - \mathrm{i}\sin(\omega_R\Delta t)\right]. \tag{4.25}$$

Note that the first term, associated with the imaginary part of ω, is the magnitude of μ, and it indicates an exponential growth or decay of the solution in time. On the other hand, the second term, associated with the real part of ω, represents oscillations in time.

This becomes clearer if we recall that a general complex number $\mu = a + \mathrm{i}b$ can be written also in the polar form $\mu = |\mu|e^{\mathrm{i}\theta}$, where $|\mu| = \sqrt{a^2 + b^2}$ is the magnitude of μ and $\theta = \arctan(b/a)$ is its phase. For the general result above we have that the magnitude is given by

$$|\mu| = e^{\omega_I\Delta t}, \tag{4.26}$$

and the phase is

$$\theta = -\omega_R\Delta t. \tag{4.27}$$

The term $|\mu|$ is associated with the changes in magnitude, and we will call this the *amplification factor*. As we have concluded from the dispersion relation, in the case of the advection equation the true ω is real. Thus, for the exact solution $\omega_I = 0$ and $|\mu| = 1$. Thus, for our exact solution the changing function is given by

$$\mu = e^{-\mathrm{i}\omega_R\Delta t} = \left[\cos(\omega_R\Delta t) - \mathrm{i}\sin(\omega_R\Delta t)\right]. \tag{4.28}$$

This changing function characterizes a solution that propagates in space without changes in amplitude, confirming our statement about the advection equation made in Chapter 3.

The main goal of the stability analysis is to determine the amplification factor for the

discrete equation. In particular, we can state that if $|\mu| \leq 1$ for the discrete equation, then the scheme is stable. Conversely, the scheme is unstable if $|\mu| > 1$. The main problem with the discrete solution is that the discrete dispersion relation is different, and thus the amplification factor is also different. So our next goal is to determine the dispersion relation for the discrete equation. We start by writing the discrete Fourier mode (4.18) as

$$u_i^n = Ae^{\mathrm{i}(ki\Delta x - \omega n\Delta t)}. \tag{4.29}$$

Note that it is easy to relate solutions at different times and different position. As an example, we have that

$$u_{i-1}^{n+1} = u_i^n e^{\mathrm{i}(-k\Delta x - \omega\Delta t)}. \tag{4.30}$$

For the FTBS scheme, we have that the recursive equation is given by (4.6), repeated below:

$$u_i^{n+1} = (1-\mathcal{C})u_i^n + \mathcal{C}u_{i-1}^n.$$

If we apply the strategy illustrated in (4.30) to write all the terms of this recursive equation using u_i^n we have

$$u_i^n e^{\mathrm{i}(-\omega\Delta t)} = (1-\mathcal{C})u_i^n + \mathcal{C}u_i^n e^{-\mathrm{i}k\Delta x}. \tag{4.31}$$

Thus, the discrete dispersion relation is given by

$$e^{\mathrm{i}(-\omega\Delta t)} = 1 - \mathcal{C}(1 - e^{-\mathrm{i}k\Delta x}). \tag{4.32}$$

It is more convenient to write this result in terms of the changing function

$$\mu = 1 - \mathcal{C}(1 - e^{-\mathrm{i}k\Delta x}) = 1 - \mathcal{C}[1 - \cos(k\Delta x) + \mathrm{i}\sin(k\Delta x)]. \tag{4.33}$$

Note that for stability purposes we are interested only in the magnitude of μ, so we can write

$$|\mu|^2 = 1 + 2\mathcal{C}(1-\mathcal{C})[\cos(k\Delta x) - 1]. \tag{4.34}$$

The value of the amplification factor is different for each wavenumber k. However, we know for sure that $[\cos(k\Delta x) - 1] \leq 0$. In order for the condition of stability $|\mu|^2 \leq 1$ to be satisfied for all values of k, we need to ensure that

$$2\mathcal{C}(1-\mathcal{C}) \geq 0. \tag{4.35}$$

This implies that the criterion for stability of the FTBS scheme applied to the 1D linear advection equation is

$$0 \leq \mathcal{C} \leq 1 \quad \text{or} \quad 0 \leq \frac{v\Delta t}{\Delta x} \leq 1 \tag{4.36}$$

This is in fact the most correct form of the CFL condition. As we have discussed, $\mathcal{C} < 0$ is only possible if $v < 0$, and this case the FTBS scheme is no longer an upwind scheme.

4.1.3 Amplitude and phase errors

We used the discrete dispersion relation to assess stability. This is the core of von Neumann's stability analysis. However, there is much more information in the discrete dispersion relation (4.33). It is convenient to separate the errors in the numerical solution into amplitude errors and phase errors. Amplitude errors are associated with the magnitude of the solution and we can define a relative amplitude error as

$$R_{amp} = \frac{|\mu|^2}{|\mu_{ex}|^2}. \tag{4.37}$$

Here $|\mu_{ex}|^2 = 1$ is the amplitude change obtained from the dispersion relation of the PDE, and $|\mu|^2$ is given by (4.34) so that we have

$$R_{amp} = \frac{|\mu|^2}{|\mu_{ex}|^2} = 1 + 2\mathcal{C}(1-\mathcal{C})[\cos(k\Delta x) - 1]. \tag{4.38}$$

This error is a function of $\mathcal{C}$ and $k\Delta x$ (or equivalently $\lambda_x \Delta x$). Let's first look at the effects of the Courant number on the error. We already know that if $\mathcal{C} = 1$ there is not amplitude error, so this is the ideal scenario. We also know that if $\mathcal{C} > 1$ the scheme is unstable and the solution will blow-up, so this is no longer a relevant case. For other values of $\mathcal{C}$, we get $R_{amp} < 1$, indicating that the amplitude of the numerical solution will be smaller than that of the exact solution (this is the effect of the numerical diffusion we talked about).

As a side note, from (4.24) we can see that the amplitude A of the numerical solution will decay by a factor of $|\mu|$ each time step (in the specific case of the advection equation, this decay is completely due to error). This means that the decay in amplitude can be written as

$$A(n) = A_0 e^{-(1-|\mu|)n}. \tag{4.39}$$

Here A_0 is the initial amplitude and n is the time index so that $t = n\Delta t$ is the total time. Thus, we anticipate an exponential decay in time as illustrated in the Fig. 4.2. Note that $|\mu|$ can be estimated from the numerical solution by plotting A/A_0 against iteration number n.

Going back to (4.38), we now look at the effect of $k\Delta x$ on the error. Suppose we fix $\mathcal{C} \neq 1$. In this case $R_{amp} \neq 1$ and the error will be different for different wavenumbers in the solution. In particular, the error will be directly proportional to the magnitude of the term in the square brackets. Note that the smallest wave length we can resolve is $\lambda_x = 2\Delta x$. For

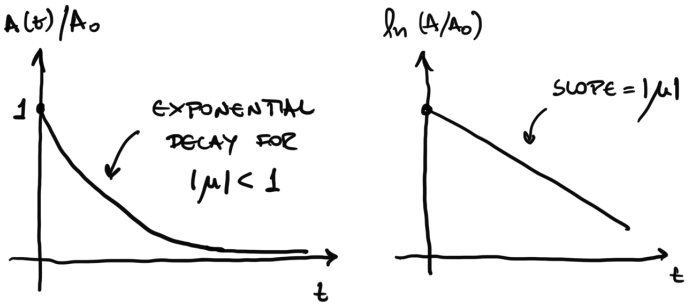

Figure 4.2: Amplitude decay caused by numerical diffusion. There is a typo in the figure: the slope is actually $-(1-|\mu|)$ instead of μ.

this case we have

$$k = \frac{2\pi}{\lambda_x} = \frac{\pi}{\Delta x} \Rightarrow k\Delta x = \pi \Rightarrow cos(k\Delta x) = -1. \tag{4.40}$$

So this is the worst case scenario, and the errors will be largest for the smallest resolved wavelength. This has important consequences: if the scheme is unstable, the smallest wavelengths will amplify first and will dominate the error before the solution diverge completely. If the scheme is stable, then the smallest wavelengths will be damped first. For any given solution (say we fix k), we can reduce the amount of damping by refining the grid (reducing Δx). As we can see from the expression for the numerical diffusivity (4.15), K_{eff} reduces as we reduce Δx. However, this is computationally expensive because a reduction in Δx also requires a reduction in Δt to maintain stability.

The combined effect of Courant number and wavelength on the error $R_{amp}(\mathcal{C}, \lambda_x \Delta x)$ as given by (4.38) is shown in Fig. 4.3, where the blue region corresponds to damped solution and red regions corresponds to regions in which the scheme is unstable.

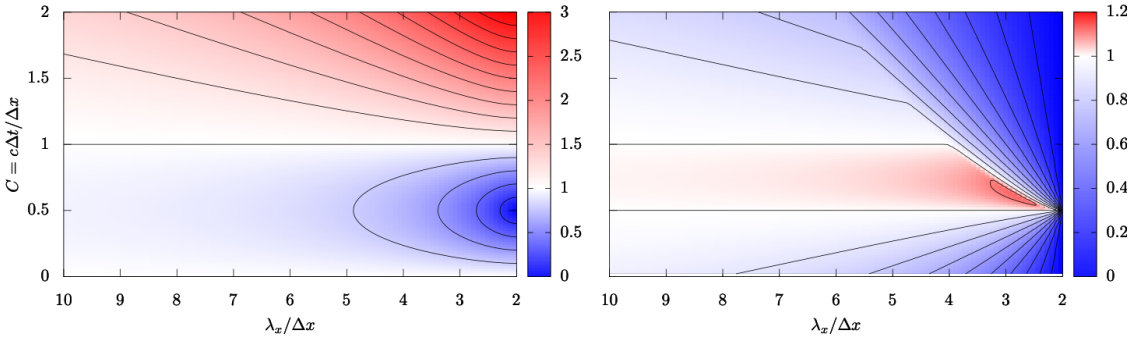

Figure 4.3: Amplification error R_{amp} (left) and phase error R_{ph} (right) for the FTBS scheme applied to the one-dimensional advection scheme (isolines shown in increments of 0.2).

The phase errors represent the errors associated with the propagation velocity. Even if the amplitude of the numerical solution is correct, the numerical solution may propagate

faster or slower than the exact solution. Similarly to (4.37), we define the phase error as

$$R_{ph} = \frac{\theta}{\theta_{ex}}. \tag{4.41}$$

Note that from the dispersion relation for the PDE we have that $\omega = vk$, so that $\omega_R = vk$. From (4.27), the phase associated with the exact solution is given by

$$\theta_{ex} = -\omega_R \Delta t = -vk\Delta t = -\mathcal{C}\Delta x. \tag{4.42}$$

From the discrete dispersion relation (4.33) we have that the phase for the numerical solution is

$$\theta = \arctan\left[\frac{-\mathcal{C}\sin(k\Delta x)}{1-\mathcal{C}+\mathcal{C}\cos(k\Delta x)}\right]. \tag{4.43}$$

Thus, the relative errors in the phase of the solution is

$$R_{ph} = \frac{\arctan\left[\frac{-\mathcal{C}\sin(k\Delta x)}{1-\mathcal{C}+\mathcal{C}\cos(k\Delta x)}\right]}{-\mathcal{C}k\Delta x}. \tag{4.44}$$

Note that $R_{ph} < 1$ implies that the numerical solution propagates slower than the exact solution, and $R_{ph} > 1$ implies that the numerical solution propagates faster than the exact solution. The best way to analyze the amplitude and phase errors is by plotting the expressions (4.38) and (4.44) above as a function of $k\Delta x$ (or equivalently $\lambda_x/\Delta x$) and $\mathcal{C}$. My best attempt at plotting this is shown in Fig. 4.3.

Here we will list the main conclusions about the use of the FTBS scheme to the linear advection equation (from Fig. 4.3):

- As we already know, the scheme is unstable for $\mathcal{C} > 1$ (as evidenced by $R_{amp} > 1$).
- Amplitude and phase errors impact mostly the short waves (say $\lambda_x \leq 4\Delta x$ or $k\Delta x \geq \pi/2$); when using this scheme we need to have a grid fine enough so that features of interest are at least of size $10\Delta x$ to minimize errors.
- For $\mathcal{C} = 1$ there are no amplitude errors and phase errors impact only very short waves, which propagate slower than they should.
- For $\mathcal{C} < 1$, solution amplitude will be damped by numerical diffusion, with stronger damping of shorter waves. Note that this can lead to some unexpected behavior (e.g., reducing Δt with Δx constant can increase errors).
- There are no phase errors for $\mathcal{C} = 0.5$, but this also corresponds to largest amplitude errors.
- Long waves propagate too fast for $0.5 < \mathcal{C} < 1$.

- Note that the phase speed for the exact solution is $c_{ph} = \omega/k = v$. Because all wavenumbers propagate at the same speed, we say that the waves are not dispersive (they all propagate together). However, for the numerical solution the speed of propagation depends on the wavenumber (as clearly shown by R_{ph}), so the numerical solution is dispersive. We refer to this as numerical dispersion, and we will see later that some schemes are dominated by numerical diffusion while other are dominated by numerical dispersion.

4.2 Discretization using centered in time (leapfrog) and centered in space scheme (CTCS)

Another commonly used explicit discretization for the advection equation is the leapfrog time discretization, which when combined with a centered scheme in space yields the CTCS scheme. The approximate equation is

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} + v\frac{(u_{i+1}^n - u_{i-1}^n)}{2\Delta x} = 0, \tag{4.45}$$

which yields the recursive equation

$$u_i^{n+1} = u_i^{n-1} - \mathcal{C}(u_{i+1}^n - u_{i-1}^n). \tag{4.46}$$

Note that the approximation of the derivative has a 2-step approach (i.e., it steps from t^{n-1} to t^{n+1}), while the right-hand side is evaluated at t^n. This results in an explicit method, which in this case is $\mathcal{O}(\Delta t^2, \Delta x^2)$. A schematic of the time-space stencil used in the CTCS scheme is shown in Fig. 4.4, where the origin of the name becomes clear, as the solution "leaps" over the point (x_i, t^n).

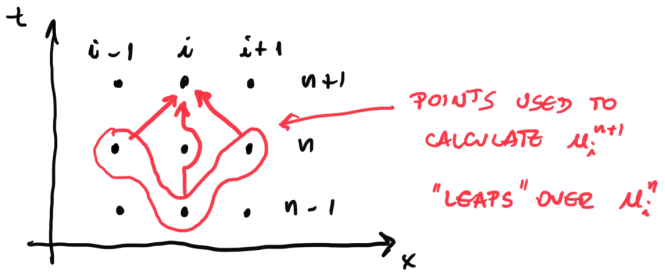

Figure 4.4: Stencil for the leapfrog scheme (CTCS).

4.2.1 Consistency and accuracy

If we follow the same approach from section 4.1.1, the truncation error for each derivative is

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} - \frac{\Delta t^2}{6}\left(\frac{\partial^3 u}{\partial t^3}\right)_i^n + \mathcal{O}\left(\Delta t^3\right) \tag{4.47}$$

$$\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{u_{i+1}^n - u_{i-1}^n}{2\Delta x} - \frac{\Delta x^2}{6}\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + \mathcal{O}\left(\Delta x^3\right). \tag{4.48}$$

We can relate spatial and time derivatives using the PDE. In the present case we have

$$\frac{\partial^3 u}{\partial t^3} = -v^3 \frac{\partial^3 u}{\partial x^3}, \tag{4.49}$$

and the resulting *modified equation* corresponding to the discrete equation (4.45) is

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v\left(\frac{\partial u}{\partial x}\right)_i^n = \frac{1}{6} v \Delta x^2 (\mathcal{C} - 1)\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n + \mathcal{O}\left(\Delta t^4, \Delta x^4\right). \tag{4.50}$$

Note that the leading order error in our approximation is not diffusive. If we define

$$-\beta_{\text{eff}} = \frac{1}{6} v \Delta x^2 (\mathcal{C} - 1), \tag{4.51}$$

the modified equation can be written as

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v\left(\frac{\partial u}{\partial x}\right)_i^n + \beta_{\text{eff}}\left(\frac{\partial^3 u}{\partial x^3}\right)_i^n = \mathcal{O}\left(\Delta t^4, \Delta x^4\right). \tag{4.52}$$

This is a linear version of the Korteweg de Vries equation (also called KdV for short), which is usually written as

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + \beta \frac{\partial^3 u}{\partial x^3} = 0, \tag{4.53}$$

and represents the behavior of dispersive waves. Thus, the appearance of a leading order error term including a third order derivative suggests that instead of numerical diffusion, the scheme has numerical dispersion and leads to non-physical oscillations. As a general rule, if the leading term in the truncation error is an odd-ordered derivative the errors are mostly dispersive in nature, and if it is even-ordered they are mostly diffusive/dissipative.

4.2.2 Stability analysis

If we replace the discrete Fourier mode (4.29) into the recursive equation (4.46), we obtain

$$\mu u_i^n = \frac{u_i^n}{\mu} - \mathcal{C}(e^{\mathrm{i}k\Delta x} - e^{-\mathrm{i}k\Delta x}) u_i^n. \tag{4.54}$$

Using Euler's formula we have that $(e^{\mathrm{i}k\Delta x} - e^{-\mathrm{i}k\Delta x}) = 2\mathrm{i}\sin(k\Delta x)$, so we obtain a quadratic equation for the discrete dispersion relation

$$\mu^2 + \mathrm{i}2\mathcal{C}\sin(k\Delta x)\mu - 1 = 0. \tag{4.55}$$

Now recall that the changing function relate the solution between two consecutive time steps:

$$\mu = \frac{u_i^{n+1}}{u_i^n}. \tag{4.56}$$

The problem of a quadratic equation for μ is that it implies the existence of two solutions u_i^{n+1} given one starting solution u_i^n. This always happens to multi-step schemes, and it will certainly cause additional problems.

To get a better understanding of the implications, let's define $a = \mathcal{C}\sin(k\Delta x)$ and solve the quadratic complex equation

$$\mu^2 + \mathrm{i}2a\mu - 1 = 0. \tag{4.57}$$

The solution is

$$\mu = -\mathrm{i}a \pm \sqrt{1 - a^2}, \tag{4.58}$$

which leads to two possibilities depending on whether $\sqrt{1-a^2}$ is real or imaginary. Let look into these two cases separately.

If $a^2 \leq 1$, then $\sqrt{1-a^2}$ is a real number and the solution is given by

$$\mu = \pm\sqrt{1 - a^2} - \mathrm{i}a, \tag{4.59}$$

which yields

$$|\mu|^2 = (1 - a^2 + a^2) = 1. \tag{4.60}$$

So in this case, the scheme is always stable and there are no errors in amplitude.

However, if $a^2 > 1$, then $\sqrt{1-a^2}$ is an imaginary number and the solution is given by

$$\mu = (\pm\sqrt{a^2 - 1} - a)\mathrm{i}, \tag{4.61}$$

which yields

$$|\mu|^2 = (\pm\sqrt{a^2 - 1} - a)^2. \tag{4.62}$$

Note that in this case, if $a < 1$ the positive root gives $|\mu|^2 > 1$, and if $a > 1$ the negative root gives $|\mu|^2 > 1$. Thus, the scheme is always unstable.

Therefore, the stability criterion becomes

$$a^2 =\leq 1 \Rightarrow -1 \leq a \leq 1, \tag{4.63}$$

or

$$-1 \leq \mathcal{C}\sin(k\Delta x) \leq 1. \tag{4.64}$$

Because we need the criterion to be satisfied for all values of k (i.e., the method must be stable for all Fourier modes), we require

$$-1 \leq \mathcal{C} \leq 1. \tag{4.65}$$

Note that the CFL condition is the same as for the FTBS, but as this is not an upwind scheme, the stability does not depend on the sign of v. As long as we keep $|\mathcal{C}| \leq 1$, the scheme is stable for waves propagating from both sides. Most schemes that use spatial discretization that are centered in space have stability criteria independent of the sign of v.

4.2.3 Physical and computational modes

From now on, we are only interested in the stable scheme when $a^2 \leq 1$. In this case the solution is

$$\mu = \pm\sqrt{1-a^2} - \mathrm{i}a, \tag{4.66}$$

where we know that the term in the square root is the real part. Clearly in this case $|\mu| = 1$ and we can show that $\theta = \arcsin(a)$, so the two solutions in polar form are

$$\mu_+ = e^{-\mathrm{i}\theta} \quad \text{and} \quad \mu_- = e^{\mathrm{i}(\theta+\pi)}. \tag{4.67}$$

Now if we write the initial condition as

$$u_i^0 = e^{\mathrm{i}ki\Delta x}, \tag{4.68}$$

then the solution at time t^n can be written as

$$u_i^n = \mu^n e^{\mathrm{i}ki\Delta x}. \tag{4.69}$$

Because we have two independent solutions for μ, we end up with a linear combination

$$u_i^n = (A_1\mu_+^n + A_2\mu_-^n)e^{\mathrm{i}ki\Delta x}, \tag{4.70}$$

where A_1 and A_2 are two constants. Note that here μ^n is really μ raised to the power n, which should not be confused with u_i^n where this represents u_i at the time t^n. Replacing the two solutions (4.67) and using the fact that $(e^{\mathrm{i}\pi})^n = (-1)^n$ we get

$$u_i^n = A_1 e^{-\mathrm{i}\theta n} e^{\mathrm{i}ki\Delta x} + A_2(-1)^n e^{\mathrm{i}\theta n} e^{\mathrm{i}ki\Delta x}. \tag{4.71}$$

Note that the opposite signs in the terms containing the phase information implies that the two components of the solution propagate in opposite directions. In addition, the second component switches signs every time step, so it is clearly the "unphysical" component in this case, and it is usually termed the *computational mode* of the solution. The importance of the computational mode depends on the specific details of the numerical method and the PDE being solved (which is given by the weights A_1 and A_2). For the CTCS scheme, the computational mode grows as the solution advances in time and it eventually dominates the solution.

If we consider only the physical mode (μ_+), we already know that there are no amplitude errors in the numerical solution (i.e., $|\mu| = 1$ and $R_{amp} = 1$). However, there are some phase errors for the physical mode that are important. In particular, we have

$$R_{ph} = \frac{\arcsin\left[\mathcal{C}\sin(k\Delta x)\right]}{\mathcal{C}k\Delta x}. \tag{4.72}$$

The phase error is not impacted much by changes in the Courant number, but it is a strong function of $k\Delta x$, with short waves propagating much slower than they should.

Thus, the main problem of the CTCS scheme is really the existence and amplification of the computational mode. One possible solution that is commonly used is to filter out the computational mode. This is usually done by using the Robert-Asselin time filter, which controls the amplitude of the computational mode. However, the inclusion of the filter leads to amplitude errors and more severe stability criteria, eliminating some of the advantages of the leapfrog scheme.

4.3 Discretization using Runge-Kutta of order 3 (RK3)

As a third and last approach to discretize the advection equation, we will introduce a multi-stage method that is frequently used in geosciences as a way to introduce the concept behind multi-stage (predictor-corrector) methods. In order to discuss this entire class of methods, it is better to start with our initial first-order ODE (2.1):

$$\frac{du}{dt} = q(u,t). \tag{4.73}$$

Recall that the Euler-Forward discretization

$$\frac{u^{n+1} - u^n}{\Delta t} = q(u^n, t^n) \tag{4.74}$$

results in an explicit scheme (because the right-hand side is evaluated using the known solution u^n) but it is only first-order in Δt. An improvement is obtained by the Crank-

Nicolson scheme (2.19)

$$u^{n+1} = u^n + \frac{\Delta t}{2}\left[q(u^n, t^n) + q(u^{n+1}, t^{n+1})\right], \tag{4.75}$$

which is second-order in Δt. However, the presence of u^{n+1} makes this an implicit scheme resulting in more elaborate implementation and higher computational cost. Note that this scheme only uses 2 points in time, and there is no computational mode as in the leapfrog scheme.

Multi-stage methods achieve higher order of accuracy by splitting one single time step into several stages. We will start with Heun's method, which illustrates how multi-stage methods can be used to increase convergence rate without using information from several steps. The idea of Heun's method is to leverage the second-order nature of the Crank-Nicolson scheme (2.19). The difficulty in using the trapezoidal rule is that it is an implicit method. To avoid this implicit nature, we split the calculation into two stages. The first stage is a "predictor", in which we estimate the solution u^{n+1} using the explicit Euler-forward. We label this estimate u^{n+1*}, to make it clear that this is not our final prediction for u^{n+1}. Thus

$$u^{n+1*} = u^n + \Delta t q(u^n, t^n). \tag{4.76}$$

Now we use this estimate to make the trapezoidal rule explicit (this is called the "corrector")

$$u^{n+1} = u^n + \frac{\Delta t}{2}\left[q(u^n, t^n) + q(u^{n+1*}, t^{n+1})\right]. \tag{4.77}$$

Thus, instead of using (2.19) we use the two stages given by (4.76) and (4.77), and achieve a second-order explicit method. In more general terms, sequences of predictors and correctors can be carefully developed to yield simultaneous gain in accuracy and stability.

4.3.1 Runge-Kutta methods

Runge-Kutta is a family of multi-step methods which split the integration interval $[t; t + \Delta t]$ into smaller segments and use a series of predictor-corrector combinations to achieve higher-order convergence together with improved stability. Runge-Kutta methods are classified based on the number of stages, so an RK2 has two stages and an RK4 has four stages. Even though RKs can be implicit or explicit, we will only focus on the latter here. Also RKn is n-order in Δt. RK1 is the same as the Euler forward scheme. The RK3 we will use here is not the standard RK3, but a different version that has gained popularity recently for combining third-order accuracy and reasonably good stability properties with fairly low computational and storage requirements. This is actually the scheme currently used in the Weather Research and Forecasting (WRF) Model. The three steps for this RK3 are:

1. Euler forward predictor advancing $\Delta t/3$:

$$u^{n+1/3*} = u^n + \frac{\Delta t}{3} q(u^n, t^n) \tag{4.78}$$

2. First corrector advancing $\Delta t/2$:

$$u^{n+1/2*} = u^n + \frac{\Delta t}{2} q(u^{n+1/3*}, t^{n+1/3}) \tag{4.79}$$

3. Mid-point corrector advancing Δt:

$$u^{n+1} = u^n + \Delta t q(u^{n+1/2*}, t^{n+1/2}) \tag{4.80}$$

The approach used in this RK3 is illustrated in Fig. 4.5.

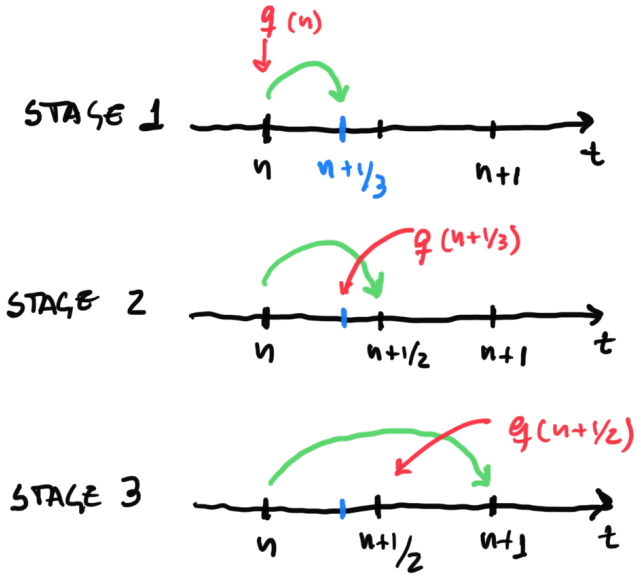

Figure 4.5: Schematic illustrating the time advancement for the RK3 scheme.

4.3.2 Application to the advection equation

The easiest way to apply this scheme to the advection scheme is to write the original PDE (4.2) as

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0 \Rightarrow \frac{du}{dt} = q(u,t) = -v\frac{\partial u}{\partial x}. \tag{4.81}$$

If we use a spatial discretization centered in space, then

$$q(u^n) = -\frac{v}{2\Delta x}(u^n_{i+1} - u^n_{i-1}). \tag{4.82}$$

Now we can simply replace this forcing into the three stage above:

1. Stage 1:
$$u_i^{n+1/3*} = u_i^n - \frac{1}{6}\mathcal{C}(u_{i+1}^n - u_{i-1}^n) \tag{4.83}$$

2. Stage 2:
$$u_i^{n+1/2*} = u_i^n - \frac{1}{4}\mathcal{C}(u_{i+1}^{n+1/3*} - u_{i-1}^{n+1/3*}) \tag{4.84}$$

3. Stage 3:
$$u_i^{n+1} = u_i^n - \frac{1}{2}\mathcal{C}(u_{i+1}^{n+1/2*} - u_{i-1}^{n+1/2*}) \tag{4.85}$$

So in each iteration of our marching in time, we need to calculate all three recursive equations above. Note that each one of the intermediate solutions ($u_i^{n+1/3*}$ and $u_i^{n+1/2*}$) must also satisfy boundary conditions, so each one of the recursive equations should be followed by proper implementation of boundary conditions.

Finally, we know that this scheme (RK3CS) is $\mathcal{O}\left(\Delta t^3, \Delta x^2\right)$, but we do not know anything else about it (stability, modified equation, etc.). So we need to improvise a bit when we are implementing this method.

4.4 Revisiting errors

It is a good opportunity to summarize Chapter 4 and revisit the original flow chart discussing the different types of errors in the numerical solution (see Fig. 4.6).

In practice we can measure errors in the numerical solution in many different ways. Two common definitions are based on the p-norm

$$\epsilon_p = \left[\sum_{i=1}^{N_x} |u_i^n - u(x_i, t^n)|^p\right]^{1/p} \tag{4.86}$$

or the maximum error

$$\epsilon_\infty = \max_i\{|u_i^n - u(x_i, t^n)|\}. \tag{4.87}$$

We usually use the p-norm error with $p = 2$ and the maximum error.

Our goal is to develop numerical schemes that are *convergent*: a numerical method is convergent if the solution of the discretized equations tens to the exact solution of the PDE as grid and time spacings tend to zero. Mathematically

$$\lim_{\Delta x, \Delta t \to 0} \epsilon_\infty = 0. \tag{4.88}$$

This implies that the numerical solution is "good enough" if we use Δx and Δt "small

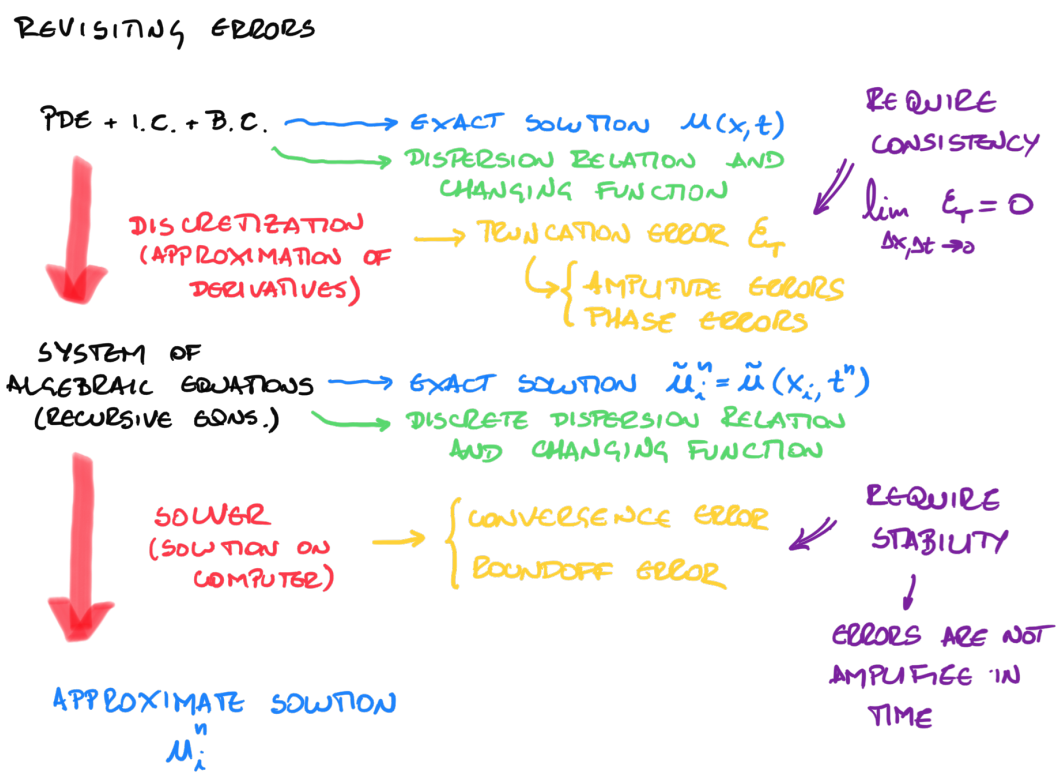

Figure 4.6: Flowchart illustrating the different approximations and errors.

enough". In practice, it means we can control how good the solution is by reducing Δx and Δt.

However, it is difficult to prove that any given scheme is convergent. Instead we rely on the LAX equivalence theorem: a *consistent* finite difference scheme method for a well-posed linear initial value problem is *convergent* if and only if it is *stable*. Thus, consistency plus stability is enough to guarantee consistency for this linear initial value problems. These are also necessary but not sufficient conditions for nonlinear problems and/or boundary value problems.

Chapter 5

The advection-diffusion equation

The goal in this chapter is to discuss discretization approaches for the advection-diffusion equation. We start with the one-dimensional version

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = \gamma\frac{\partial^2 u}{\partial x^2}. \tag{5.1}$$

However, we first start with the diffusion equation.

5.1 The diffusion equation

We start with the one-dimensional diffusion equation

$$\frac{\partial u}{\partial t} = \gamma\frac{\partial^2 u}{\partial x^2}, \tag{5.2}$$

defined for $0 \leq x \leq L_x$ and $t \geq 0$. Here we will take $\gamma > 0$ as the diffusivity and we will consider an initial condition $u(x, t = 0) = u_0(x)$. For boundary conditions we need to impose one condition at each end of the domain (one at $x = 0$ and one at $x = L_x$).

Suppose we use the Euler-forward scheme combined with the centered difference scheme for the second-order derivative in space (FTCS)

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\gamma}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \tag{5.3}$$

The recursive equation can be written as

$$u_i^{n+1} = u_i^n + \mathcal{S}(u_{i+1}^n - 2u_i^n + u_{i-1}^n), \tag{5.4}$$

where

$$\mathcal{S} = \frac{\gamma\Delta t}{\Delta x^2} \tag{5.5}$$

is the *von Neumann number*.

Using Taylor series expansions and the PDE to write time derivates in terms of spatial derivatives similarly to (4.12) yields the modified equation

$$\left(\frac{\partial u}{\partial t}\right)_i^n = \gamma \left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + \left(\frac{\Delta x^2}{12} - \frac{\gamma \Delta t}{2}\right) \gamma \left(\frac{\partial^4 u}{\partial x^4}\right)_i^n + \mathcal{O}\left(\Delta t^2, \Delta x^4\right). \tag{5.6}$$

As expected, the scheme is $\mathcal{O}\left(\Delta t, \Delta x^2\right)$, except for the special case in which

$$\frac{\Delta x^2}{12} = \frac{\gamma \Delta t}{2} \Rightarrow \mathcal{S} = \frac{1}{6}, \tag{5.7}$$

for which the scheme is $\mathcal{O}\left(\Delta t^2, \Delta x^4\right)$. It is interesting to note that the leading order error in the approximation is a hyperdiffusion term (i.e., a diffusion-like term with an even-ordered derivative of order larger than 2). This will manifest as an increase in the true diffusivity, but with more pronounced effect on the shorter waves.

If we perform stability analysis following the approach from Chapter 4 we find the dispersion relation

$$\mu = 1 + \mathcal{S}[e^{\mathrm{i}k\Delta x} - 2 + e^{-\mathrm{i}k\Delta x}] = 1 + 2\mathcal{S}[\cos(k\Delta x) - 1]. \tag{5.8}$$

For stability we require $|\mu| \leq 1$, which implies

$$-1 \leq 1 + 2\mathcal{S}[\cos(k\Delta x) - 1] \leq 1 \tag{5.9}$$

of

$$-1 \leq \mathcal{S}[\cos(k\Delta x) - 1] \leq 0. \tag{5.10}$$

Note that because $[\cos(k\Delta x) - 1] \leq 0$ (and the von Neumann number is always positive), the upper limit (≤ 0) is always satisfied. The worst case scenario for the lower limit ($-1 \leq$) occurs for $k\Delta x$ such that $[\cos(k\Delta x) - 1] = -2$, which yields the stability criterion

$$\mathcal{S} \leq \frac{1}{2}. \tag{5.11}$$

This is similar to the CFL condition for the advection equation. However, this also shows that improving the solution by refining the mesh in the diffusion equation is computationally much more expensive than for the advection equation. Note that for advection we have

$$\mathcal{C} = \frac{v\Delta t}{\Delta x} \leq 1, \tag{5.12}$$

so that if we reduce Δx by a factor of 10 we also need to reduce Δt by a factor of 10 to keep

the Courant number constant. In the diffusion equation,

$$\mathcal{S} = \frac{\gamma \Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{5.13}$$

implies that if we reduce Δx by a factor of 10 we must reduce Δt by a factor of 100.

However, recall that μ is the changing function, and if $\mu \leq 0$ the solution will be changing sign every time step. In practice, this only happens for some values of $k\Delta x$, mostly for short waves and in particular $\lambda_x = 2\Delta x$. This effect will manifest itself as unphysical oscillations in the solution. These oscillations do not cause numerical instability, because once they form they are quickly dissipated. The solution gets contaminated by unphysical oscillations that form and dissipate. To avoid this behavior, we can require $\mu \leq 0$ in addition to the stability criterion, resulting in the slightly more strict condition

$$\mathcal{S} \leq \frac{1}{4}. \tag{5.14}$$

Finally, as a side note, if we apply the leapfrog scheme to the diffusion equation we have

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} = \frac{\gamma}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \tag{5.15}$$

The changing function for this equation is

$$\mu = 2\mathcal{S}[\cos(k\Delta x) - 1] \pm \sqrt{4\mathcal{S}^2[\cos(k\Delta x) - 1]^2 + 1}. \tag{5.16}$$

In this case the worst case scenario is also for $[\cos(k\Delta x) - 1] = -2$, which in this case yields

$$\mu = -4\mathcal{S} \pm \sqrt{16\mathcal{S}^2 + 1}. \tag{5.17}$$

Because $\sqrt{16\mathcal{S}^2 + 1} \leq 1$, the scheme is always unstable.

5.2 The advection-diffusion equation

Now we shift our attention to the advection-diffusion equation, written as

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = \gamma \frac{\partial^2 u}{\partial x^2}. \tag{5.18}$$

One clear issue that we see from the previous discussion is that some schemes are better to handle advection and other schemes are better for diffusion (of course some are good for both). For example, FTBS works well for advection (as long as it is upwind), but it is not as good for diffusion (because it is not symmetric in space and that can introduce artificial asymmetry in the diffusion process). FTCS works well for diffusion, but it is always unstable

for advection. CTCS works well for advection, but it is always unstable for diffusion.

In general, stable treatment of diffusion can stabilize advection, but not vice-versa. As an example, take the FTCS scheme, which is conditionally stable for diffusion (absolutely stable for $\mathcal{S} \leq 1/2$) and unconditionally unstable for advection (i.e., it will always produce negative diffusion, no matter the value of $\mathcal{C}$). Discretization of the advection-diffusion equation yields

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} + \frac{v}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) = \frac{\gamma}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \tag{5.19}$$

with recursive equation

$$u_i^{n+1} = u_i^n - \frac{\mathcal{C}}{2}(u_{i+1}^n - u_{i-1}^n) + \mathcal{S}(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \tag{5.20}$$

If we perform von Neumann stability analysis, we find that the scheme is absolutely stable for

$$|\mu|^2 = [1 + 2\mathcal{S}(\cos\theta - 1)]^2 + (\mathcal{C}\sin\theta)^2 \leq 1, \tag{5.21}$$

which is satisfied as long as

$$\mathcal{S} \leq \frac{1}{2} \quad \text{and} \quad \mathcal{C} \leq 2(1 - \mathcal{S}). \tag{5.22}$$

The first criterion is required for the treatment of diffusion to be stable (i.e., for the FTCS to be stable for the diffusion equation), and the second is required for diffusion to stabilize the otherwise unstable treatment of advection. Note that the latter can be written as

$$\frac{v\Delta t}{\Delta x} \leq 2\left(1 - \gamma\frac{\Delta t}{\Delta x^2}\right) \Rightarrow Pe \leq 2. \tag{5.23}$$

Here,

$$Pe = \frac{v\Delta x}{\gamma} \tag{5.24}$$

is the grid Péclet number, representing the ratio between advective transport and diffusive transport over a distance Δx.

However, the requirement on the grid Péclet number is very restrictive for advection dominated problems and it is usually not a good idea to use the same scheme for advection and diffusion. Even though some schemes can be easily combined (e.g., FTUpwind for advection and FTCS for diffusion), a more general approach that allows for combination of different time discretizations is usually employed. This is done by using operator splitting techniques.

We will not have a formal discussion of discuss of operator splitting techniques here, but we will see one simple example combining leapfrog with centered in space for advection (CTCS) and FTCS for diffusion. In this case, if we separate the complete equation into the

advection and diffusion we have

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} = -\frac{v}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) \tag{5.25}$$

for the advection part and

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \frac{\gamma}{\Delta x^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n). \tag{5.26}$$

for the diffusion part.

Note that we can make the time derivative of the two schemes compatible by using FTCS with a timestep of $2\Delta t$ for diffusion:

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} = \frac{\gamma}{\Delta x^2}(u_{i+1}^{n-1} - 2u_i^{n-1} + u_{i-1}^{n-1}). \tag{5.27}$$

If we now apply these schemes to the complete equation we have

$$\frac{u_i^{n+1} - u_i^{n-1}}{2\Delta t} = -\frac{v}{2\Delta x}(u_{i+1}^n - u_{i-1}^n) + \frac{\gamma}{\Delta x^2}(u_{i+1}^{n-1} - 2u_i^{n-1} + u_{i-1}^{n-1}). \tag{5.28}$$

At this point the true stability analysis becomes a bit more complicated, and we can simply use the individual criteria as an approximation. For the advection part we have the CFL condition

$$\mathcal{C} = \frac{v\Delta t}{\Delta x} \leq 1, \tag{5.29}$$

and for the diffusion part we have

$$\mathcal{S} = \frac{2\gamma\Delta t}{\Delta x^2} \leq \frac{1}{2}. \tag{5.30}$$

Note that since diffusion uses a $2\Delta t$ step, we defined the von Neumann number using $2\Delta t$. Also keep in mind that to prevent artificial numerical oscillations it may be better to use $\mathcal{S} \leq (1/4)$.

5.3 Advection and diffusion in 2 spatial dimensions

So far we have done advection and diffusion in one spatial dimension (1D). Now we move to the 2D case, starting with the 2D advection equation.

$$\frac{\partial u}{\partial t} + v_x\frac{\partial u}{\partial x} + v_y\frac{\partial u}{\partial y} = 0, \tag{5.31}$$

on $0 \leq x \leq L_x$ and $0 \leq y \leq L_y$ with $v_x > 0$, $v_y > 0$, and appropriate initial and boundary conditions (in this case one boundary condition at $x = 0$ and one at $y = 0$). Note that

in this case the solution is $u(x, y, t)$, and we will use $u_{i,j}^n$ where the index i corresponds to the node number in the x-direction and the index j corresponds to the node number in the y-direction.

[Figure]

For simplicity, let's look at the upwind scheme FTBS given by

$$u_{i,j}^{n+1} = u_{i,j}^n - \frac{v_x \Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - \frac{v_y \Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n). \tag{5.32}$$

The modified equation is now slightly more complicated because it includes terms with mixed derivatives

$$\left(\frac{\partial u}{\partial t}\right)_i^n + v_x \left(\frac{\partial u}{\partial x}\right)_i^n + v_y \left(\frac{\partial u}{\partial y}\right)_i^n = \frac{v_x \Delta x}{2}\left(1 - \frac{v_x \Delta t}{\Delta x}\right)\left(\frac{\partial^2 u}{\partial x^2}\right)_i^n + \\ + \frac{v_y \Delta y}{2}\left(1 - \frac{v_y \Delta t}{\Delta y}\right)\left(\frac{\partial^2 u}{\partial y^2}\right)_i^n + v_x v_y \Delta t \left(\frac{\partial^2 u}{\partial x \partial y}\right)_i^n + \mathcal{O}\left(\Delta t^2, \Delta x^2\right). \tag{5.33}$$

The last term is first-order in Δt and it produces a distortion of the wave, which manifests itself as an anti-diffusive term in the direction of advection and as a diffusive term in the direction perpendicular to the advection.

The stability analysis deserves some quick comments too. Now the discrete solution is represented as

$$u_{i,j}^n = e^{\mathrm{i}(k_x i \Delta x + k_y j \Delta y - \omega n \Delta t)}, \tag{5.34}$$

and the von Neumann analysis leads to a modified CFL condition for absolute stability given by

$$\frac{v_x \Delta t}{\Delta x} + \frac{v_y \Delta t}{\Delta y} \leq 1. \tag{5.35}$$

For the condition $v_x = v_y = v$ and $\Delta x = \Delta y = \delta$ this results in

$$\frac{v \Delta t}{\delta} \leq \frac{1}{2}, \tag{5.36}$$

which is clearly more stringent than in the one-dimensional case.

Similarly, if we use the leapfrog scheme (CTCS) we have

$$u_{i,j}^{n+1} = u_{i,j}^{n-1} - \frac{v_x \Delta t}{\Delta x}(u_{i,j}^n - u_{i-1,j}^n) - \frac{v_y \Delta t}{\Delta y}(u_{i,j}^n - u_{i,j-1}^n). \tag{5.37}$$

This also results in the same stability criterion as above.

In the case of diffusion in two spatial dimensions we have

$$\frac{\partial u}{\partial t} = \gamma_x \frac{\partial^2 u}{\partial x^2} + \gamma_y \frac{\partial^2 u}{\partial y^2}. \tag{5.38}$$

Let's look at the FTCS scheme, which in this case yields

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\gamma_x \Delta t}{\Delta x^2}(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}) + \frac{\gamma_y \Delta t}{\Delta y^2}(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n}), \tag{5.39}$$

the the stability condition yields

$$\frac{\gamma_x \Delta t}{\Delta x^2} + \frac{\gamma_y \Delta t}{\Delta y^2} \leq \frac{1}{2}. \tag{5.40}$$

As before, we can use the more restrict limit of $1/4$ to avoid unphysical oscillations. Note that if we use $\gamma_x = \gamma_y = \gamma$ and $\Delta x = \Delta y = \delta$ we have

$$\frac{\gamma \Delta t}{\delta^2} \leq \frac{1}{4}, \tag{5.41}$$

or $\leq (1/8)$ to avoid physical oscillations. As in the case of the advection equation, the stability criterion is more stringent than in the 1D case.

Chapter 6

The barotropic QG equations

The quasi-geostrophic (QG) approximation is a simplification of the Navier-Stokes equations on the Earth's rotating frame of reference (angular velocity Ω) designed to study large scale (synoptic scale) flows in the atmosphere and in the ocean in the mid-latitudes. In essence, these equations can be obtained from the full compressible Navier-Stokes equations under the assumptions that (i) hydrostatic balance holds in the vertical direction, (ii) the Reynolds number is very large ($Re = UL/\nu \gg 1$), and (iii) the Rossby number is very small ($Ro = U/(fL) < 1$, with $f = 2\Omega \sin\phi$ being the Coriolis parameter and ϕ the latitude). Under these conditions, the governing equations become:

$$\frac{Du}{Dt} = -\frac{\partial \Phi}{\partial x} + fv \tag{6.1}$$

$$\frac{Dv}{Dt} = -\frac{\partial \Phi}{\partial y} - fu \tag{6.2}$$

$$\frac{\partial \Phi}{\partial p} = -\frac{1}{\rho} \tag{6.3}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial \omega}{\partial p} = 0 \tag{6.4}$$

$$p = \rho R_d T. \tag{6.5}$$

Here, (6.1) and (6.2) are the two horizontal components of the momentum equations, (6.3) is the hydrostatic equation obtained from the vertical momentum equation, (6.4) is the full compressible conservation of mass (or continuity equation), and (6.5) is the ideal gas law (R_d is the gas constant for dry air). Note that because of hydrostatic balance, there is a monotonic relation between pressure and height and these equations are written in pressure coordinates by defining the geopotential height via

$$d\Phi = g dz, \tag{6.6}$$

and replacing the independent variable z by p (this can be done formally via coordinate transformation). Therefore, we switch from a system where we describe velocity, temperature, density, and pressure as a function of 3D position and time to a system in which we describe velocity, temperature, density, and (geopotential) height as a function of x, y, p, and t. This is indicated below:

$$\begin{cases} u = u(x,y,z,t) \\ v = v(x,y,z,t) \\ w = w(x,y,z,t) \\ T = T(x,y,z,t) \\ \rho = \rho(x,y,z,t) \\ p = p(x,y,z,t) \end{cases} \Rightarrow \begin{cases} u = u(x,y,p,t) \\ v = v(x,y,p,t) \\ \omega = \omega(x,y,p,t) \\ T = T(x,y,p,t) \\ \rho = \rho(x,y,p,t) \\ \Phi = \Phi(x,y,p,t) \end{cases} . \tag{6.7}$$

In this isobaric coordinate system, the pressure gradient force is given by

$$-\frac{1}{\rho}\nabla p = -\nabla\Phi, \tag{6.8}$$

and the vertical velocity is defined as

$$\omega = \frac{Dp}{Dt}. \tag{6.9}$$

Finally, the material derivative here is also written in pressure coordinates

$$\frac{D}{Dt} \equiv \frac{\partial}{\partial t} + u\frac{\partial}{\partial x} + v\frac{\partial}{\partial y} + \omega\frac{\partial}{\partial p}. \tag{6.10}$$

To complete the system we also need an internal energy equation obtained from the first-law of thermodynamics, but we will not need this here. Note that despite all the simplifications, the QG equations still yield a three-dimensional flow in which the vertical velocity ω is diagnosed from the horizontal velocity components via conservation of mass.

Note that the first-order approximation to the system of equations for the case $Ro \ll 1$ is given by geostrophic balance (i.e., a balance of pressure gradient forces and Coriolis forces)

$$u_g = -\frac{1}{f_0}\frac{\partial\Phi}{\partial y} \quad \text{and} \quad v_g = \frac{1}{f_0}\frac{\partial\Phi}{\partial x}. \tag{6.11}$$

However, the numerical solution of the unsteady system in the form given above is very difficult. We will make a series of approximations that are often used for large scale flows in the atmosphere and ocean, to recast this system in a simpler form that is more amenable to numerical implementation.

6.1 Simplifications of the QG system

We will start by invoking the β-plane approximation for the Coriolis term. The issue here is that, even though we are using a local cartesian coordinate system (the equations are not in the original spherical coordinates), the Coriolis parameter is still a function of the latitude ϕ. We will maintain the latitude dependence of the Coriolis parameter using a Taylor series expansion to write

$$f \approx f_0 + \beta y, \tag{6.12}$$

where f_0 is a constant representing the Coriolis parameter at the center of the domain of interest and $\beta = df/dy$ is a first-order approximation for the dependence of f on latitude (here represented in the tangent plane via y).

To simplify the QG system of equations, it is useful to split the horizontal velocity $\mathbf{v} = (\mathbf{u}, \mathbf{v}, \mathbf{0})$ into its geostrophic ($\mathbf{v_g}$) and ageostrophic $\mathbf{v_a}$) components, keeping in mind that because of the small Rossby number we expect $|\mathbf{v_g}| \gg |\mathbf{v_a}|$. It is also easy to show from (6.11) that $\nabla \cdot \mathbf{v_g} = \mathbf{0}$, so that the continuity equation becomes

$$\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y} + \frac{\partial \omega}{\partial p} = 0. \tag{6.13}$$

So in the QG system, all the vertical motion is associated with the ageostrophic component of the flow. Next we approximate the material derivative by neglecting the ageostrophic terms in comparison with the much large geostrophic ones:

$$\frac{D}{Dt} = \frac{\partial}{\partial t} + (u_g + u_a)\frac{\partial}{\partial x} + (v_g + v_a)\frac{\partial}{\partial y} + \omega\frac{\partial}{\partial p} \approx \frac{\partial}{\partial t} + u_g\frac{\partial}{\partial x} + v_g\frac{\partial}{\partial y} \equiv \frac{D_g}{Dt} \tag{6.14}$$

Here D_g/Dt is the QG material derivative.

Next we simplify the equations for the horizontal momentum balance. For the zonal momentum balance (i.e. the x component) we have

$$\frac{D_g(u_g + u_a)}{Dt} = -\frac{\partial \Phi}{\partial x} + f_0(v_g + v_a) + \beta y(v_g + v_a). \tag{6.15}$$

Note that by the definition (6.11), the first two terms on the right-hand side cancel out, and we write in approximate form

$$\frac{D_g u_g}{Dt} = f_0 v_a + \beta y v_g. \tag{6.16}$$

If we repeat the same approach to the meridional momentum balance, we arrive at a simpler

QG system

$$\frac{D_g u_g}{Dt} = f_0 v_a + \beta y v_g \tag{6.17}$$

$$\frac{D_g v_g}{Dt} = -f_0 u_a - \beta y u_g \tag{6.18}$$

$$\frac{\partial \Phi}{\partial p} = -\frac{1}{\rho} \tag{6.19}$$

$$\frac{\partial u_a}{\partial x} + \frac{\partial v_a}{\partial y} + \frac{\partial \omega}{\partial p} = 0. \tag{6.20}$$

Even though the system is simpler than the original one, we still have a system of coupled PDEs and a three-dimensional velocity field. We will simplify the problem further by taking two steps:

1. We will recast the system in terms of vorticity and streamfunction, as this allows us to eliminate one of the unknowns and one of the equations;

2. We will use the assumption that the flow is barotropic to eliminate the vertical velocity and reduce the three-dimensionality of the flow.

However, we cannot separate the two steps, they will be combined in our approach.

6.2 Vorticity-streamfunction system for barotorpic flow

We start by defining the (vertical component of) geostrophic vorticity as

$$\zeta_g \equiv \frac{\partial v_g}{\partial x} - \frac{\partial u_g}{\partial y}. \tag{6.21}$$

Thus, in order to write a prognostic equation fro ζ_g we subtract $\partial(6.17)/\partial y$ from $\partial(6.18)/\partial x$

$$\frac{\partial}{\partial x}\left(\frac{D_g v_g}{Dt}\right) - \frac{\partial}{\partial y}\left(\frac{D_g u_g}{Dt}\right) = \frac{\partial}{\partial x}(-f_0 u_a - \beta y u_g) - \frac{\partial}{\partial y}(f_0 v_a + \beta y v_g). \tag{6.22}$$

Using the fact that the geostrophic flow is divergence-free (i.e., $\nabla \cdot \mathbf{v_g} = \mathbf{0}$) and (6.20) together with the definition (6.21), we can write Eq. (6.22) as

$$\frac{\partial \zeta_g}{\partial t} + u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} + \beta v_g = f_0 \frac{\partial \omega}{\partial p}. \tag{6.23}$$

This is the QG vorticity equation and the terms represent in order: (i) geostrophic vorticity tendency, (ii, iii) advection of geostrophic vorticity, (iv) advection of planetary vorticity, and (v) stretching of planetary vorticity by the ageostrophic flow (i.e., changes in relative vorticity from stretching of vertical columns).

Now we will make the assumption that we are dealing with a barotropic flow, which means that density is a function of pressure alone (i.e., temperature and salinity do not impact the density). Recall from the thermal wind relations that the shear in the geostrophic flow can be related to the horizontal temperature gradients

$$\frac{\partial \mathbf{v_g}}{\partial \ln p} = -\frac{R_d}{f}\hat{\mathbf{k}} \times \nabla_p T. \tag{6.24}$$

Now for a barotropic flow, $\rho = \rho(p)$ and obviously density must be constant along an isobaric surface. From the ideal gas law we can infer that temperature is also constant along an isobaric surface, so that $\nabla_p T = 0$. This implies that u_g and v_g are constant with height (given in terms of pressure levels), and so is ζ_g. This allows us to integrate the QG vorticity equation for the entire depth of the atmosphere as

$$\int_0^{p_s} \left(\frac{\partial \zeta_g}{\partial t} + u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} + \beta v_g \right) dp = \int_0^{p_s} \left(f_0 \frac{\partial \omega}{\partial p} \right) dp, \tag{6.25}$$

where p_s is surface pressure (i.e., at the ground) and $p = 0$ at the top of the atmosphere. In the barotropic atmosphere, the integrand on the left-hand side is constant with respect to p, so that we have

$$\left(\frac{\partial \zeta_g}{\partial t} + u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} + \beta v_g \right) p_s = f_0(\omega_s - \omega_0). \tag{6.26}$$

We will make the assumption that $\omega_0 = 0$, and that the vertical velocity at the surface is forced by the topography via hydrostatic equation. Thus, we have $p_s = \rho g(H - z_s)$ with H being the depth of the atmosphere and z_s the surface height used to represent topography. In this case,

$$\omega_s = \frac{Dp_s}{Dt} = \frac{\partial p_s}{\partial t} + \mathbf{v}_g \cdot \nabla p_s = -\rho g \mathbf{v}_g \cdot \nabla z_s. \tag{6.27}$$

This yields the final form of the QG equation for a barotropic flow

$$\frac{\partial \zeta_g}{\partial t} + \mathbf{v}_g \cdot \nabla \zeta_g = -\beta v_g - \frac{f_0}{(H - z_s)} \mathbf{v}_g \cdot \nabla z_s. \tag{6.28}$$

For applications in the atmosphere, it is more convenient to use the ideal gas law and write

$$\frac{\partial \zeta_g}{\partial t} + \mathbf{v}_g \cdot \nabla \zeta_g = -\beta v_g - \frac{f_0 g}{R_d T_s} \mathbf{v}_g \cdot \nabla z_s. \tag{6.29}$$

Here the two terms on the right-hand side represent changes in the relative vorticity of an air parcel due to changes in latitude and sloped terrain (T_s is the surface temperature).

Now that we are dealing with a two-dimensional flow (because of the vertical integration), it becomes advantageous to introduce a streamfunction ψ to represent the velocity. We define

the streamfunction via the relations

$$u_g = -\frac{\partial \psi}{\partial y} \quad \text{and} \quad v_g = \frac{\partial \psi}{\partial x}. \tag{6.30}$$

By definition, the value of the streamfucntion is constant a long a streamline (i.e., a line that is tangent to the velocity field everywhere, see Fig. 6.1). The reason to introduce the stremfunction is that the velocity field obtained from ψ automatically satisfies the divergence-free condition

$$\frac{\partial u_g}{\partial x} + \frac{\partial v_g}{\partial y} = -\frac{\partial^2 \psi}{\partial x \partial y} + \frac{\partial^2 \psi}{\partial y \partial x} = 0. \tag{6.31}$$

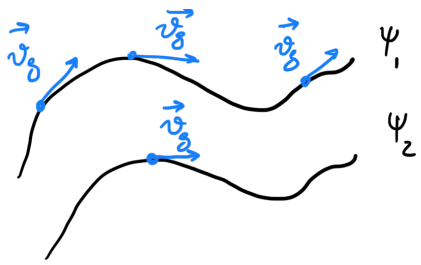

Figure 6.1: Schematic illustrating the concept of streamfunctions.

Finally, note that now (6.21) yields

$$\zeta_g = \frac{\partial v_g}{\partial x} - \frac{\partial u_g}{\partial y} = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \nabla^2 \psi. \tag{6.32}$$

Our final barotropic QG system in vorticity streamfunction formulation is given by

$$\frac{\partial \zeta_g}{\partial t} + \mathbf{v}_g \cdot \nabla \zeta_g = -\beta v_g - \frac{f_0 g}{R_d T_s} \mathbf{v}_g \cdot \nabla z_s \tag{6.33}$$

$$\nabla^2 \psi = \zeta_g, \tag{6.34}$$

complemented by the relations between the streamfunction and the velocity given by (6.30). So we have reduced our system to one single prognostic equation for vorticity and one Poisson equation for the streamfunction. This is as simple as it gets for the barotropic QG system. Before we can solve this system, we need to deal with three remaining issues:

1. Solution of elliptic equations (for the Poisson's equation);
2. Discretization of the nonlinear advection;
3. Boundary conditions for ψ and ζ_g.

6.3 Comparison to two-dimensional Navier-Stokes (extra reading)

At this point, it may be useful to make an analogy between the QG system described above, and the more general problem of solving the two-dimensional incompressible Navier-Stokes equations.

$$\frac{Du}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\nabla^2 u \tag{6.35}$$

$$\frac{Dv}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\nabla^2 v \tag{6.36}$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{6.37}$$

There are many challenges associated with solving the equations in this form (also known as the velocity-pressure formulation), which will be discussed later in the course. For now, we simply want to note that the same approach adopted above, of writing the system in vorticity-streamfunction formulation, yields a simpler system of equations. The vertical component of vorticity is defined as

$$\omega_z \equiv \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}. \tag{6.38}$$

Note that the fluid dynamics community usually adopts a different sign convention for the definition of streamfunction

$$u = \frac{\partial \psi}{\partial y} \quad \text{and} \quad v = -\frac{\partial \psi}{\partial x}. \tag{6.39}$$

In this case, by introducing the streamfunction we are already enforcing incompressibility. We proceed as above and obtain an equation for the vertical vorticity ω_z and a relation between vorticity and streamfucntion. The final system of equations is

$$\frac{\partial \omega_z}{\partial t} + \mathbf{v}\cdot\nabla\omega_z = \nu\nabla^2\omega_z \tag{6.40}$$

$$\nabla^2\psi = -\omega_z. \tag{6.41}$$

Note that the forcing term on the Poisson's equation has a negative sign, because of the different sign convention used for the streamfunction.

Comparing the velocity-pressure and the vorticity-streamfunction formulations, the main gain is the reduction in the number of PDEs and unknowns, from three to two. Note that pressure no longer appears in the equation. In the incompressible system, the only role of pressure is to maintain the divergence-free condition, which is now automatically enforced by the streamfunction. Because we have not made the assumption of large Reynolds number used in the QG system, the viscous term is present in the final formulation. For small

Reynolds numbers, this problem is easier than the QG system because viscosity can keep the nonlinear instability in check. Even a simple approach with FTCS for the vorticity equation combined with a Poisson solver can work with appropriate choices of grid size and time step (this approach is useless for the QG system). A difficult problem that arises here is the specification of proper boundary conditions, specially no-slip boundary conditions at solid walls.

Chapter 7

Poisson's equations

Last class we realized that we need to solve an elliptic equation for the streamfunction as part of our barotropic Q.G. system.

$$\nabla^2\psi = \frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = \zeta_g. \tag{7.1}$$

This is a very different problem from the advection and diffusion equations, as there is no time evolution. Let's start with a simple 1D model problem:

$$\frac{d^2u}{dx^2} = q(x), \tag{7.2}$$

defined for $0 \leq x \leq L_x$ with Dirichlet boundary conditions given by

$$u(x=0) = a \qquad\qquad u(x=L_x) = b. \tag{7.3}$$

In this type of problem, we know the solution on both ends of the domain, and the marching scheme cannot be used.

If we use N_x points and the centered scheme for the second-order derivative we have a discrete system of N_x coupled algebraic equations:

$$\begin{cases} \dfrac{1}{\Delta x^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) = q_i \quad \text{for } i = 2, ..., (N_x - 1) \\ u_1 = a \\ u_{N_x} = b \end{cases} . \tag{7.4}$$

In this case, we used the ordinary differential equation (ODE) to write discrete equations for the interior nodes, and the boundary conditions to specify the values for the boundary nodes.

We are now looking for the discrete solution vector with N_x elements $[u] = [u_1\ u_2\ u_3\ ...\ u_{N_x}]^T$.

It is usually better to multiply the discrete equation by Δx^2 and write the system of equations in matrix form:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & \vdots & \vdots & \vdots \\ 0 & 1 & -2 & 1 & \dots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \dots & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N_x-1} \\ u_{N_x} \end{bmatrix} = \begin{bmatrix} a \\ \Delta x^2 q_2 \\ \Delta x^2 q_3 \\ \vdots \\ \Delta x^2 q_{N_x-1} \\ b \end{bmatrix}. \tag{7.5}$$

Therefore, we need to solve the $N_x \times N_x$ linear system $[A][u] = [q]$. We can also write the system as $[u] = [A]^{-1}[q]$, so finding a solution is equivalent to inverting the square matrix $[A]$. Usually this matrix has nice properties (sparse, diagonally dominant, etc.) that help the numerical solution. There are two group of approach to solve the system of equations: (i) direct methods consist of inverting $[A]$ numerically (e.g. Gauss elimination) and (ii) iterative methods consist of choosing an initial guess solution $[u]$ and iterating it until some convergence criterion is reached (e.g., Jacobi, Gauss-Seidel, SOR, etc.). In this specific example, $[A]$ is a banded matrix with bandwidth 3 (also called a tridiagonal matrix). For tridiagonal matrices, Thomas algorithm is a direct method that provides an extremely fast and accurate solution.

7.1 Solution of elliptic equations

Here we are going to use Poisson's equation in two dimensions as a prototype problem for elliptic equations:

$$\nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = q(x, y), \tag{7.6}$$

defined on $0 \leq x \leq L_x$ and $0 \leq y \leq L_y$ with appropriate boundary conditions.

We discretize the domain using $N_x \times N_y$ points, such that:

$$x_i = (i-1)\Delta x; \quad i = 1, ..., N_x; \quad \Delta x = \frac{L_x}{N_x - 1} \tag{7.7}$$

$$y_j = (j-1)\Delta y; \quad j = 1, ..., N_y; \quad \Delta y = \frac{L_y}{N_y - 1}. \tag{7.8}$$

We now have two sets of indices so that

$$u(x_i, y_j) = u_{i,j} \quad \text{and} \quad q(x_i, y_j) = q_{i,j}. \tag{7.9}$$

If we use the second-order centered scheme for both directions we have

$$\left(\frac{\partial^2 u}{\partial x^2}\right)_{i,j} = \frac{1}{\Delta x^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) + \mathcal{O}\left(\Delta x^2\right) \tag{7.10}$$

$$\left(\frac{\partial^2 u}{\partial y^2}\right)_{i,j} = \frac{1}{\Delta y^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) + \mathcal{O}\left(\Delta y^2\right), \tag{7.11}$$

and the discrete form of the Poisson's equation becomes

$$\frac{1}{\Delta x^2}\left(u_{i-1,j} - 2u_{i,j} + u_{i+1,j}\right) + \frac{1}{\Delta y^2}\left(u_{i,j-1} - 2u_{i,j} + u_{i,j+1}\right) = q_{i,j} + \mathcal{O}\left(\Delta x^2, \Delta y^2\right). \tag{7.12}$$

The scheme corresponds to the 5-point stencil represented in Fig. 7.1.

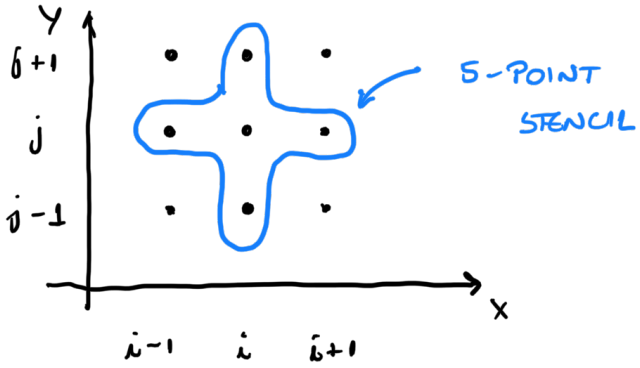

Figure 7.1: 5-point stencil for Poisson's equation with centered second-order finite difference scheme.

Note that now, the discrete dependent variable $[u]$ is a matrix of $N_x \times N_y$ points. If we want to represent the system of algebraic equations in matrix form $[A][u] = [q]$ to solve it using conventional method, we need to collapse the matrices $u_{i,j}$ and $q_{i,j}$ into vectors of size $N_x N_y$. This process is called ordering, and even though it is not hard it requires great book-keeping. We will not use this approach here. Instead, we will use iterative methods to avoid this complication.

For iterative methods, the simplest approach is the Jacobi method and its (faster) adaptations Gauss-Seidel and successive over-relaxation (SOR). We will briefly review the basics of the Jacobi method, as the other two consist of small changes.

Suppose we are interested in the solution of the generic system of equations $[A][u] = [q]$. The basic idea is to split $[A]$ into its diagonal component ($[D]$) and the remainder ($[R]$), so

that $[A] = [D] + [R]$ with:

$$[D] = \begin{bmatrix} a_{11} & 0 & 0 & \dots \\ 0 & a_{22} & 0 & \dots \\ 0 & 0 & a_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix} \quad \text{and} \quad [R] = \begin{bmatrix} 0 & a_{12} & a_{13} & \dots \\ a_{21} & 0 & a_{23} & \dots \\ a_{31} & a_{32} & 0 & \\ \vdots & \vdots & & \ddots \end{bmatrix}. \tag{7.13}$$

Note that we can use this decomposition to rewrite the system as an implicit equation for $[u]$

$$[A][u] = [D][u] + [R][u] = [q]. \tag{7.14}$$

Now we note that inverting the diagonal matrix $[D]$ is trivial

$$[D]^{-1} = \begin{bmatrix} 1/a_{11} & 0 & 0 & \dots \\ 0 & 1/a_{22} & 0 & \dots \\ 0 & 0 & 1/a_{33} & \\ \vdots & \vdots & & \ddots \end{bmatrix}. \tag{7.15}$$

Therefore, it is advantageous to write the system as

$$[D][u] = [q] - [R][u] \Rightarrow [u] = [D]^{-1}\left([q] - [R][u]\right). \tag{7.16}$$

This is an implicit equation for the solution vector $[u]$, which can then be solved iteratively by "guessing" an initial solution $[u]^0$ and refining it using

$$[u]^{k+1} = [D]^{-1}\left([q] - [R][u]^k\right). \tag{7.17}$$

In this expression, $[u]^k$ is the solution vector in the current iteration, $[D]$, $[q]$, and $[R]$ are known.

The recursive equation for implementation is given by

$$u_i^{k+1} = \frac{1}{a_{ii}}\left(q_i - \sum_{j \neq i} a_{ij} u_j^k\right). \tag{7.18}$$

Here the index k is an iteration counter. Because we will solve the system using iterations, there is no need to collapse $u_{i,j}$ into a vector and explicitly setup the matrix $[A]$. We can simply iterate the values of the matrix $u_{i,j}$ on the grid. But first we need to figure out the details of the recursive equation corresponding to the 2D Poisson's equation. In order to do

so, we can re-write the discrete equation (7.12) as

$$\frac{1}{\Delta x^2}\left(u_{i-1,j}+u_{i+1,j}\right)+\frac{1}{\Delta y^2}\left(u_{i,j-1}+u_{i,j+1}\right)-2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)u_{i,j}=q_{i,j}. \tag{7.19}$$

By comparison with the matrix form we can infer that the terms in the recursive equation (7.18) correspond to

$$\begin{aligned} u_i &= u_{i,j} \\ a_{ii} &= -2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right) \\ \sum_{j\neq i} a_{ij}u_j &= \frac{1}{\Delta x^2}\left(u_{i-1,j}+u_{i+1,j}\right)+\frac{1}{\Delta y^2}\left(u_{i,j-1}+u_{i,j+1}\right) \\ q_i &= q_{i,j} \end{aligned}$$

Thus, for the Poisson's equation the Jacobi iterations are given by

$$u_{i,j}^{k+1}=\frac{-1}{2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)}\left[q_{i,j}-\frac{1}{\Delta x^2}\left(u_{i-1,j}^{k}+u_{i+1,j}^{k}\right)-\frac{1}{\Delta y^2}\left(u_{i,j-1}^{k}+u_{i,j+1}^{k}\right)\right]. \tag{7.20}$$

The idea is that we start with an initial guess $u_{i,j}^0$ and iterate the solution until it is good enough. Defining how good is good enough is not an easy task, but we will get there.

Note that in one iteration we need to loop in the 2D space to update the solution at all spatial positions. If we loop $i=1,N_x$ and $j=1,N_y$, when we are calculating $u_{i,j}^{k+1}$ we have already calculated $u_{i-1,j}^{k+1}$ and $u_{i,j-1}^{k+1}$. If we use these updated values, our Jacobi scheme becomes the Gauss-Seidel scheme, which has a faster convergence rate. In this case, the recursive equation is

$$u_{i,j}^{k+1}=\frac{-1}{2\left(\frac{1}{\Delta x^2}+\frac{1}{\Delta y^2}\right)}\left[q_{i,j}-\frac{1}{\Delta x^2}\left(u_{i-1,j}^{k+1}+u_{i+1,j}^{k}\right)-\frac{1}{\Delta y^2}\left(u_{i,j-1}^{k+1}+u_{i,j+1}^{k}\right)\right], \tag{7.21}$$

where the only difference from (7.20) is the use of two updated values (at $(k+1)$) on the right-hand side. In addition to the faster convergence rate, using Gauss-Seidel also saves memory, as we can iterate without the need to store the old iteration.

In practice, it is easier to code these methods by defining a residual at each grid point

$$R_{i,j}=\frac{1}{\Delta x^2}\left(u_{i-1,j}-2u_{i,j}+u_{i+1,j}\right)+\frac{1}{\Delta y^2}\left(u_{i,j-1}-2u_{i,j}+u_{i,j+1}\right)-q_{i,j}. \tag{7.22}$$

Note that the residual is basically the discrete version of the PDE written as $R=\nabla^2 u-q=0$. In practice, the residual measures how far the current solution is from the true solution. If

the residual becomes zero at every grid point, we have found the solution. In practice, we just iterate until the residual is small enough.

It is easy to re-write the recursive equations in terms of the residual. For the Jacobi method, the recursive equation (7.20) becomes

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \frac{1}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} R_{i,j}^{k}. \tag{7.23}$$

For the Gauss-Seidel we write

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \frac{1}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} R_{i,j}^{k/k+1}, \tag{7.24}$$

where the index of the residual $k/k+1$ indicates that some terms are calculated based on k and others based on $k+1$.

A third approach, called successive over relaxation (SOR), is a slightly modified version of the Gauss-Seidel iteration:

$$u_{i,j}^{k+1} = u_{i,j}^{k} + \frac{\alpha}{2\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)} R_{i,j}^{k/k+1}, \tag{7.25}$$

where $1 < \alpha < 2$ is the over-relaxation coefficient (note that $\alpha = 1$ corresponds to Gauss-Seidel). SOR usually converges much faster than Gauss-Seidel, but its convergence rate depends on the value of α. We can estimate the optimal value of α using

$$\alpha_{opt} = \frac{2}{1 + \sqrt{1 - \sigma^2}}, \tag{7.26}$$

where

$$\sigma = \frac{1}{1 + \beta^2}\left[\cos\left(\frac{\pi}{N_x}\right) + \beta^2 \cos\left(\frac{\pi}{N_y}\right)\right] \tag{7.27}$$

and

$$\beta = \frac{\Delta x}{\Delta y}. \tag{7.28}$$

This optimal coefficient also depends on the choice of boundary conditions, and for periodic boundary conditions we need to replace π by 2π inside the cos. As a final note, in SOR the error sometimes increases before decreasing again. Convergence is still faster, though. This error increase can be prevented using a technique called Chebyshev acceleration.

7.1.1 A note on solvability condition for Poisson's equation

Suppose we are interested in the general Poisson's equation written as

$$\nabla^2 u = q \tag{7.29}$$

on a two-dimensional domain bounded by a closed curve S with surface normal $\mathbf{n}$.

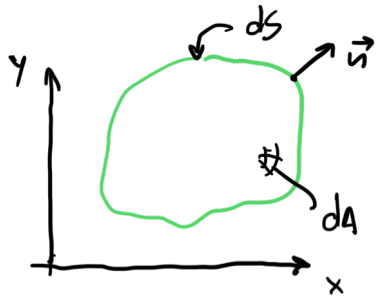

Figure 7.2: Domain illustrating solvability condition.

If we integrate the equation over the entire domain

$$\int \nabla^2 u dA = \int q dA, \tag{7.30}$$

and note that $\nabla^2 u = \nabla \cdot (\nabla u)$, so that

$$\int \nabla^2 u dA = \int \nabla \cdot (\nabla u)\, dA = \oint (\nabla u) \cdot \mathbf{n} dS = \oint \frac{\partial u}{\partial n} dS. \tag{7.31}$$

The second equality stands from Gauss theorem. Thus, the solvability condition for Poisson's equation reads

$$\oint \frac{\partial u}{\partial n} dS = \int q dA. \tag{7.32}$$

The right hand side is simply the average value of the forcing q within the domain multiplied by the total area $\bar{q}A$.

If the Poisson's equation is accompanied by Dirichlet boundary conditions, the solvability condition is not relevant, as the solution will be such that the condition above will be naturally satisfied. However, if Neumann conditions are specified, two issues must be kept in mind. If the boundary conditions do not satisfy (7.32), the problem is ill posed and a solution does not exist. If the solvability condition is satisfied, then the solution exists but it is not unique and this can cause problems in the process of numerical solution (sometimes one has to add and extra condition that makes the solution unique). The last case worth mentioning is the case of periodic (or cyclic) boundary conditions. In this case, the integral

on the left-hand side of (7.32) is identically zero, and the problem can only be solved if the forcing q also integrates to zero in the domain (or equivalently, its average within the domain $\overline{q} = 0$).

Chapter 8

Nonlinear advection

8.1 Burgers' equation and aliasing errors

The last step before we can solve the barotropic QG vorticity equation is to discuss the effects of nonlinear advection. The critical question is in the differences between the linear advection problem

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0 \tag{8.1}$$

and the nonlinear case

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0. \tag{8.2}$$

This nonlinear problem is the one-dimensional inviscid Burgers' equation (note that now, u must be a velocity). The viscous Burgers' equation is given by

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial x^2}. \tag{8.3}$$

In this chapter we will use the inviscid Burgers' equation to explore some concepts associated with nonlinear advection. Our approach will be far less rigorous than in the previous chapters. Note that the inviscid Burgers' equation is a conservative equation (it conserves energy), while this is not the case for the viscous version. (Just as a curiosity, Burgers' equation can be transformed into a linear diffusion equation using the Cole-Hopf transformation.)

Suppose we are interested in problems (8.1) and (8.2) with the initial condition $u(x, t = 0) = \sin(kx)$. Note that this initial condition has only one wavenumber (k), and all the kinetic energy is contained at this wavenumber. If we advance the solution using the linear advection equation (8.1), the sine wave will propagate in space but the solution will still have only one wavenumber. Numerical errors (e.g., the computational mode of the leapfrog scheme) may create additional spurious wavenumbers, but as long as we satisfy the stability criterion this will not cause big problems.

If instead we use the nonlinear advection equation (8.2), things are drastically different

(even for the exact solution). At the initial time, the nonlinear term yields

$$\left(u\frac{\partial u}{\partial x}\right)\Big|_{t=0} = \sin(kx)k\cos(kx) = \frac{1}{2}k\sin(2kx). \tag{8.4}$$

Thus, some of the energy that is initially associated with the wavenumber k is transferred to the the wavenumber $2k$ (i.e., to smaller scales as the wavelength is half). This is one of the key effects of nonlinear advection: it transfers energy to increasingly smaller scales. As shown in the figure below, the progressive transfer of energy to smaller scales is associated with a steepening of the gradients in the solution (note that we need a larger number of Fourier modes to represent the steep gradients that form). In the inviscid Burgers equation, this steepening eventually leads to the formation of shock waves, which are a nightmare for numerical solutions. In the viscous Burgers equation, viscous diffusion limits the maximum steepness possible and dictates the mesh requirements as illustrated in Fig. 8.1.

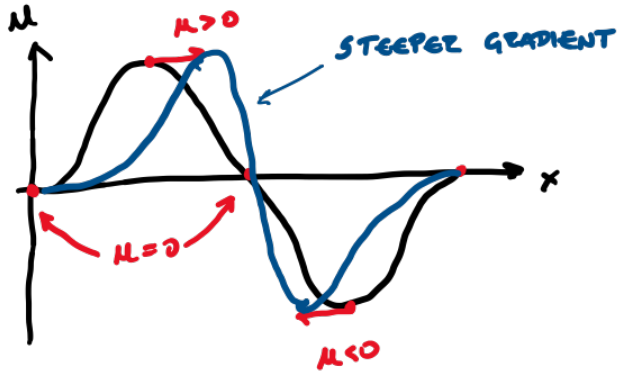

Figure 8.1: Solution of Burgers' equation.

The main problem caused by the nonlinearity is that for a grid of size Δx, the smallest resolved wavelength is $\lambda_x^{\min} = 2\Delta x$ (or the largest wavenumber $k_x^{\max} = \pi/\Delta x$). During the advancement of the solution in time, energy associated with wavelengths smaller than $\lambda_x^{\min}$ appears associated with longer wavelengths. This is called *aliasing error*.

As an example, for the simple initial condition discussed above, suppose that we have all the energy at $\lambda_x = (8/3)\Delta x$. In this case, the nonlinear term will transfer some energy to $\lambda_x = (4/3)\Delta x < \lambda_x^{\min}$. These wavelengths are illustrated in the Fig. 8.2. Note that the energy at $\lambda_x = (4/3)\Delta x$, when represented on the discrete grid, appears to be at $\lambda_x = 4\Delta x$.

So the energy that was supposed to be at $k = (3/2)k^{\max}$ (which cannot be represented on the grid) is aliased onto $k = (1/2)k^{\max}$ (which can be represented on the grid). In general, we can show that the energy at any wavenumber $\hat{k} > k^{\max}$ is aliased onto $k^* = 2k^{\max} - \hat{k}$. This is a symmetric reflection about the maximum wavenumber that can be represented on the discrete grid as illustrated in Fig. 8.3.

There are two main issues associated with aliasing errors. The first is that it causes

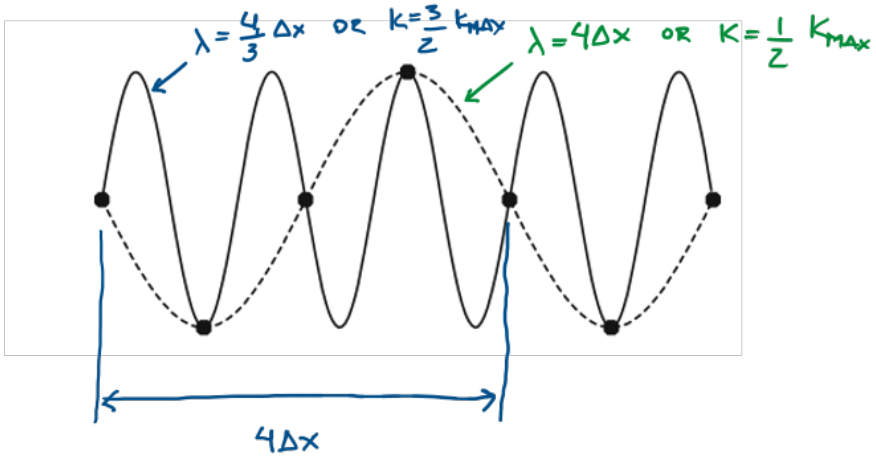

Figure 8.2: Misrepresentation of $\lambda_x = (4/3)\Delta x$ on a discrete grid.

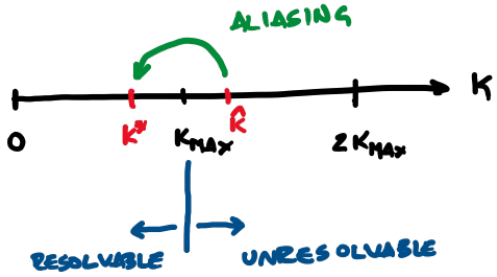

Figure 8.3: Aliasing in wavenumber space.

energy to accumulate in shorter waves and with time this error grows and dominates the solution. The second is that it produces nonlinear instabilities which, in the absence of enough dissipation at the shorter wavelengths, causes the solution to blow up. The nonlinear instability manifests itself in the same way as the linear stability we discussed in the previous chapters, but it is not related to the amplification factor and it cannot be controlled by reducing Δt.

8.1.1 Controlling nonlinear instability in the Burgers' equation

One possible approach to control the nonlinear instability is to add artificial viscosity to the inviscid Burgers' equation to help dissipate the the aliased energy

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu_{art}\frac{\partial^2 u}{\partial x^2} \tag{8.5}$$

This looks just like the viscous Burgers' equation, except that the value of the artificial viscosity ν_{art} is not determined by physics, but rather selected to dissipate the energy needed to maintain stability. Suppose we discretize this equation using leapfrog for advection and

FTCS with $2\Delta t$ for the diffusion. The amplification factor for the diffusion part will be

$$|\mu|^2 = 1 + \frac{2\nu_{art}\Delta t}{\Delta x^2}[2\cos(k\Delta x) - 2]. \tag{8.6}$$

The fact that μ is smaller for large k is good, because the artificial viscosity will damp energy in the large k faster than in the low k. Recall that the aliasing will occur most in the large k.

Now suppose we want to completely remove all the energy at $k_{max} = \pi/\Delta x$ in one single time step (this is excessive, but it gives us an idea of how to choose the value of ν_{art}). In this case, if we set $\mu = 0$ for $k = \pi/\Delta x$ into (8.6) we get

$$0 = 1 + \frac{2\nu_{art}\Delta t}{\Delta x^2}[2\cos(\pi) - 2] \Rightarrow \nu_{art} = \frac{\Delta x^2}{8\Delta t}. \tag{8.7}$$

Just as a guideline, in the 2D case this would become

$$\nu_{art} = \frac{(1/\Delta x^2 + 1/\Delta y^2)^{-1}}{8\Delta t}. \tag{8.8}$$

A relevant question is the impact of this artificial diffusion on the longer waves we are interested in. For example, if we take a wavelength $\lambda = 8\Delta x$, corresponding to $k = \pi/(4\Delta x)$, we have $+\mu|^2 \approx 0.68$ or $|\mu| \approx 0.83$. This means that we will loose 17% of the energy (amplitude) per time step for this mode (which is a bit too much). There is some art and a lot of trial-and-error in finding the ideal value for ν_{art}. Sometimes it is better to use an artificial hyperviscosity (e.g., with a fourth-order derivative)

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu_{art}\frac{\partial^4 u}{\partial x^4}. \tag{8.9}$$

The hyperviscosity acts predominantly on the very short wavelengths, having a smaller impact on the resolved features of interest in the solution. Sometimes, in order to avoid damping the main features in the solution, hyperviscosity with eighth-order derivatives are employed in practice.

However, the most common approach to controlling aliasing errors and the consequent nonlinear instability is by designing better discretization approaches for the nonlinear term. This can be usually achieved by writing the governing equation in its conservative form (or flux form) given by

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u^2}{2}\right) = 0. \tag{8.10}$$

Even though the two PDEs (8.2) and (8.10) and their exact solutions are the same, the resulting discrete equations and approximate solutions will be different. If we use a centered

scheme, the non-conservative form yields

$$\frac{\partial u_i}{\partial t} = -u_i \frac{(u_{i+1} - u_{i-1})}{2\Delta x}, \tag{8.11}$$

while the semi-discrete equation for the conservative form is

$$\frac{\partial u_i}{\partial t} = -\frac{(u_{i+1}^2 - u_{i-1}^2)}{4\Delta x}. \tag{8.12}$$

Obviously the two discrete solutions will be different, and we can show that neither conserves energy (for this reason, Durran prefers to use the terminology "flux form" instead of "conservative form"; I disagree). What is more interesting is that one can actually develop a scheme that does conserve energy, by adopting a weighted average of the two forms of the PDE given by

$$\frac{\partial u}{\partial t} + \frac{1}{3}\left[u\frac{\partial u}{\partial x}\right] + \frac{2}{3}\left[\frac{\partial}{\partial x}\left(\frac{u^2}{2}\right)\right] = \nu\frac{\partial^2 u}{\partial x^2}. \tag{8.13}$$

This is sometimes called the "Arakawa" approach, because it is inspired by a discretization of advection that we will discuss in the next chapter. We can show that a centered difference discretization produces a discrete system that does conserve energy.

In general, for small values of ν (in the viscous Burgers' equation), direct discretization of (8.3) leads to excessive aliasing errors and nonlinear instability even for very small Δt. Discretization of the conservative form (8.10) and the alternative form (8.13) leads to schemes that develop spurious oscillations but are stable. The oscillations tend to be weaker in the conservative form (which is my preferred approach).

8.2 Nonlinear advection in the barotropic QG equations

Our final barotropic QG system in vorticity streamfunction formulation is given by

$$\frac{\partial \zeta_g}{\partial t} + \mathbf{v}_g \cdot \nabla \zeta_g = -\beta v_g - \frac{f_0 g}{R_d T_s}\mathbf{v}_g \cdot \nabla z_s \tag{8.14}$$

$$\nabla^2 \psi = \zeta_g, \tag{8.15}$$

complemented by the relations between the streamfunction and the velocity given by (6.30). So we have reduced our system to one single prognostic equation for vorticity and one Poisson equation for the streamfunction. The last point that requires discussion is the discretization of the nonlinear advection term in the vorticity equation and the issue of nonlinear instability. As suggested in the treatment of Burgers' equation, one could certainly add a diffusive (or hyperdiffusive) term in the vorticity equation to control nonlinear instability. However, another option is to take proper care in the discretization of the nonlinear term (just as we

did in the discretization of Burgers' equation). Even though we will base our discussion on the vorticity-streamfunction formulation formulation for the QG system, the same approach applies to the two-dimensional Navier-Stokes equations.

We start by writing the nonlinear advection term in Eq. (8.14) using the streamfunction

$$\mathbf{v}_g \cdot \nabla \zeta_g = u_g \frac{\partial \zeta_g}{\partial x} + v_g \frac{\partial \zeta_g}{\partial y} = -\frac{\partial \psi}{\partial y}\frac{\partial \zeta_g}{\partial x} + \frac{\partial \psi}{\partial x}\frac{\partial \zeta_g}{\partial y} = J(\psi, \zeta_g). \tag{8.16}$$

Here we have defined the Jacobian operator

$$J(a,b) = \frac{\partial a}{\partial x}\frac{\partial b}{\partial y} - \frac{\partial a}{\partial y}\frac{\partial b}{\partial x}. \tag{8.17}$$

Direct discretization of the advection in this form leads to nonlinear instability.

Note that in the absence of β-effects and topography, Eq. (8.14) is a conservation equation for the vertical vorticity. We can show that in this case, the system should conserve several properties, two of which are important here: the total kinetic energy and the total enstrophy in the domain (enstrophy is a property of the vorticity field that is analogous to the kinetic energy; in the present case the total enstrophy is ζ_g^2). In this system, ensuring conservation of the total kinetic energy is enough to guarantee that the scheme is stable.

Using the same approach we used for Burgers' equation, we note that the Jacobian can be written in two other conservative forms

$$J(a,b) = \underbrace{\frac{\partial}{\partial x}\left(a\frac{\partial b}{\partial y}\right) - \frac{\partial}{\partial y}\left(a\frac{\partial b}{\partial x}\right)}_{=\hat{J}(a,b)} = \underbrace{\frac{\partial}{\partial y}\left(b\frac{\partial a}{\partial x}\right) - \frac{\partial}{\partial x}\left(b\frac{\partial a}{\partial y}\right)}_{=\tilde{J}(a,b)}. \tag{8.18}$$

Note that although all three forms are identical, their discrete versions are not. Neither one of the three forms of the Jacobian conserves kinetic energy when discretized with centered differences in space. However, the "Arakawa discretization" of the Jacobian, which consists of discretizing with a second order centered scheme the average of the three forms

$$J_a(a,b) = \frac{1}{3}\left[J(a,b) + \hat{J}(a,b) + \tilde{J}(a,b)\right], \tag{8.19}$$

conserves both the kinetic energy and enstrophy. In addition, we can show that this discretization also inhibits the accumulation of energy at the small scales, yielding fairly smooth solutions. In this sense, this is the best possible discretization with second-order accuracy.

Note that in the QG system, the forcing term on the vorticity equation associated with the topography can also be written using the Jacobian

$$\frac{\partial \zeta_g}{\partial t} + J(\psi, \zeta_g) = -\beta v_g - \frac{f_0 g}{R_d T_s} J(\psi, z_s). \tag{8.20}$$

This term is a linear term, so it does not require special treatment. You can choose between treating it with a standard centered difference in space or with the Arakawa Jacobian.

Chapter 9

Buoyancy-driven flows

We now return to the primitive equations from Chapter 1, but we make an extra assumption about density fluctuations that is usually termed the *Boussinesq approximation*. In summary, we write the density as a sum of a reference part (that is constant) and deviations from this reference state

$$\rho(\mathbf{x}, t) = \rho_0 + \rho'(\mathbf{x}, t), \tag{9.1}$$

and then make the assumption that

$$\frac{\rho'}{\rho_0} \ll 1. \tag{9.2}$$

This is usually applicable in the lower portion of the atmosphere (e.g., lowest 2 km).

Without going into details, this assumptions allows one to simplify the the primitive equations considerably. The conservation of mass given by

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{u} = 0 \tag{9.3}$$

is replaced by the condition that the velocity field be divergence-free

$$\nabla \cdot \mathbf{u} = 0. \tag{9.4}$$

We can also show that, in the momentum equations originally written as

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla p + \mathbf{g} - 2\boldsymbol{\Omega} \times \mathbf{u} + \nu \nabla^2 \mathbf{u}, \tag{9.5}$$

density fluctuations must be retained only in the buoyancy term

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho_0}\nabla p + \frac{\rho'}{\rho_0}\mathbf{g} - 2\boldsymbol{\Omega} \times \mathbf{u} + \nu \nabla^2 \mathbf{u}. \tag{9.6}$$

For the atmosphere we can express density fluctuations in terms of the potential temperature

$$\frac{\rho'}{\rho_0} \approx -\frac{\theta'}{\theta_0}. \tag{9.7}$$

Potential temperature is a conserved quantity, so we can express the primitive equations as

$$\nabla \cdot \mathbf{u} = 0, \tag{9.8}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho_0}\nabla p - \frac{\theta'}{\theta_0}\mathbf{g} - 2\mathbf{\Omega} \times \mathbf{u} + \nu\nabla^2\mathbf{u}, \tag{9.9}$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla\theta = \gamma\nabla^2\theta. \tag{9.10}$$

From here on, we will consider a two-dimensional version of these equations on an $x - z$ plane and set $v = 0$. The idea is that we will be simulating flows that do not change in the y-direction, so simulating any plane is enough to capture the entire physics. Note that in 2D the Coriolis force disappears. For now we will also neglect the viscous effects, as we know it is fairly simple to add them back on later if needed. Of course, the 2D assumption is also a mathematical convenience, because in 2D we can use the vorticity-streamfunction formulation. Our starting point is the set of equations

$$\frac{\partial u}{\partial x} + \frac{\partial w}{\partial z} = 0, \tag{9.11}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x}, \tag{9.12}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho_0}\frac{\partial p}{\partial z} + \frac{\theta'}{\theta_0}g, \tag{9.13}$$

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} + w\frac{\partial \theta}{\partial z} = 0. \tag{9.14}$$

The main difficulty with this set of equations is that there is no prognostic equation for pressure. We will discuss two very distinctive approaches to discretize the 2D Boussinesq equation system above.

9.1 The vorticity-streamfunction approach

Here the idea is to write an equation for the vorticity as this will eliminate the pressure (as in the QG system) and solve the issue that we do not have a prognostic equation for pressure. Also as in the QG system, we will introduce a streamfunction to satisfy the conservation of mass. We will see that the final set of equations is not too different from the one obtained for the QG system.

Let's start by defining the y-component of the vorticity

$$\omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x}. \tag{9.15}$$

The prognostic equation for ω_y can be obtained from the Boussinesq system and is given by

$$\frac{\partial \omega_y}{\partial t} + u\frac{\partial \omega_y}{\partial x} + w\frac{\partial \omega_y}{\partial z} = -\frac{g}{\theta_0}\frac{\partial \theta}{\partial x}. \tag{9.16}$$

Note that since θ_0 is constant, we replaced θ' by θ on the right hand side. The term including θ is the baroclinic production of vorticity, which is very important in geophysical fluids.

Next we define a streamfunction on the $x - z$ plane via

$$u = -\frac{\partial \psi}{\partial z} \quad \text{and} \quad w = \frac{\partial \psi}{\partial x}. \tag{9.17}$$

The vorticity equation can now be written in terms of the streamfunction as

$$\frac{\partial \omega_y}{\partial t} - \frac{\partial \psi}{\partial z}\frac{\partial \omega_y}{\partial x} + \frac{\partial \psi}{\partial x}\frac{\partial \omega_y}{\partial z} = -\frac{g}{\theta_0}\frac{\partial \theta}{\partial x}, \tag{9.18}$$

or, using the Jacobian,

$$\frac{\partial \omega_y}{\partial t} + J(\psi, \omega_y) = -\frac{g}{\theta_0}\frac{\partial \theta}{\partial x}. \tag{9.19}$$

Similarly, the equation for the potential temperature is written as

$$\frac{\partial \theta}{\partial t} + J(\psi, \theta) = 0. \tag{9.20}$$

Finally, from the definition of the vorticity we can write

$$\omega_y = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} = -\frac{\partial^2 \psi}{\partial z^2} - \frac{\partial^2 \psi}{\partial x^2}, \tag{9.21}$$

which yields the Poissons' equation for the streamfunction

$$\nabla^2 \psi = -\omega_y. \tag{9.22}$$

Note the negative sign in front of the vorticity, which is different from the equation for the QG system because we switched sign in the definition of the streamfunction.

Our final form of the 2D Boussinesq system in vorticity-streamfunction formulation is

$$\frac{\partial \omega_y}{\partial t} = -J(\psi, \omega_y) - \frac{g}{\theta_0}\frac{\partial \theta}{\partial x}, \tag{9.23}$$

$$\frac{\partial \theta}{\partial t} = -J(\psi, \theta), \tag{9.24}$$

$$\nabla^2 \psi = -\omega_y. \tag{9.25}$$

Note that this is very similar to the QG barotropic system, except that we need to solve one additional equation for θ.

9.2 The velocity-pressure formulation

The second approach is to solve the 2D Boussinesq equation system in its original velocity and pressure variables. Even though this is a bit more difficult than using the vorticity-streamfunction approach, it is more general and can be extended to 3D problems as well. In order to do that, we need to deal with the issue that we do not have an equation to determine the time evolution of the pressure field and we also do not have boundary conditions for pressure (boundary conditions for the velocity field are always clearly imposed by the physical setting of the problem, but those for pressure are not).

The first issue can be addressed by manipulating the equations of motion in order to derive an equation for the pressure field. If we take the divergence of the momentum equation (9.9) we have

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t}\right) + \nabla \cdot [(\mathbf{u} \cdot \nabla)\mathbf{u}] = -\frac{1}{\rho_0}\nabla \cdot (\nabla p) - \frac{1}{\theta_0}\nabla \cdot (\theta' \mathbf{g}) - 2\nabla \cdot (\mathbf{\Omega} \times \mathbf{u}) + \nu \nabla \cdot (\nabla^2 \mathbf{u}). \tag{9.26}$$

Using operations from vector calculus and the conservation of mass (9.8), it is easy to show that many terms in the equation above vanish. As an example,

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t}\right) = \frac{\partial}{\partial t}(\nabla \cdot \mathbf{u}) = 0. \tag{9.27}$$

The only three non-zero terms are the ones involving the nonlinear advection, the pressure-gradient force, and the buoyancy term. The resulting equation is

$$\nabla^2 p = -\rho_0 \nabla \cdot [(\mathbf{u} \cdot \nabla)\mathbf{u}] - \frac{\rho_0}{\theta_0}\mathbf{g} \cdot \nabla \theta'. \tag{9.28}$$

The is the Poissons' equation for pressure (PPE). Note that this equation is only valid if (9.8) is satisfied. Thus, the PPE is equivalent to imposing $\nabla \cdot \mathbf{u} = 0$, and we can replace

(9.8) by (9.28). Our Boussinesq system becomes (neglecting viscous terms)

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u} = -\frac{1}{\rho_0}\nabla p - \frac{\theta'}{\theta_0}\mathbf{g} - 2\boldsymbol{\Omega}\times\mathbf{u}, \tag{9.29}$$

$$\nabla^2 p = -\rho_0 \nabla\cdot[(\mathbf{u}\cdot\nabla)\mathbf{u}] - \frac{\rho_0}{\theta_0}\mathbf{g}\cdot\nabla\theta', \tag{9.30}$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u}\cdot\nabla\theta = 0. \tag{9.31}$$

Once again, if we write the 2D system on an $x-z$ plane we now have

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x}, \tag{9.32}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial x} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho_0}\frac{\partial p}{\partial z} + \frac{\theta'}{\theta_0}g, \tag{9.33}$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial z^2} = -\rho_0\left[\left(\frac{\partial u}{\partial x}\right)^2 + 2\frac{\partial u}{\partial x}\frac{\partial w}{\partial z} + \left(\frac{\partial w}{\partial z}\right)^2\right] + \frac{\rho_0 g}{\theta_0}\frac{\partial \theta'}{\partial z} \tag{9.34}$$

$$\frac{\partial \theta}{\partial t} + u\frac{\partial \theta}{\partial x} + w\frac{\partial \theta}{\partial z} = 0. \tag{9.35}$$

This manipulation of the equations has solved a critical problem: now we can calculate the pressure field by solving the PPE equation. There is still one big challenge in solving this system: if p satisfies the PPE, then the velocity in the momentum equation will satisfy $\nabla\cdot\mathbf{u} = 0$, but the PPE is only valid if $\nabla\cdot\mathbf{u} = 0$. So, we cannot use any simple explicit method, as we need to solve the momentum equations and the PPE simultaneously. We will address this a bit later. For now we will focus on the spatial discretization of the 2D equations.

The first important decision in the spatial discretization is the set up of the grid. The obvious choice is to continue using the approach we have been using so far, and define all variables on the same locations (nodes). This is also called a textitcollocated grid, and in the geosciences community it is referred to as the Arakawa A grid. This option is always used with nodes defined on the boundaries (see Figure 9.1 below), as we have been doing so far. The main advantages of using a collocated grid are its simplicity and the fact that all variables are defined on all boundaries, which is helpful for enforcing boundary conditions. The disadvantages of this choice are also important: (i) because the pressure is defined on the boundaries, it is harder to impose boundary conditions for the PPE (more on this later), (ii) there are numerical issues in the coupling between pressure and velocity, and this setup introduces very large dispersive errors in the propagation of inertial and gravity waves, which are very important in geophysical problems.

There are many other possible configurations based on staggered arrangements, in which not all variables are defined at the same locations (these are called Arakawa B, C, D, and E

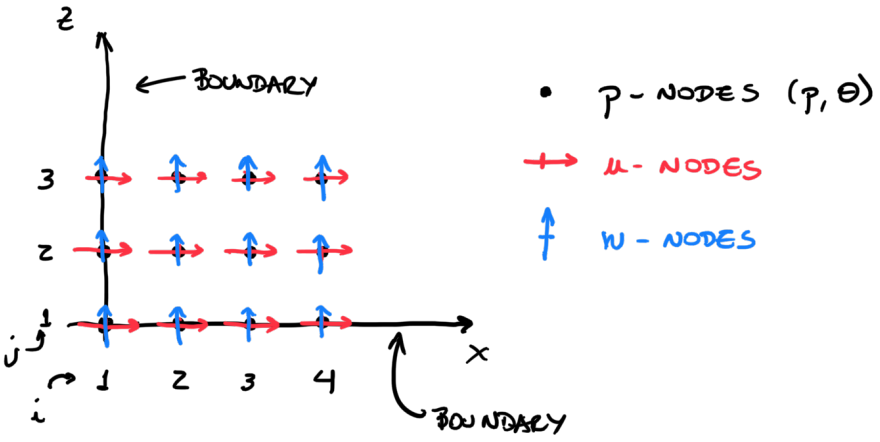

Figure 9.1: Arakawa A grid.

grids). From the staggered grids, the most commonly used is the Arakawa C grid, which is also called the marker-and-cell (or MAC) grid. There are two possible distribution of variables following this grid (see Figure 9.2), and the choice depends on the boundary conditions to be enforced. The main advantage of the Arakawa C grid is that it is excellent for enforcing conservation of mass. We can also show that this grid leads to much smaller errors in the propagation of waves and that most errors restricted to very short waves. However, only one component of velocity is defined on each boundary (which is a problem for enforcing boundary conditions). In addition, evolving the solution typically requires a lot of interpolations because variables are not always defined where they are needed. Finally, something not to be dismissed, is that staggered grids in general require excellent bookkeeping.

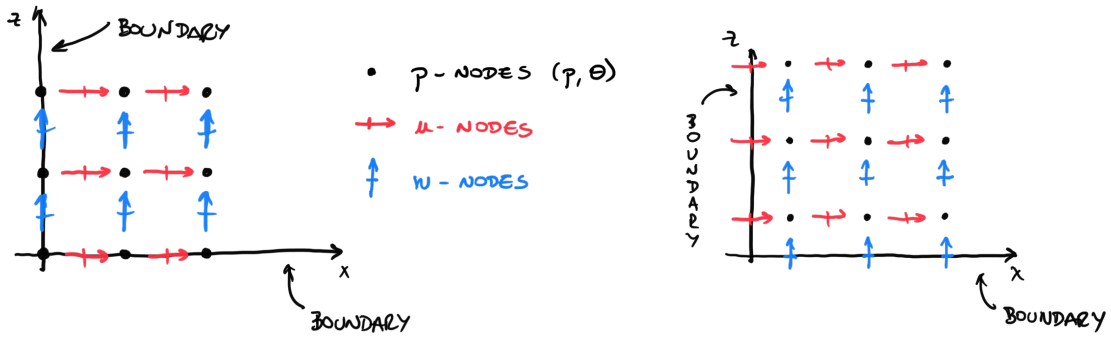

Figure 9.2: Two possible configurations for the Arakawa C grid.

One of the challenges of staggered grid is the node numbering. As an example, suppose we adopt the standard node numbering illustrated on Figure 9.3. Note that each equation is discretized at different locations. So equations (9.34) and (9.35) are discretized on p-nodes, equation (9.32) is discretized on u-nodes, and equation (9.33) is discretized on w-nodes. In general, when variables are needed in location where they are not available, we can simply

use bi-linear interpolations.

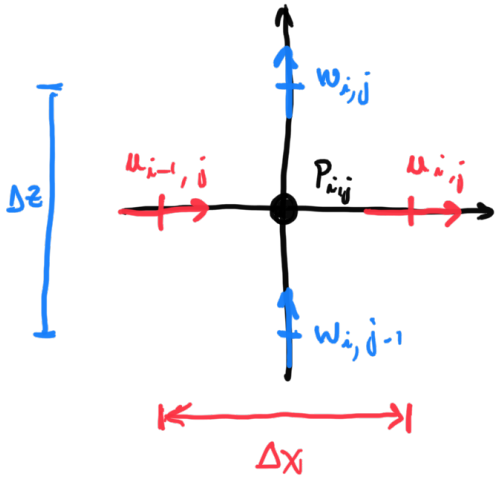

Figure 9.3: Node numbering for the Arakawa C grid.

One has to be very careful with this index convention, as $u_{i,j}$, $w_{i,j}$, and $p_{i,j}$ are all at different locations (even though their indices are the same). As an example, let's focus on the spatial discretization of the momentum equation for u given by (9.32) and written as

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x} - w\frac{\partial u}{\partial z} - \frac{1}{\rho_0}\frac{\partial p}{\partial x} \tag{9.36}$$

Following the convention from Fig. 9.3, this equation must be discretized at u-nodes. Let's write down the equation for $u_{i,j}$:

$$\begin{aligned}\frac{\partial u_{i,j}}{\partial t} = &- u_{i,j}\frac{1}{2\Delta x}(u_{i+1,j} - u_{i-1,j}) \\ &- \frac{1}{4}(w_{i,j-1} + w_{i,j} + w_{i+1,j-1} + w_{i+1,j})\frac{1}{2\Delta z}(u_{i,j+1} - u_{i,j-1}) \\ &- \frac{1}{\rho_0}\frac{1}{\Delta x}(p_{i+1,j} - p_{i,j}).\end{aligned} \tag{9.37}$$

Note that w is not available at the location where the equation is being discretized, so we must use bi-linear interpolation. In addition, the centered difference for the velocity derivatives uses a spacing of $2\Delta x$, but we can do centered differencing for the pressure with Δx due to the staggered grid. The resulting stencil is illustrated in Figure 9.4

The equation for the vertical momentum equation (9.33) and the temperature advection equations are very similar, and also require some interpolations. One of the big advantages of the Arakawa C grid is in the conservation of mass, and it applies both to the usual

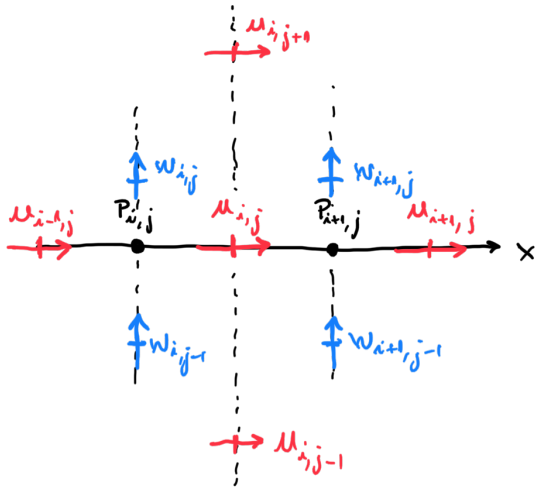

Figure 9.4: Stencil for the u-momentum equation on the Arakawa C grid.

form (9.11) and the PPE form (9.30). Let's start with (9.11), which must be discretized on p-nodes. We have

$$\frac{1}{\Delta x}(u_{i,j} - u_{i-1,j}) + \frac{1}{\Delta z}(w_{i,j} - w_{i,j-1}) = 0. \tag{9.38}$$

Note that no interpolations are required, and both discretizations are centered in space with spacings Δx and Δz. Similarly, if we discretize directly the PPE we have

$$\begin{aligned} &\frac{1}{\Delta x^2}\left(p_{i-1,j} - 2p_{i,j} + p_{i+1,j}\right) + \frac{1}{\Delta z^2}\left(p_{i,j-1} - 2p_{i,j} + p_{i,j+1}\right) = \\ -\rho_0 &\left[\frac{1}{\Delta x^2}(u_{i,j} - u_{i-1,j})^2 + \frac{2}{\Delta x \Delta z}(u_{i,j} - u_{i-1,j})(w_{i,j} - w_{i,j-1})\right. \\ &\left. + \frac{1}{\Delta z^2}(w_{i,j} - w_{i,j-1})^2\right] + \frac{\rho_0 g}{2\theta_0 \Delta z}(\theta'_{i,j+1} - \theta'_{i,j-1}) \end{aligned} \tag{9.39}$$

Once again, no interpolations are needed, and the largest error in this term comes from the centered difference of the temperature derivative (using $2\Delta z$).

Now that we have discussed the details of the spatial discretization, we need to deal with the time discretization and the issues associated with velocity-pressure coupling and the critical constraint $\nabla \cdot \mathbf{u} = 0$. We will briefly discuss two approaches here.

9.2.1 The MAC method

The MAC method was originally developed by Harlow and Welch at Los Alamos in 1965, and then (independently?) by Arakawa and Lamb in 1977 at UCLA. The basic idea is to discretize the Navier-Stokes equations (or in our case the 2D Boussinesq equations) on the MAC grid (Arakawa C) in original form (9.8)-(9.10) or (9.11)-(9.14), and then combine the discrete equations to produce a discrete PPE. The method uses second-order centered

differences in space. Based on the comments of the previous section, we note that pressure must be treated implicitly in order to ensure conservation of mass. A simple approach is to use BTCS for pressure and FTCS for all other terms (you could easily use leapfrog or some other more accurate scheme for the other terms, but the treatment of pressure must be implicit).

As an example, we will add the time discretization to the the x-momentum equation (9.37):

$$\begin{aligned} u_{i,j}^{n+1} = u_{i,j}^n + \Delta t \Bigg[& -u_{i,j}^n \frac{1}{2\Delta x}(u_{i+1,j}^n - u_{i-1,j}^n) \\ & - \frac{1}{4}(w_{i,j-1}^n + w_{i,j}^n + w_{i+1,j-1}^n + w_{i+1,j}^n)\frac{1}{2\Delta z}(u_{i,j+1}^n - u_{i,j-1}^n) \\ & - \frac{1}{\rho_0}\frac{1}{\Delta x}(p_{i+1,j}^{n+1} - p_{i,j}^{n+1}) \Bigg]. \end{aligned} \tag{9.40}$$

The equation for the vertical velocity is similar to (9.40).

Finally, conservation of mass is discretized at p-nodes "implicitly" as

$$\left(\frac{\partial u}{\partial x}\right)_{i,j}^{n+1} + \left(\frac{\partial w}{\partial z}\right)_{i,j}^{n+1} = \frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x} + \frac{w_{i,j}^{n+1} - w_{i,j-1}^{n+1}}{\Delta z} = 0. \tag{9.41}$$

Note that (9.41) involves $u_{i,j}^{n+1}$ and $u_{i-1,j}^{n+1}$. While we have an equation for the former (9.40), we can also obtain an equation for the latter by simply replacing i by $i-1$ in (9.40). Replacing both equations (and the equations for w) into (9.41) we obtain a discrete PPE in which the forcing (i.e., the right hand side) is explicit. In short notation:

$$\frac{1}{\Delta x^2}(p_{i-1,j}^{n+1} - 2p_{i,j}^{n+1} + p_{i+1,j}^{n+1}) + \frac{1}{\Delta z^2}(p_{i,j-1}^{n+1} - 2p_{i,j}^{n+1} + p_{i,j+1}^{n+1}) = -\rho_0 [\cdots] = q_{i,j}^n \tag{9.42}$$

All that we need to do is to solve a Poissons' equation to obtain the pressure.

Thus, the time stepping algorithm for the MAC method is composed of two parts:

1. given u^n and w^n, solve for p^{n+1} using the discrete PPE (9.42)
2. given u^n, w^n, and p^{n+1}, solve for u^{n+1} and w^{n+1} using the discrete momentum equations (9.40) and similar equation for w

The critical idea point is that in the conservation of mass we related p^{n+1} to u^{n+1} and w^{n+1}.

9.3 The projection method

The projection method consists of the application of the operator splitting technique to the Navier-Stokes equation, separating the entire system of PDEs into two parts. The

idea, introduced by Chorin in 1968, is to separate the nonlinear advection which must be treated explicitly from the pressure gradient and continuity, which must be treated implicitly. Buoyancy (and viscous diffusion if present) is included in the explicit part. This separation is shown schematically in the figure below.

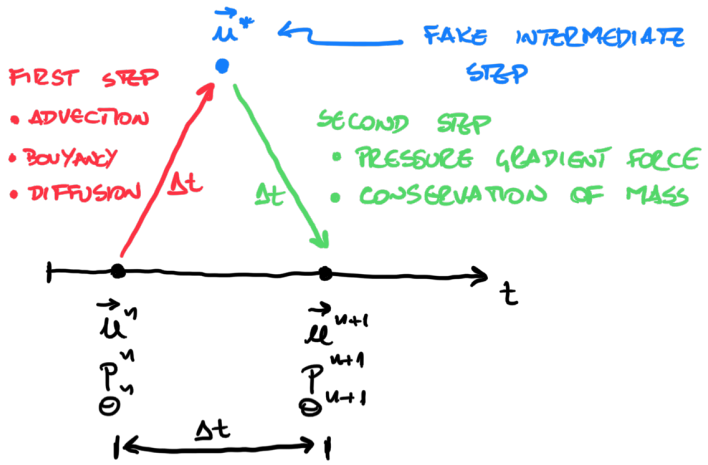

Figure 9.5: Operator splitting for the projection method.

For the first step we use an explicit scheme. For simplicity, we adopt the Euler-forward scheme in this example.

$$\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} = -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n - \frac{\theta^n}{\theta_0}\mathbf{g} + \nu\nabla^2\mathbf{u}^n. \tag{9.43}$$

Note that because the nonlinear term is calculated in this first step, this is the place where using a better time advancement scheme is valuable (e.g., AB3 or RK4).

The second step uses an implicit scheme to enforce conservation of mass. Here we use Euler-Backward as an example

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t} = -\frac{1}{\rho_0}\nabla p^{n+1} \tag{9.44}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0. \tag{9.45}$$

This second step actually projects the intermediate solution $\mathbf{u}^*$ onto a subspace of solutions that are divergence-free (this is why the method is called the projection method).

In practice, we use the system above to obtain a PPE valid for the second operator by taking the divergence of (9.44) to obtain

$$\nabla \cdot \left[\frac{\mathbf{u}^{n+1} - \mathbf{u}^*}{\Delta t}\right] = -\frac{1}{\rho_0}\nabla^2 p^{n+1}. \tag{9.46}$$

Noting that $\nabla \cdot \mathbf{u}^* \neq 0$, we obtain a different PPE equation given by

$$\nabla^2 p^{n+1} = \frac{\rho_0}{\Delta t} \nabla \cdot \mathbf{u}^*. \tag{9.47}$$

The boundary condition for pressure comes from projecting (9.44) onto the direction normal to the boundary

$$\left(\frac{\partial p}{\partial n} \right)_\Gamma^{n+1} = -\frac{\rho_0}{\Delta t} \left(\mathbf{u}_\Gamma^{n+1} - \mathbf{u}_\Gamma^* \right) \cdot \hat{\mathbf{n}}. \tag{9.48}$$

Here, $\mathbf{u}_\Gamma^{n+1}$ is the true boundary condition and $\mathbf{u}_\Gamma^*$ is an intermediate condition that is spurious in the sense that it has no analogue in the continuous system. However, for this specific formulation the solution does not depend on the choice for $\mathbf{u}_\Gamma^*$. For the first step, the solution in the interior nodes does not depend on the choice of $\mathbf{u}_\Gamma^*$ because this step is explicit and the solution in the interior nodes is entirely determined based on $\mathbf{u}_\Gamma^n$. For the second step, $\mathbf{u}_\Gamma^*$ appears to the forcing of the equation and on the Neumann boundary condition, and the two cancel out. In this case, we can choose anything for $\mathbf{u}_\Gamma^*$, and the best choice is to set $\mathbf{u}_\Gamma^* = \mathbf{u}_\Gamma^{n+1}$ because this simplifies the Neumann boundary condition for the pressure to

$$\left(\frac{\partial p}{\partial n} \right)_\Gamma^{n+1} = 0. \tag{9.49}$$

The projection method can be used with any grid arrangement, but it is usually most efficient and accurate on the MAC grid. The time stepping algorithm for the projection method is composed of three parts:

1. given $\mathbf{u}^n$, solve for $\mathbf{u}^*$ using the first step (9.43)
2. given $\mathbf{u}^*$, solve for p^{n+1} using the PPE (9.47) and the boundary condition (9.49)
3. given $\mathbf{u}^*$ and p^{n+1}, solve for $\mathbf{u}^{n+1}$ using second step of the momentum equations (9.44)

Chapter 10

A note on boundary conditions

Boundary conditions are a very important part of our numerical methods. So far, we have avoided the issue of boundary conditions by specifying from the beginning periodic boundary conditions for the solution. At some point we introduced walls as boundary conditions, and we briefly discussed some of the issues that may arise in the solution of Poisson's equation, but we never really had a a more systematic discussion of appropriate conditions.

In general, improper specification of boundary conditions can lead to problems that do not have solution or problems in which the solution is not unique. A *well-posed* boundary value problem has a solution that is unique and depends continuously on the boundary conditions. For ill-posed problems, there are three possible situations: (i) if we specify too many initial/boundary conditions, a solution does not exist; (ii) if we specify too few initial/boundary conditions, the solutions exists but it is not unique; and if we specify boundary conditions at the incorrect boundary, the solution exists and it is unique, but it will not depend smoothly on the boundary conditions. In all 3 cases, the numerical solution will likely diverge.

If we go back to our original, simple PDEs (before we dove into nonlinear advection), we can list the required initial/boundary conditions for well-posed problems. Elliptic equations (e.g., Poisson's equation) require on set of boundary conditions on each boundary (recall also the solvability condition). Parabolic equations (e.g., diffusion equation) require one initial condition and one set of boundary conditions on each boundary. Hyperbolic equations (e.g., the linear advection equation) are slightly more complicated, as they require one initial condition and one boundary condition. However, the location of the boundary condition is critical, as illustrated in Fig. 10.1. So if the advection velocity $v > 0$ and the solution travels in the positive x-direction, the proper boundary condition is specified on the left boundary. For problems involving nonlinear advection, the situation is much less obvious.

Before we proceed, let's list the three basic types of boundary conditions (Fig. 10.2):

- Dirichlet boundary conditions, when the value of the dependent variable is prescribed at the boundary, e.g. $u(0) = a$;

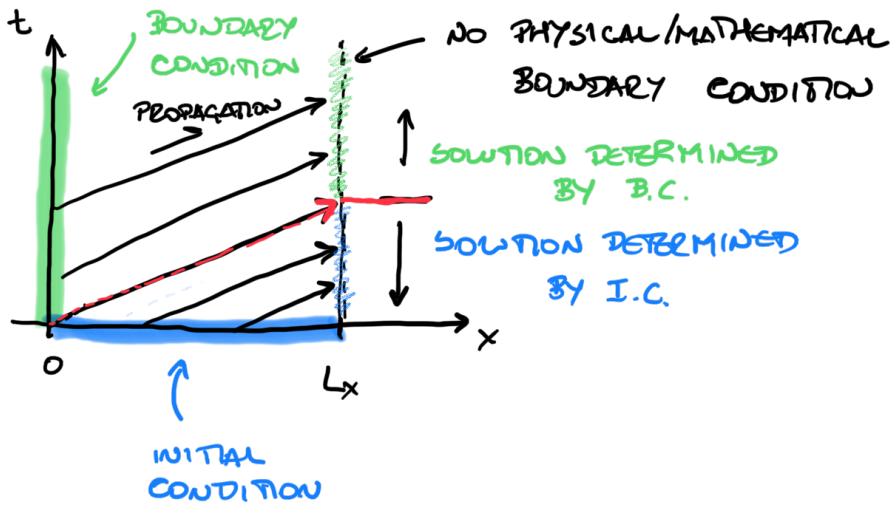

Figure 10.1: Boundary conditions for advection equation with $v > 0$.

- Neumann boundary conditions, when the derivative of the dependent variable is prescribed at the boundary, e.g. $(\partial u/\partial x)_0 = a$ or $(\partial u/\partial y)_0 = a$;
- Robin boundary conditions, a weighted combination of Dirichlet and Neumann, e.g., $\alpha u(0) + \beta(\partial u/\partial x)_0 = a$;

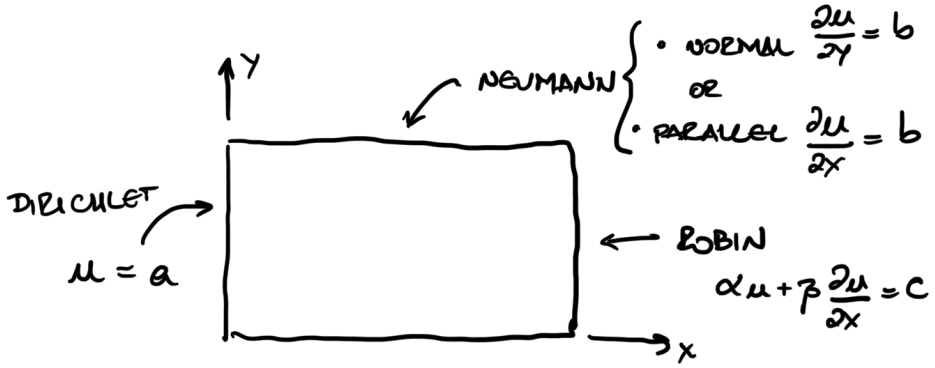

Figure 10.2: Different types of boundary conditions.

In the realm of numerical solutions on discrete space-time domains, Dirichlet boundary conditions can be enforced exactly, but Neumann and Robin conditions will always be approximate. As an example, suppose we have a Neumann boundary condition $(\partial u/\partial x)_{x=L_x} = a$, as illustrated in Fig. 10.3.

One possibility is to use a first-order one-sided approximation using only the interior

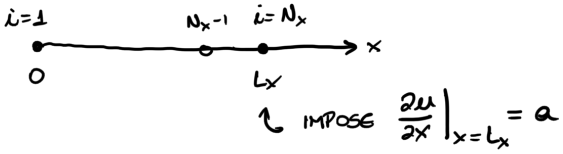

Figure 10.3: Example of Neumann boundary condition.

node. In this case we have

$$\frac{u_{N_x} - u_{N_x-1}}{\Delta x} = a. \tag{10.1}$$

In this case, we can solve the recursive equation for the interior nodes, and then manipulate the equation above to calculate the value of the solution at the boundary

$$u_{N_x} = u_{N_x-1} + a\Delta x. \tag{10.2}$$

An alternative approach is to use a ghost node and use a one-sided first-order approximation outside of the domain

$$\frac{u_{N_x+1} - u_{N_x}}{\Delta x} = a. \tag{10.3}$$

In this case, we use the recursive equation for the interior and boundary nodes, and then manipulate the equation above to set the solution value on the ghost node

$$u_{N_x+1} = u_{N_x} + a\Delta x. \tag{10.4}$$

The main advantage of using a ghost node is that we can easily use a centered second-order difference at the boundary (which is consistent with the treatment used for the interior nodes)

$$\frac{u_{N_x+1} - u_{N_x-1}}{2\Delta x} = a, \tag{10.5}$$

with an equation for the ghost node given by

$$u_{N_x+1} = u_{N_x-1} + 2a\Delta x. \tag{10.6}$$

So in general, we can deal fairly easily with all three types of boundary conditions indicated above. The difficulty arises when we need a numerical boundary condition at a point where a proper (physical/mathematical) boundary conditions does not exist. This is the case in the advection equation. In the example shown in Fig. 10.1, even though there is no true mathematical boundary condition at the right boundary, we need to impose something in the numerical solution. Our goal is to impose a boundary conditions that allows the wave to propagate out of the domain as "naturally" as possible. These are usually called *radiation*

or *outflow* boundary conditions.

For the linear advection equation, if we simply set $u = 0$ at the boundary, this will work and the solution will disappear at the boundary as if it is propagating out. However, for nonlinear advection, this simple approach will cause the wave to reflect back into the domain and contaminate the solution. In this case, there are two main approaches: (i) to use the original PDE on the boundary or (ii) to implement an absorbing layer near the boundary.

The linear advection equation is a very simple case, so let's start there. Suppose we have, as before,

$$\frac{\partial u}{\partial t} + v\frac{\partial u}{\partial x} = 0, \tag{10.7}$$

defined for $0 \leq x \leq L_x$ and $t \geq 0$, with $v > 0$, one initial condition and a proper boundary condition at $x = 0$. The issue is to select the appropriate numerical boundary condition at $x = L_x$. Note that for this simple case, even $u(x = L_x, t) = 0$ would work. However, a slightly better approach is to use the PDE with a one sided difference at the boundary to evolve the solution at the boundary:

$$\left(\frac{\partial u}{\partial t}\right) + |v|\frac{(u_{Nx} - u_{Nx-1})}{\Delta x} = 0. \tag{10.8}$$

Here we used the absolute value $|v|$, to emphasize that the wave is moving out of the domain. Even if we had $v < 0$, if we needed a numerical boundary condition on the right boundary, we could use this equation as long as we include the absolute value (so information always moves out of the domain). Similarly, on the left boundary we can use

$$\left(\frac{\partial u}{\partial t}\right) - |v|\frac{(u_2 - u_1)}{\Delta x} = 0. \tag{10.9}$$

Note that this approach can be extended to nonlinear equations if we know the phase speed of the propagating wave. For example, in the case of deepwater surface gravity waves we have $c_{ph} = \sqrt{gH}$ and we can replace $|v| = |c_{ph}|$ in the two equations above. Note that for many problems of interest we can calculate phase speeds ahead of time.

In general, there are not satisfactory solutions to deal with problems involving nonlinear advection. One possible approach is to add wave-absorbing layers (also called *sponge layers*) near the boundaries. This can be done by adding a spatially varying artificial viscosity $\nu(x)$ or a Rayleigh damping $R(x)$ term to the equations. As an example, for the x component of the momentum equation we can write

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho_0}\frac{\partial p}{\partial x} + \nu(x)\frac{\partial^2 u}{\partial x^2} - R(x)u. \tag{10.10}$$

Note that one would use either option, but usually not both at the same time. The idea

is that $\nu(x)$ or $R(x)$ should be zero within most of the domain (where the solution will be accurate) and increase near the boundaries damping the solution to avoid reflections on the boundaries (see Fig. 10.4). In this case, since the true boundaries are not very relevant, one can simply use Dirichlet boundary conditions (e.g., $u(0) = u(L_x) = 0$) on both boundaries. The sponge layers must be thick enough for $\nu(x)$ or $R(x)$ to increase slowly, otherwise waves will reflect on the sponge layer.

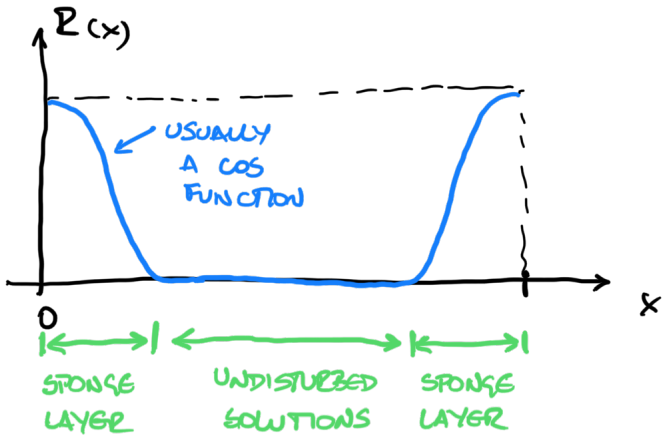

Figure 10.4: Sponge layers using artificial viscosity or Rayleigh damping to prevent wave reflections on the boundaries of the domain.

Chapter 11

Finite-volume methods

The finite-volume method takes a very different approach from the finite-difference method. Instead of discretizing the domain in nodes and approximating the derivatives, the idea is to discretize the domain in small volumes (or small areas in the 2D case) and write conservation equations for each volume. In practice, we integrate the governing equations over each volume before making any approximations. This approach is ideal to enforce physical constraints such as mass conservation.

To gain a basic understanding of the approach, we will use a generic conservation equation including advection, diffusion, and a source/sink term:

$$\frac{D\theta}{Dt} = \gamma \nabla^2 \theta + S_\theta. \tag{11.1}$$

Here S_θ is the source/sink term.

Recall that the material derivative can be written as

$$\frac{D\theta}{Dt} = \frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \frac{\partial \theta}{\partial t} + \nabla \cdot (\mathbf{u}\theta). \tag{11.2}$$

The second equality holds under the assumption that $\nabla \cdot \mathbf{u} = 0$ applied to the vector identity

$$\nabla \cdot (\mathbf{u}\theta) = \mathbf{u} \cdot \nabla \theta + \theta (\nabla \cdot \mathbf{u}). \tag{11.3}$$

We can write the original conservation equation (11.1) in *conservative form* (also called *flux form*):

$$\frac{\partial \theta}{\partial t} + \nabla \cdot (\mathbf{u}\theta) = \nabla \cdot (\gamma \nabla \theta) + S_\theta. \tag{11.4}$$

Next we discretize the domain into small volumes (as indicated in Fig. **??**) and integrate the conservative form of the equation over one of these volumes. Thus, we write

$$\int_V \frac{\partial \theta}{\partial t} dV + \int_V \nabla \cdot (\mathbf{u}\theta) dV = \int_V \nabla \cdot (\gamma \nabla \theta) dV + \int_V S_\theta dV. \tag{11.5}$$

This integral equation is valid for each volume represented in Fig. 11.7. So we went from a partial differential equations that was valid at each and every point to an integral equation that is valid for each and every volume.

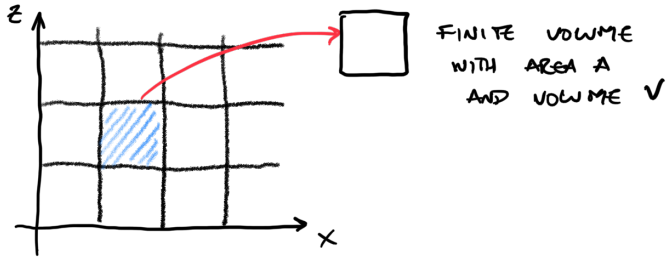

Figure 11.1: Finite-volume discretization.

Next we use Gauss' divergence theorem to the two terms that involve the divergence operators:

$$\int_V \frac{\partial \theta}{\partial t} dV + \int_A (\mathbf{u}\theta) \cdot \hat{\mathbf{n}} dA = \int_A (\gamma \nabla \theta) \cdot \hat{\mathbf{n}} dA + \int_V S_\theta dV. \tag{11.6}$$

Finally, we define the average of the quantity θ over the volume as

$$\overline{\theta} = \frac{1}{V} \int_V \theta dV, \tag{11.7}$$

and re-write the governing equation as

$$\frac{\partial \overline{\theta}}{\partial t} + \int_A (\mathbf{u}\theta) \cdot \hat{\mathbf{n}} dA = \int_A (\gamma \nabla \theta) \cdot \hat{\mathbf{n}} dA + \overline{S_\theta}. \tag{11.8}$$

The second term on the left-hand side represents all the advective fluxes across the boundary of the volume, and the first term on the right-hand side the diffusive fluxes across the same boundary. This equation is still exact, even though it contains less information than the original PDE (from the original PDE we can get the solution at every single point, and from this equation we can only get the solution averaged over small volumes).

A critical difference between the finite-difference and the finite-volume methods is that, instead of approximating the spatial derivatives, here we approximate the integral of the fluxes across the surfaces. In addition, we are not really solving an approximate equation for the solution ad a nodal value, but rather an equation for the average within a volume.

To get a better sense of how things work in practice, we will start with a simple linear advection problem in one dimension (in conservative form):

$$\frac{\partial \theta}{\partial t} + \frac{\partial (v\theta)}{\partial x} = 0. \tag{11.9}$$

In one dimension, the "finite volumes" are actually finite line segments as illustrated in Fig. 11.2. We will represent the value at the center of the volume, even though we know we are actually calculating the average over the entire volume.

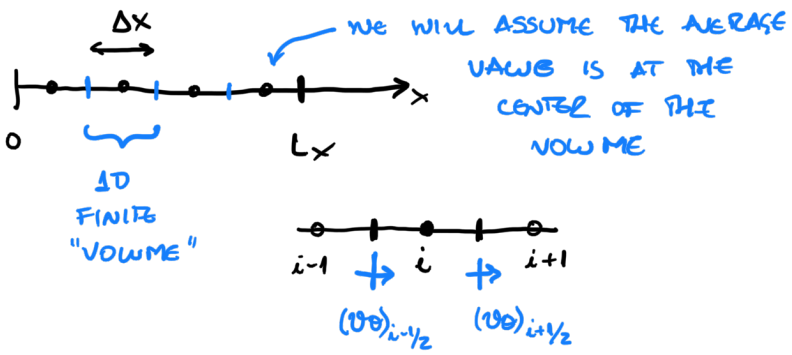

Figure 11.2: One dimensional finite-volume discretization.

For the discretization in Fig. 11.2, the average is defined as

$$\overline{\theta}_i = \frac{1}{\Delta x}\int_{x_i-\Delta x/2}^{x_i+\Delta x/2} \theta dx. \tag{11.10}$$

Also, in this case we have that

$$\int_A (\mathbf{u}\theta)\cdot\hat{\mathbf{n}} dA = \frac{1}{\Delta x}\left[(v\theta)_{i+1/2} - (v\theta)_{i-1/2}\right]. \tag{11.11}$$

Therefore, the integrated equation becomes

$$\frac{d\overline{\theta}_i}{dt} + \frac{1}{\Delta x}\left[(v\theta)_{i+1/2} - (v\theta)_{i-1/2}\right] = 0. \tag{11.12}$$

Now we need to choose a scheme for the time discretization. If we use leapfrog we have

$$\overline{\theta}_i^{n+1} = \overline{\theta}_i^{n-1} - \frac{2\Delta t}{\Delta x}\left[(v\theta)_{i+1/2}^n - (v\theta)_{i-1/2}^n\right]. \tag{11.13}$$

This is almost the recursive equation. The critical step is the decision regarding the approximation of the fluxes: we need to interpolate θ (and sometimes the velocity v) to the surfaces of the elements, which is represented by the locations $(i+1/2)$ and $(i-1/2)$ in this case. Note that for the linear advection equation, v is constant and is not an issue. If $v = u$ is the streamwise velocity field from solving the Navier-Stokes equations, then the velocity also needs to be interpolated.

The simplest approach is to use linear interpolations. In this case we have

$$(v\theta)^n_{i-1/2} = \frac{(v\bar{\theta})^n_i + (v\bar{\theta})^n_{i-1}}{2}$$
$$(v\theta)^n_{i+1/2} = \frac{(v\bar{\theta})^n_{i+1} + (v\bar{\theta})^n_i}{2}. \tag{11.14}$$

In this case we have

$$\bar{\theta}^{n+1}_i = \bar{\theta}^{n-1}_i - \frac{\Delta t}{\Delta x}\left[(v\bar{\theta})^n_{i+1} - (v\bar{\theta})^n_{i-1}\right]. \tag{11.15}$$

In practice, this is exactly the same equation we would get if we had used centered finite differences applied to the conservative form of the equations (the interpretation is slightly different, though). However, there are many other possible interpolation schemes, some of which are designed to guarantee physical constraints on the solution.

11.1 Higher-order schemes

The linear interpolation used for the fluxes in Eq. (11.14) introduces an error of order Δx. If we want to use quadratic interpolations instead, we need to include one more point. As we learned from the finite difference scheme, it is better to use upwind schemes due to the nature of advection. The quadratic upwind scheme is usually called QUICK (see Figure 11.3).

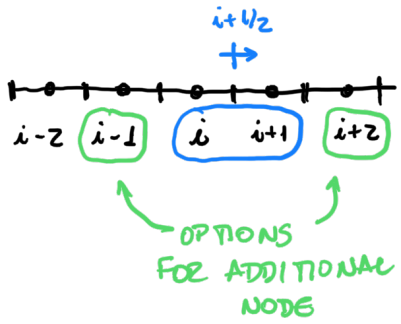

Figure 11.3: QUICK quadratic upwind interpolation.

In the case of the QUICK scheme applied to the grid shown in Figure 11.3, the quadratic interpolation results in

$$(v\theta)_{i+1/2} = (6/8)(v\bar{\theta})_i + (3/8)(v\bar{\theta})_{i+1} - (1/8)(v\bar{\theta})_u, \tag{11.16}$$

where the index $_u$ represents the upwind point such that

$$(v\bar{\theta})_u = \begin{cases} (v\bar{\theta})_{i-1} \text{ if } v > 0 \\ (v\bar{\theta})_{i+2} \text{ if } v < 0 \end{cases} . \tag{11.17}$$

A class of useful schemes developed for finite-volume methods is the monotonic schemes. For our purposes, a monotonic advection scheme is one for which given an initial condition that is monotonic, the scheme produces solutions that remain monotonic after advection. This essentially means that the numerical scheme does not create new extrema nor amplify existing extrema. In many problems in GFD, monotonicity is a really important constraint. For example, many scalar fields of interest cannot take on negative values and a monotonic scheme guarantees that, as long as the initial condition does not have negative values, the scheme will not produce negative values. A typical problem with non-monotonic schemes is that they usually produce spurious oscillations in the wake of waves with sharp gradients as illustrated below.

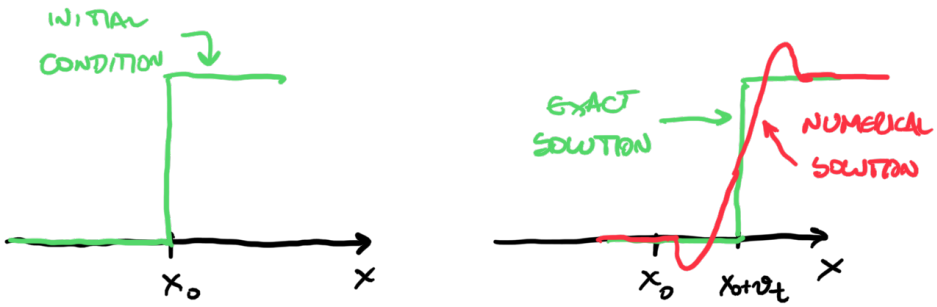

Figure 11.4: Example of a non-monotonic scheme producing new maxima and minima (including negative values) in an advection problem.

In problems where monotonicity is important, we want to use a monotonic scheme. However, *Godunov's theorem* states that a consistent linear numerical scheme that is monotonic can at most be first-order accurate in space. So in principle, we have to use the first-order upwind scheme. However, this scheme is particularly problematic near sharp gradients, as it contaminates the solution with large numerical diffusion that artificially smooths the sharp gradients. The choice between a centered difference that is more accurate and maintains the gradients but is non-monotonic and the upwind scheme that is monotonic but smoothes out the gradient is not very satisfying.

Before we can find a better solution, let's understand better the source of the problem. In the finite-volume method we can actually track the source of the problem to the interpolation of the fluxes. As illustrated in Figure 11.6, near the sharp gradient, the interpolation leads

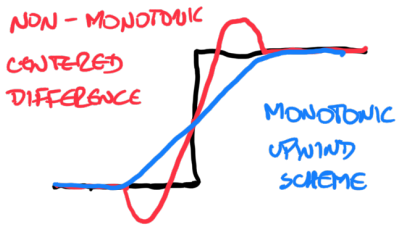

Figure 11.5: Monotonic and non-monotonic schemes near sharp gradients.

to a positive flux on the downwind edge of the finite volume, even though $\theta_i = 0$ within the element.

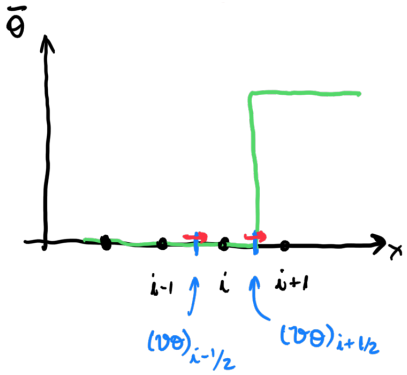

Figure 11.6: Fluxes near a sharp gradient.

Note that in the case depicted above, the use of linear interpolation leads to $(v\theta)_{i+1/2} > 0$ and $(v\theta)_{i-1/2} = 0$, so that

$$\frac{\partial \overline{\theta}_i}{\partial t} = -\frac{(v\theta)_{i+1/2} - (v\theta)_{i-1/2}}{\Delta x} < 0, \tag{11.18}$$

and

$$\overline{\theta}_i^n = 0 \Rightarrow \overline{\theta}_i^{n+1} < 0, \tag{11.19}$$

producing the negative bump behind the propagating sharp gradient. The same effect explains the positive bump ahead of the gradient.

In order to better understand the issue, let's look at the first-order upwind scheme, which is monotonic. If we go back to (11.9), for the case $v > 0$ we have

$$\frac{d\overline{\theta}_i}{dt} + \frac{(v\overline{\theta})_i - (v\overline{\theta})_{i-1}}{\Delta x} = 0 \tag{11.20}$$

and for $v < 0$

$$\frac{d\overline{\theta}_i}{dt} + \frac{(v\overline{\theta})_{i+1} - (v\overline{\theta})_i}{\Delta x} = 0. \tag{11.21}$$

Comparing these expressions with the finite-volume discretization (11.12) we can see that in order to recover the first-order upwind scheme we need to replace the linear interpolations (11.14) by

$$(v\theta)_{i+1/2} = \begin{cases} (v\overline{\theta})_{i+1} & \text{if } v < 0 \\ (v\overline{\theta})_i & \text{if } v > 0 \end{cases} \tag{11.22}$$

and

$$(v\theta)_{i-1/2} = \begin{cases} (v\overline{\theta})_i & \text{if } v < 0 \\ (v\overline{\theta})_{i-1} & \text{if } v > 0 \end{cases}. \tag{11.23}$$

Note that in the example of the sharp gradient in the figure above, this scheme would result in $(v\theta)_{i+1/2} = (v\theta)_{i-1/2} = 0$, avoiding the problem of the second-order scheme.

Clearly, the solution has to be associated with better approaches to estimate $(v\theta)_{i+1/2}$ near the sharp gradient. There are two main classes of methods discussed below. The first class of methods are the flux-corrected transport (FCT) schemes. The basic idea is to approximate the fluxes using two separate schemes:

- $(v\theta)^L_{i+1/2}$ is the flux obtained from a low-order monotonic scheme such as (11.22);
- $(v\theta)^H_{i+1/2}$ is the flux obtained from a high-order non-monotonic scheme such as (11.14);

Then both approximations are combined giving as much weight as possible to $(v\theta)^H_{i+1/2}$ without violating monotinicity. This is usually done by first calculating a low-order solution θ^L, which is then corrected using $(v\theta)^H_{i+1/2}$. We will not discuss this approach in detail (refer to Durran 5.4 for more details).

The flux-limiter methods are more commonly used, in part because they do not require the calculation of an intermediate solution θ^L. They approximate the final flux to be used as an average of the two estimates

$$(v\theta)_{i+1/2} = (v\theta)^L_{i+1/2} + C_{1+1/2}((v\theta)^H_{i+1/2} - (v\theta)^L_{i+1/2}), \tag{11.24}$$

where $C_{1+1/2}$ is a multiplicative limiter calculated as a nonlinear function of the local solution by using the ratio between solution slopes upwind and across $x_{i+1/2}$. The slope across $x_{i+1/2}$ is given by $(\overline{\theta}_{i+1} - \overline{\theta}_i)/\Delta x$. As an example, in the case $v_{i+1/2} > 0$, the upwind slope is $(\overline{\theta}_i - \overline{\theta}_{i-1})/\Delta x$. The ratio is defined as

$$r_{i+1/2} = \frac{\overline{\theta}_i - \overline{\theta}_{i-1}}{\overline{\theta}_{i+1} - \overline{\theta}_i} \quad \text{for } v_{i+1/2} > 0 \tag{11.25}$$

and

$$r_{i+1/2} = \frac{\overline{\theta}_{i+2} - \overline{\theta}_{i+1}}{\overline{\theta}_{i+1} - \overline{\theta}_i} \quad \text{for } v_{i+1/2} < 0. \tag{11.26}$$

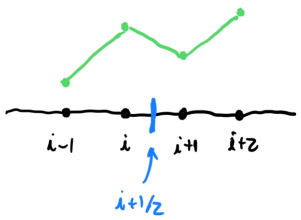

Figure 11.7: Stencil for flux-limiters.

Then, the multiplicative limiter is determined by the limiter function $C(r)$

$$C_{i+1/2} = C(r_{i+1/2}). \tag{11.27}$$

Individual schemes are obtained by different functions $C(r)$, which have to satisfy formal restrictions to ensure consistency and monotinicity. Some commonly used flux-limiter schemes are:

- Minmod limiter $C(r) = \max\{0, \min(1, r)\}$
- Superbee limiter $C(r) = \max\{0, \min(1, 2r), \min(2, r)\}$
- van Leer limiter $C(r) = \dfrac{r + |r|}{1 + |r|}$

For more details on flux-limiters, see Durran 5.5.

If you are interested in modern monotonic schemes, look for the essentially nonoscilatory (ENO) and the weighted essentially nonoscillatory (WENO) methods. These methods use polynomial interpolations and adaptive stencils to construct approximations for the flux and the flux divergence. Using WENO, it is very easy to construct higher-order schemes for three-dimensional problems in finite differences. There is a brief discussion in Durran 5.6.