Geophysical Fluid Dynamics, theory and modeling

S. Thual, Fudan University 2019

Lecture 4: Computational Fluid Dynamics

In this chapter, we introduce a few notions of Computational fluid dynamics (CFD) that will be useful to solve the equations that govern the ocean and atmosphere in the rest of the course. CFD focuses on building and solving numerical representations that approximate fluid mechanics problems in the real world. It has become a very useful and common tool with the rise of computer capabilities. CFD combines notions of numerical analysis, algorithms and computer sciences and always requires a strong physical understanding of the fluid mechanics problem itself. Atmospheric or oceanic scientists that focus more on the later aspect can benefit from at least a basic understanding of CFD (for example when choosing a numerical method for their model).

Contents

1 Finite Difference Methods

In order to solve a system of equations that describes the ocean or atmosphere on the computer, one may represent it in discrete version (typically on a model grid). Several method exists for this: finite elements, finite differences, finite volumes, spectral methods, etc. Here we will focus only on finite difference methods because there are simplest to understand and implement.

1.1 Continuous versus discrete system

Consider a function $f(t)$ that depends on variable t, with evolution given by:

$$\frac{df}{dt} \mid_t = Q(f) \mid_t \tag{1}$$

The above system in equation 39 is continuous. In order to solve the system on a computer, we may consider instead a discrete representation of it. Figure 1 shows an example of a

continuous function $f(t)$ represented as a discrete function $f = [f_1, f_2, ..., f_N]$ on a discrete axis $t = [t_1, t_2, ..., t_N]$, with $f_k = f(t_k)$, $t_k = k\Delta t$. This a point-wise representation, i.e. we only treat values of $f(t)$ at specific points.

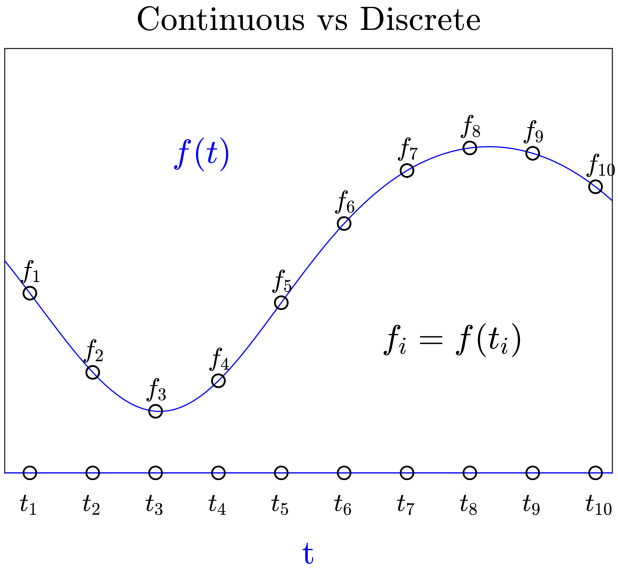

Figure 1: Continuous function $f(t)$ and its discrete representation (point-wise).

1.2 Taylor series

For a given function $f(t)$ we can express the conditions at $f(t+\Delta t)$ using an expansion into Taylor series. This reads:

$$\begin{aligned} f(t+\Delta t) &= f(t) + \Delta t \frac{df}{dt} \mid_t + O(\Delta t^2) \\ f(t+\Delta t) &= f(t) + \Delta t f \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + O(\Delta t^3) \\ f(t+\Delta t) &= f(t) + \Delta t \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + \frac{\Delta t^3}{6} \frac{d^3 f}{dt^3} \mid_t + O(\Delta t^4) \end{aligned} \tag{2}$$

where $\frac{df}{dt} \mid_t$ indicates the first derivative, $\frac{d^2 f}{dt^2} \mid_t$ the second derivative, and so on. The Taylor series are expanded up to a first few terms, and $O(\Delta t^2)$ for example that additional terms proportional to Δt^2 are remaining (it is always possible to expand the series further). Note that we can also

express the conditions at $f(t-\Delta t)$:

$$\begin{aligned} f(t-\Delta t) &= f(t) - \Delta t \frac{df}{dt} \mid_t + O(\Delta t^2) \\ f(t-\Delta t) &= f(t) - \Delta t \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + O(\Delta t^3) \\ f(t-\Delta t) &= f(t) - \Delta t \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t - \frac{\Delta t^3}{6} \frac{d^3 f}{dt^3} \mid_t + O(\Delta t^4) \end{aligned} \tag{3}$$

1.3 Finite Differences Approximation

Using the Taylor series, we can find several approximations of the derivatives of $f(t)$. These are called finite difference methods because, as shown below, the derivatives are expressed as a combination of nearby values of f.

For example, we note from equation 2 that:

$$\frac{df}{dt} \mid_t = \frac{f(t+\Delta t) - f(t)}{\Delta t} + O(\Delta t) \tag{4}$$

which can be approximated as:

$$\frac{df}{dt} \mid_t \approx \frac{f(t+\Delta t) - f(t)}{\Delta t}. \tag{5}$$

in which case we omit an additional term of magnitude $O(\Delta t)$. Similarly, from equation 3 we can find another approximation from:

$$\frac{df}{dt} \mid_t = \frac{f(t) - f(t-\Delta t)}{\Delta t} + O(\Delta t). \tag{6}$$

We can also approximate higher order derivatives in the same way, for example from:

$$\frac{d^2 f}{dt^2} \mid_t = \frac{f(t+\Delta t) + 2f(t) + f(t-\Delta t)}{\Delta t^2} + O(\Delta t^3) \ . \tag{7}$$

All of these approximations become true when Δt tends to zero, because:

$$\lim_{\Delta t \to 0} O(\Delta t^n) = 0 \tag{8}$$

for any given n.

1.4 Finite Differences Methods in Space

Discrete Derivative:

The Finite Difference approximation presented above can be used to approximate discrete derivatives of $f(x)$, where x represents space. Consider the discrete version $f_i = f(x_i)$ with $x_i = i\Delta x$. For example:

$$\frac{df}{dx} \mid_x \approx \frac{f(x+\Delta x) - f(x)}{\Delta x} \tag{9}$$

expressed at $x = x_i$, with $f(x_i) = f_i$ and $f(x_i + \Delta x) = f_{i+1}$ becomes:

$$\frac{df}{dx} \mid_i \approx \frac{f_{i+1} - f_i}{\Delta x} \tag{10}$$

Common Methods:

There are many ways to approximate derivatives with the finite differences, called methods or schemes. We list below in Table 1 a few well-known methods for dealing with derivatives in space. All these schemes are obtained from the Taylor series. To obtain the forward scheme, we use equation 2 at $O(\Delta x^2)$. To obtain the backward scheme, we use instead equation 3 at $O(\Delta x^2)$. To obtain the centered schemes, we combine equation 2 and equation 3 at $O(\Delta x^3)$. We can also obtain higher order schemes by using more terms of the Taylor series (not shown).

$\frac{df}{dx}$ becomes	method
$\frac{df}{dx} \mid_i = \frac{f_{i+1} - f_i}{\Delta x}$	forward scheme (1st order)
$\frac{df}{dx} \mid_i = \frac{f_i - f_{i-1}}{\Delta x}$	backward scheme (1st order)
$\frac{df}{dx} \mid_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x}$	centered scheme 2nd order
$\frac{d^2f}{dx^2} \mid_i = \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$	centered scheme 2nd order

Table 1: Common Finite Difference methods in space.

1.5 Finite Differences Methods in Time

Starting Continuous System:

Finite differences methods that approximate derivatives in time are usually more advanced that finite difference methods in space. Consider again a continuous function $f(t)$, where t is time, that evolves according to:

$$\frac{df}{dt} \mid_t = Q(f) \mid_t \tag{11}$$

where Q summarizes all terms that can be found in the dynamical system. Here f can be either a scalar or a vector representing all locations in space, for which the conclusions in this section are the same.

List of Finite Difference Methods in Time:

Finite differences methods in time approximate the above relationship in several ways in order to predict conditions at later times. Table 2 lists some common methods, where we note $f_k = f(t_k)$, $Q_k = Q(f) \mid_{t_k}$ with $t_k = k\Delta t$. In practice, there are many more methods to choose from, and they can even be combined (for example it is common to treat linear terms of Q with an implicit method and nonlinear terms of Q with an explicit methods).

prediction	$\frac{df}{dt} = Q(f)$ becomes	method
$f_{k+1} = f_k + Q_k \Delta t$	$\frac{f_{k+1}-f_k}{\Delta t} = Q_k$	explicit Euler method (1st order)
$f_{k+1} - \Delta t Q_{k+1} = f_k$	$\frac{f_{k+1}-f_k}{\Delta t} = Q_k$	implicit Euler method (1st order)
$f_{k+1} = 2Q_k \Delta t + f_{k-1}$	$\frac{f_{k+1}-f_{k-1}}{2\Delta t} = Q_k$	Leap-frog method (explicit, 2nd order)
$f_{k+1} - \Delta t \frac{Q_{k+1}}{2} = f_k + \frac{Q_k}{2} \Delta t$	$\frac{f_{k+1}-f_k}{\Delta t} = \frac{Q_{k+1}+Q_k}{2}$	Crank-Nicolson (implicit, 2nd order)
$f_{k+1} = f_k + (\frac{3}{2}Q_k - \frac{1}{2}Q_{k-1})\Delta t$	$\frac{f_{k+1}-f_k}{\Delta t} = \frac{3}{2}Q_k - \frac{1}{2}Q_{k-1}$	Adams-Bashforth (explicit, 2nd order)
$f_{k+1} = f_k + \Delta t Q(f_k + \Delta t Q(f_k))$	see text	Runge-Kutta 2nd order (explicit, iterative)
see text	see text	Runge-Kutta 4th order (explicit, iterative)

Table 2: Common Finite Difference methods in time.

Implicit Methods:

Implicit methods are methods in which the predictor f_{k+1} is obtained by solving an implicit

system. The implicit Euler method in Table 2 is a simple example:

$$f_{k+1} - \Delta t Q_{k+1} = f_k \tag{12}$$

which is an implicit equation: we must solve for f_{k+1} and Q_{k+1} together given f_k. Another example in Table 2 is the Crank-Nicolson method. Implicit methods are more accurate and stable than explicit methods (see hereafter), but due to the solving of implicit equations they are significantly more expensive to perform on a computer. For example the nonlinear Stuart-Landau equation $df/dt = af - bf^3$ combined with the implicit Euler method leads to $(1 - \Delta t a) f_{k+1} + \Delta t b f_{k+1}^3 = f_k$ for which we solve for f_{k+1} as root of a polynomial equation. As another example, the linear equation $d\underline{f}/dt = \underline{\underline{M}}\underline{f}$ where $\underline{f}$ is a vector and $\underline{\underline{M}}$ a matrix combined with the implicit Euler method leads to $(\underline{\underline{I}} - \Delta t \underline{\underline{M}})\underline{f}_{k+1} = \underline{f}_k$ that is solved by inverting the matrix $(\underline{\underline{I}} - \Delta t \underline{\underline{M}})$ (where $\underline{\underline{I}}$ is the identity matrix).

Iterative methods:

Iterative methods (also called predictor-corrector methods) compute the solution f_{k+1} in several steps. The most famous is the Runge-Kutta method, for which several versions exist. The Runge-Kutta method at second order adds an intermediary step by computing a function f^*:

$$\frac{f^* - f_k}{\Delta t} = Q(f_k),\ \frac{f_{k+1} - f_k}{\Delta t} = Q(f^*) \text{ which gives} \tag{13}$$

$$f_{k+1} = f_k + \Delta t Q(f_k + \Delta t Q(f_k)) \text{: Runge Kutta second order} \tag{14}$$

The Runge-Kutta method at third order adds two intermediary steps with f^*, f^{**} (not shown). Finally, the Runge-Kutta method at fourth order is the most popular method. It reads:

$$\frac{f^* - f_k}{\Delta t} = \frac{Q(f_k)}{2},\ \frac{f^{**} - f_k}{\Delta t} = \frac{Q(f^*)}{2},\ \frac{f^{***} - f_k}{\Delta t} = Q(f^{**}),\ \text{and} \tag{15}$$

$$\frac{f_{k+1} - f_k}{\Delta t} = \frac{1}{6}[Q(f_k) + 2Q(f^*) + 2Q(f^{**}) + Q(f^{***})] \tag{16}$$

where the coefficients are chosen to minimize the error (see hereafter).

1.6 Combining Finite Difference Methods in Space and Time

Assume now that we want to represent a continuous function $f(x,t)$ that depends on both space x and time t. In that case we simply combine finite difference methods. Consider for example the system

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial x} = P(f) \tag{17}$$

and its discrete version $f_{ik} = f(x_i, t_k)$, $P_{ik} = P(f_{ik})$ with $x_i = i\Delta x$ and $t_i = k\Delta t$. We discretize space first, for example using a centered scheme of 2nd order, which gives:

$$\frac{\partial f}{\partial t}\ |_{i,k} + c\frac{f_{i+1,k} - f_{i-1,k}}{2\Delta x} = P_{ik} \tag{18}$$

and then put the system in the form $\partial f/\partial t = Q(f)$ to discretize time:

$$\frac{\partial f}{\partial t}\ |_{i,k} = Q_{ik},\ Q_{ik} = P_{ik} - c\frac{f_{i+1,k} - f_{i-1,k}}{2\Delta x}. \tag{19}$$

An explicit Euler method in time for example gives:

$$f_{i,k+1} = f_{i,k} + \Delta t P_{ik} - \Delta t c\frac{f_{i+1} - f_{i-1}}{2\Delta x}. \tag{20}$$

2 Choosing Finite Difference Methods for a Model

In practice, many factors come in play when choosing which finite difference methods to use in a model (in both space and time). Generally speaking, these factors can be summarized as:

a) Cost to compute with arithmetic operations

b) Error r in representation (accuracy)

c) If numerical instability can happen

d) Other practical issues (aliasing, numerical diffusion,etc)

Usually, there is a trade-off to make between the factors. Methods that are more costly are

usually more accurate, and inversely. Also those issues depend on the physical problem that is represented.

2.1 Cost of Finite Difference Methods

Computer simulations use arithmetic operations to compute the solutions of a model, These arithmetic operations are repeated many times, at each grid mesh and timestep. For example a very simple model with a grid x, y of size 100x100 and integrated over time t for around 100 timesteps repeats arithmetic operations at each grid point 1 million times. Because computing resources are in general limited, it is always useful to choose finite difference methods that minimize each arithmetic operation. For example, we can see intuitively that and explicit Euler method $f_{k+1} = Q_k \Delta t + f_k$ (one addition) is cheaper than an Adams-Bashforth $f_{k+1} = (\frac{3}{2} Q_k - \frac{1}{2} Q_{k-1}) \Delta t + f_k$ (two additions), and both are much cheaper than implicit Euler implicit $f_{k+1} - \Delta t Q_{k+1} = f_k$ (solve an implicit equation).

2.2 Accuracy of Finite Difference Methods

True continuous system versus discrete system:

Because finite difference methods use approximations the discrete system is not equivalent to the continuous system it represents:

$$\frac{df}{dt} \mid_t = Q(f)|_t \quad \neq \quad \frac{df}{dt} \mid_k = Q_k \tag{21}$$

The accuracy of a finite difference method is given by the error r made in representing the continuous system:

$$r = \left(\frac{df}{dt} \mid_k - \frac{df}{dt} \mid_t \right). \tag{22}$$

We are usually interested in the order n of magnitude of the error $r = O(\Delta t^n)$ ($n = 1$ for 1st order, $n = 2$ for 2nd order...), which gives the accuracy of the method. The smaller the order n the better. In some cases we are also interested in the convergence of the method, i.e. how $r \to 0$ when $\Delta t \to 0$. We can also see that r introduces artificial dynamics in our system, because the

approximated continuous system that we are really solving within $t_k \leq t \leq t_{k+1}$ is:

$$\frac{df}{dt} \mid_t = Q_k - r \tag{23}$$

where $Q_k$is treated as a constant, and r can depend on f or/and t.

How to compute the error order:

To compute the error r, put the equations in continuous form and compare them with the Taylor series . For example the explicit Euler scheme gives:

$$f_{k+1} = f_k + Q_k \Delta t, \text{ therefore} \tag{24}$$

$$f(t + \Delta t) = f(t) + \Delta t \frac{df}{dt} \mid_k = f(t) + \Delta t \left[\frac{df}{dt} \mid_t + r\Delta t \right] \tag{25}$$

and from comparison with the Taylor series in equation 2 we conclude that $r = O(\Delta t)$. Therefore the explicit Euler scheme is of order 1.

As another example, if we use the Crank-Nicolson scheme gives:

$$f_{k+1} = \frac{\Delta t}{2} (Q_{k+1} + Q_k) + f_k, \text{ therefore} \tag{26}$$

$$f(t + \Delta t) = f(t) + \frac{\Delta t}{2} \left(\frac{df}{dt} \mid_{t+\Delta t} + \frac{df}{dt} \mid_t \right) + r\Delta t \tag{27}$$

and we must combine the Taylor series (expressed from t and from $t + \Delta t$) to find a comparison:

$$f(t + \Delta t) = f(t) + \Delta t \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + O(\Delta t^3) \text{ and} \tag{28}$$

$$f(t) = f(t + \Delta t) - \Delta t \frac{df}{dt} \mid_{t+\Delta t} + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_{t+\Delta t} + O(\Delta t^3) \tag{29}$$

and (equation 28 - equation 29)/2 gives:

$$f(t + \Delta t) = f(t) + \frac{\Delta t}{2} \left(\frac{df}{dt} \mid_{t+\Delta t} + \frac{df}{dt} \mid_t \right) + O(\Delta t^3) \tag{30}$$

from which we deduce that $r = O(\Delta t^2)$. Therefore the Crank-Nicolson scheme is of order 2. Note that we can even expand r using the next terms of the Taylor series.

2.3 Numerical Instability

Numerical Instability occurs when errors in representation r are magnified over time. In that case the numerical simulation is very unphysical and can even stall because infinite values are eventually generated that cannot be treated by arithmetic operations. Beware to distinguish numerical instability (infinite values due to error r) from dynamical instability (infinite values due to the system alone, even for $r = 0$). In order to avoid this situation, some conditions must be satisfied that depend on the finite difference methods used and physical problem.

CFL condition for propagations:

For a system with propagations, the Courant-Friedrichs-Lewy (CFL) condition must be satisfied to avoid numerical instability. Consider a system of the form:

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial x} = Q(f) \tag{31}$$

in which there are propagations at characteristic speed c (for example currents or waves). The CFL condition reads:

$$C = c\frac{\Delta t}{\Delta x} \leq C_{max} \tag{32}$$

where C is called the Courant number, and C_{max} is a constant. When using explicit methods $C_{max} = 1$, however when using implicit methods that are more accurate the condition C_{max} takes sometimes higher values. We show in Figure 2 an intuitive example of what the CFL condition implies for explicit methods ($C_{max} = 1$). Consider a signal initially at x_i, t_k and propagating at speed c. For $c\Delta t \leq \Delta x$ ($C \leq 1$) the CFL condition is satisfied and the signal at t_{k+1} remains within the grid cell. For $c\Delta t > \Delta x$ ($C > 1$) the CFL condition is not satisfied and the signal overshoots the grid cell. A more formal way to demonstrate the CFL condition is to analyze the

error r in representation (not shown).

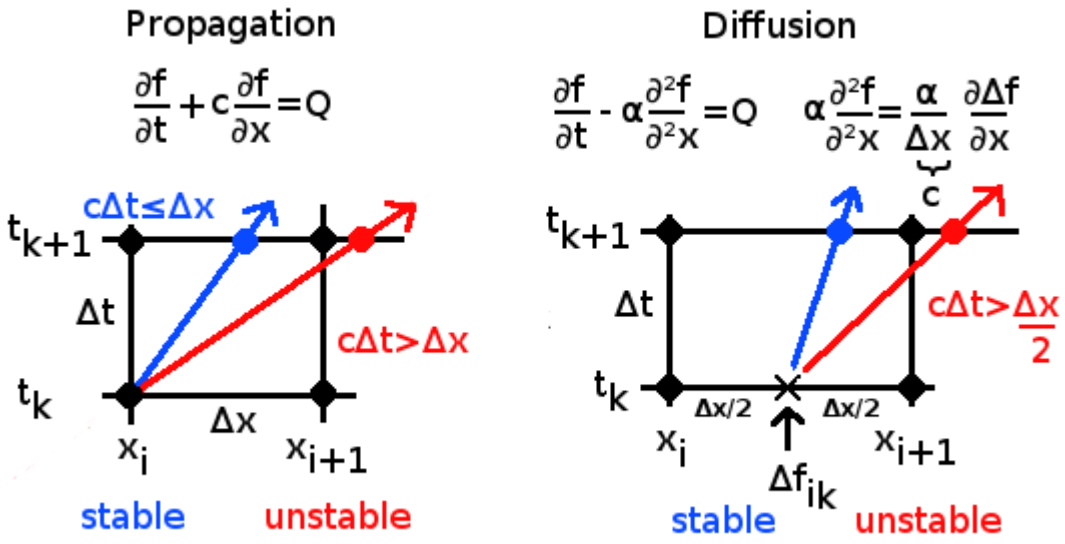

Figure 2: Illustration of CFL condition for propagation and Von Neumann condition for diffusion within a grid cell (explicit methods).

Von Neuman stability condition for diffusion:

Another condition to avoid numerical instability was given by Von Neumann, which must be satisfied when there are diffusions in the system. Consider now a system of the form:

$$\frac{\partial f}{\partial t} - \alpha \frac{\partial^2 f}{\partial x^2} = Q(f) \tag{33}$$

in which there is diffusion with viscosity coefficient $\alpha > 0$. Then numerical stability is ensured if:

$$D = \alpha \frac{\Delta t}{\Delta x^2} \leq D_{max}. \tag{34}$$

where D_{max} depends on the problem and method used. For a centered scheme in space and explicit method in time we get $D_{max} = 1/2$. For some implicit methods D_{max} can take higher values (e.g. $D_{max} = 1$ for Crank-Nicolson combined with a centered scheme in space). Figure 2 also shows an illustration of what the Von Neumann condition for diffusion means for explicit methods ($D_{max} = 1/2$). Intuitively, we can interpret this as a CFL condition: first discretize

$\partial f/\partial x = \Delta f/\Delta x$ with a centered scheme, where $\Delta f_{i,k} = f_{i+1,k} - f_{i,k}$ is expressed at a fictive grid point at the center of the grid cell. Then, the diffusion term can be treated as a propagation term with speed $c = \alpha/\Delta x$: to ensure stability $c\Delta t \leq \Delta x/2$ or the signal overshoots the grid cell. Again a more formal way to demonstrate the Von Neumann diffusion condition is to analyze the error r in representation (not shown).

How to avoid numerical instability:

In practice, ensuring the numerical stability in a model can be done in several ways:

1) lower Δt (the simulation is longer to compute)

2) increase Δx (the grid mesh is coarser)

3) Use a more accurate method, for example an implicit method (more expensive)

2.4 Other Practical Issues

We give here a few examples of known other practical issues that can arise when choosing finite difference methods.

Numerical diffusion and dispersion:

The error r in representation for a discrete system may sometimes introduce artificial processes that compare to physical ones. For example, the linear transport equation:

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial x} = 0 \tag{35}$$

(with $c \geq 0$) solved with explicit Euler in time and forward scheme in space has an error in representation r that develops as:

$$r = -c\frac{\Delta x}{2}\frac{\partial^2 f}{\partial x^2} + O(\Delta x^2) \tag{36}$$

which consists of a diffusion term at order $O(\Delta x)$ (it will be more prominent for larger Δx) and remaining terms at next orders. Here, the diffusion is obviously artificial, but it will appear as

real diffusion in the numerical solutions (where it will damp/dissipate the signals over time). This is called numerical diffusion. Another example is numerical dispersion which artificially disperse waves (not shown). It is important to identify these effects and distinguish them from real physical processes of the system.

Practical issues with the forward and backward schemes:

The forward and backward schemes in space :

$$\frac{df}{dx}\,|_i = \frac{f_{i+1} - f_i}{\Delta x} \text{ (forward)}, \quad \frac{df}{dx}\,|_i = \frac{f_i - f_{i-1}}{\Delta x} \text{ (backward)} \tag{37}$$

are rarely used in practice (a centered scheme or higher order scheme is usually preferred). The simple reason for this is that they completely ignore information incoming from part of the space domain. For example the forward scheme ignores all changes occuring at $x < x_i$, becauses f_{i-1} is ignored (see exercices). The only case these schemes are useful is for propagating signals in one direction only.

Aliasing:

Aliasing is a phenomena that occurs when phenomena with very small scale cannot be represented by the finite difference methods. A simple example is given in Figure 3 for a sinusoidal signal. The signal is correctly represented for small sampling Δt, but for large Δt the representation is very unphysical and leads to important errors. Roughly speaking, signals with frequency ω are correctly represented by the finite difference methods if:

$$|\omega| \leq \frac{\pi}{\Delta t} \tag{38}$$

where $\pi/\Delta t$ is called the Nyquist frequency (at least two samples must be taken over one period). If this condition is not met, then aliasing occurs which can lead in errors of representation r, or even numerical instability (if errors r are magnified).

Figure 3: Illustration of aliasing for a simple sinusoidal signal. Adapted from Cushman-Roisin.

3 Building and Solving a Numerical Model

We give here a complete tutorial on how to build an solve a numerical model entirely. For this, we will use the simple example of a shallow-water model. We discretize the model on a simple grid and use simple finite difference methods to estimate the derivatives.

3.1 Main components of a model

The major components of a numerical model can be summarized as follows:

a) Starting Continuous system of equations

b) Model Grid

c) Discrete system obtained from finite difference methods

d) Boundary conditions at the limits of the domain

Then, from those conditions we can determine through time integration the numerical solutions for the model. Each of these components is detailed in the next sections.

3.2 Starting continuous system of equations:

We will take here the example of a shallow-water model, that we note:

$$\begin{aligned} &\frac{\partial}{\partial t}u - fv = -\frac{\partial}{\partial x}p \\ &\frac{\partial}{\partial t}v + fu = -\frac{\partial}{\partial y}p \\ &\frac{\partial}{\partial t}p + G(\frac{\partial}{\partial x}u + \frac{\partial}{\partial y}v) = 0 \end{aligned} \tag{39}$$

These equations are written in compact form with only two parameters f and G. The reason for this is we want to write a code as simple as possible. Nevertheless, the above equations that can represent all the shallow-water models seen in previous lecture. To obtain them, do the following variable changes:

- Shallow-water model for a vertical mode n: $u_n = u$, $v_n = v$, $p_n = p/\rho_o$, $c_n = \sqrt{G}$.
- Shallow-water model for one vertical layer: $u'_1 = u$, $v'_1 = v$, $h'_1 = p/g$, $\overline{h}_1 = G/g$
- Shallow-water model for one 1/2 vertical layer: $u'_2 = u$, $v'_2 = v$, $h'_2 = p/g*$, $\overline{h}_2 = G/g*$

Note that because the system is defined over a domain, it also needs boundary conditions. But we will introduce these boundary conditions later in the section.

3.3 Model Grid

Grid:

In order to solve the shallow-water model in equation 39 on a computer, we must consider a discrete representation of it on a model grid. The spatial grid chosen here is shown in Figure 4. We note the axis as follows:

- The discrete zonal axis is an array $x = [x_1, x_2..., x_{Nx}]$ where $x_i = i\Delta x$, $i = 1, ..., N_x$.
- The discrete meridional axis is an array $y = [y_1, y_2..., y_{N_y}]$ where $y_i = j\Delta y$, $j = 1, ..., N_y$.
- The discrete temporal axis is an array $t = [t_1, t_2..., t_{N_t}]$ where $t_k = k\Delta t$, where $k = 1, ..., N_t$.

Here N_x, N_y, N_t are the number of grid points along each dimension. The domain range is $\Delta x \leq x \leq L_x$, $\Delta y \leq y \leq L_y$, $\Delta t \leq t \leq L_t$ where $L_x = \Delta x N_x$, $L_y = \Delta y N_y$ and $L_t = \Delta t N_t$.

Discrete system of equations:

The variables u, v, p discretized on the grid are multidimensional arrays noted as:

$$\begin{aligned} u_{i,j,k} &= u(x_i, y_j, t_k) \\ v_{i,j,k} &= v(x_i, y_j, t_k) \\ p_{i,j,k} &= p(x_i, y_j, t_k) \end{aligned} \tag{40}$$

and we note the discrete derivatives as:

$$\frac{\partial}{\partial t}u(x_i, y_j, t_k) = \frac{\partial u}{\partial t}\mid_{i,j,k} \tag{41}$$

Therefore, the discrete version of the shallow-water model continuous equations 39 reads:

$$\begin{aligned}
&\tfrac{\partial u}{\partial t}\mid_{i,j,k} - f v_{i,j,k} = -\tfrac{\partial p}{\partial x}\mid_{i,j,k} \\
&\tfrac{\partial v}{\partial t}\mid_{i,j,k} + f u_{i,j,k} = -\tfrac{\partial p}{\partial y}\mid_{i,j,k} \\
&\tfrac{\partial p}{\partial t}\mid_{i,j,k} + G(\tfrac{\partial u}{\partial x}\mid_{i,j,k} + \tfrac{\partial v}{\partial y}\mid_{i,j,k}) = 0
\end{aligned} \tag{42}$$

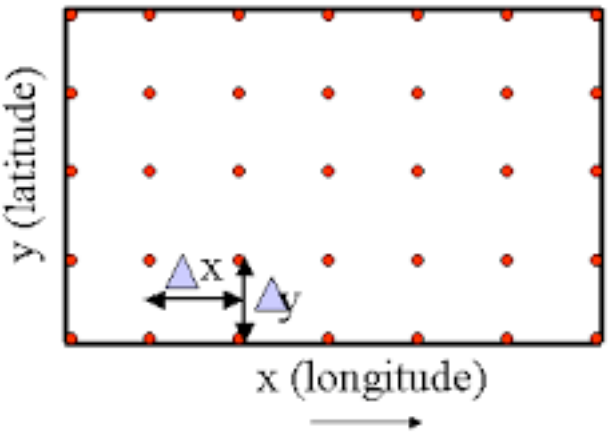

Figure 4: Spatial grid of the model.

3.4 Finite Difference Methods in Space and Time

In order to discretize the derivatives, we must choose finite difference methods. Here we choose an explicit Euler method in time (for $\partial/\partial t$) and centered scheme in space (for both $\partial/\partial x$ and $\partial/\partial y$):

$$\frac{\partial f}{\partial x}\mid_{i,j,k} \approx \frac{f_{i+1,j,k} - f_{i-1,j,k}}{2\Delta x}, \quad \frac{\partial f}{\partial y}\mid_{i,j,k} \approx \frac{f_{i,j+1,k} - f_{i,j-1,k}}{2\Delta y} \quad \text{and} \quad \frac{\partial f}{\partial t}\mid_{i,j,k} \approx \frac{f_{i,j,k+1} - f_{i,j,k}}{\Delta t} \tag{43}$$

The numerical method is of order 1 in time and order 2 in space. This is not a very accurate method but it is very inexpensive, which we prefer in order to shorten computation time. If we

develop we obtain the complete system of discrete equations:

$$u_{i,j,k+1} = u_{i,j,k} + \left(f v_{ijk} - \frac{p_{i+1,j,k} - p_{i-1,j,k}}{2\Delta x} \right) \Delta t \tag{44}$$

$$v_{i,j,k+1} = v_{i,j,k} + \left(-f u_{ijk} - \frac{p_{i,j+1,k} - p_{i,j-1,k}}{2\Delta y} \right) \Delta t \tag{45}$$

$$p_{i,j,k+1} = p_{i,j,k} - G \left(\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2\Delta y} \right) \Delta t. \tag{46}$$

which can predict the conditions at $t+\Delta t = (k+1)\Delta t$ provided all conditions are known at $t = k\Delta t$. For the above scheme we must satisfy two CFL conditions $\sqrt{G}\Delta t/\Delta x \leq 1$ and $\sqrt{G}\Delta t/\Delta y \leq 1$.

3.5 Boundary conditions

We must provide boundary conditions at the edge of the spatial domain ($i = 1$, $i = N_x$ and $j = 1$, $j = N_y$). These conditions must be justified physically, and it is also important to choose a finite difference method in space adapted to them. We show below different types of boundary conditions that can be considered in the shallow-water model depending on the physical problem it represents.

Periodic boundary conditions:

Consider an atmosphere that circles the equator along the zonal axis x, or similarly an ocean (aquaplanet). In that case the domain loops back to itself, and the boundary conditions are periodic:

$$\begin{aligned} u(0, y, t) &= u(L_x, y, t) \\ v(0, y, t) &= v(L_x, y, t) \\ p(0, y, t) &= p(L_x, y, t) \end{aligned} \tag{47}$$

The discrete version is easy to implement by looping the grid as well: for example the derivative from a centered scheme 2nd order reads:

$$\frac{\partial u}{\partial x} \mid_{1,j,k} = \frac{u_{2,j,k} - u_{N_x,j,k}}{2\Delta x}, \quad \frac{\partial u}{\partial x} \mid_{N_x,j,k} = \frac{u_{1,j,k} - u_{N_x-1,j,k}}{2\Delta x} \tag{48}$$

and similarly for u and v. We can also consider periodic boundary conditions along the meridional axis y. The periodic boundary conditions are the easiest to implement.

Reflecting boundary condition:

Consider an ocean with a fixed coast at $y = \Delta y$ oriented from east to west, that reflects all signals. In the case of the shallow-water model, the boundary condition read

$$\begin{aligned} u(x, \Delta y, t) &= 0 \\ v(x, \Delta y, t) &= 0 \\ \tfrac{\partial p}{\partial y}(x, \Delta y, t) &= 0 \end{aligned} \tag{49}$$

The condition $v = 0$ is easy to justify because there is no flow through the coast. The condition $\partial p/\partial y = 0$ assumes continuity of pressure between the ocean and the coast. Finally, the condition $u = 0$ is deduced from the other ones and the equations. The discrete version is easy to implement:

$$u_{i,1,k} = 0, \quad v_{i,1,k} = 0, \quad p_{i,1,k} = p_{i,2,k} \tag{50}$$

where we used $\partial p/\partial y = (p_{i,2,k} - p_{i,1,k})/\Delta y$ (with a first order schemes because it only uses grid points inside the domain).

Absorbing boundary condition:

On a side note, another possible boundary condition called absorbing boundary condition is $u, v, p = 0$. It is very easy to implement, however the physical justification for it is much less obvious for the present shallow-water model.

Open boundary conditions:

An open boundary condition is a boundary for which all signals are evacuated outside of the domain. This is, however, one of the most difficult boundary conditions to implement with the finite difference methods. A common strategy is to extend the model domain with a buffer layer in which the signals are dissipated progressively. This is proposed as an advanced lecture exercice

(see exerice section).

Implementation:

Figure 5 shows a sketch of areas of the grid domain that will likely have different numerical schemes. The inner domain (pink) follows equation 44-46, while the edges (cyan and orange) follow boundary conditions. The corners (green) may also be different when they combine two boundary conditions. Consider the example where we have periodic boundary conditions in x and reflecting boundary conditions at $y = \Delta y$ and $y = L_y$. If we use a centered scheme in space and explicit Euler scheme in time then the model prediction is different in each region:

Region $2 \leq i \leq N_x - 1$ and $2 \leq j \leq N_y - 1$ (pink) that is the inner domain:

$$u_{i,j,k+1} = u_{i,j,k} + \left(f v_{i,j,k} - \frac{p_{i+1,j,k} - p_{i-1,j,k}}{2\Delta x} \right) \Delta t \tag{51}$$

$$v_{i,j,k+1} = v_{i,j,k} + \left(-f u_{i,j,k} - \frac{p_{i,j+1,k} - p_{i,j-1,k}}{2\Delta y} \right) \Delta t \tag{52}$$

$$p_{i,j,k+1} = p_{i,j,k} - G \left(\frac{u_{i+1,j,k} - u_{i-1,j,k}}{2\Delta x} + \frac{v_{i,j+1,k} - v_{i,j-1,k}}{2\Delta y} \right) \Delta t. \tag{53}$$

Region $i = 1$ and $2 \leq j \leq N_y - 1$ (blue) with periodic boundary condition:

$$u_{1,j,k+1} = u_{1,j,k} + \left(f v_{1,j,k} - \frac{p_{2,j,k} - p_{N_x,j,k}}{2\Delta x} \right) \Delta t \tag{54}$$

$$v_{1,j,k+1} = v_{1,j,k} + \left(-f u_{1,j,k} - \frac{p_{1,j+1,k} - p_{1,j-1,k}}{2\Delta y} \right) \Delta t \tag{55}$$

$$p_{1,j,k+1} = p_{1,j,k} - G \left(\frac{u_{2,j,k} - u_{N_x,j,k}}{2\Delta x} + \frac{v_{1,j+1,k} - v_{1,j-1,k}}{2\Delta y} \right) \Delta t. \tag{56}$$

Region $i = N_x$ and $2 \leq j \leq N_y - 1$ (blue) with periodic boundary condition:

$$u_{N_x,j,k+1} = u_{N_x,j,k} + \left(f v_{N_x,j,k} - \frac{p_{1,j,k} - p_{N_x-1,j,k}}{2\Delta x} \right) \Delta t \tag{57}$$

$$v_{N_x,j,k+1} = v_{N_x,j,k} + \left(-f u_{N_x,j,k} - \frac{p_{N_x,j+1,k} - p_{N_x,j-1,k}}{2\Delta y} \right) \Delta t \tag{58}$$

$$p_{N_x,j,k+1} = p_{N_x,j,k} - G\left(\frac{u_{1,j,k} - u_{N_x-1,j,k}}{2\Delta x} + \frac{v_{N_x,j+1,k} - v_{N_x,j-1,k}}{2\Delta y}\right)\Delta t. \tag{59}$$

Region $1 \leq i \leq N_x$ and $j = 1$ (orange and green) with reflecting boundary condition:

$$u_{i,1,k+1} = 0, \quad v_{i,1,k+1} = 0, \quad p_{i,1,k+1} = p_{i,2,k+1} \text{(make sure to compute } p_{i,2,k+1} \text{ beforehand)} \tag{60}$$

Region $1 \leq i \leq N_x$ and $j = N_y$ (orange and green) with reflecting boundary condition:

$$u_{i,N_y,k+1} = 0, \quad v_{i,N_y,k+1} = 0, \quad p_{i,N_y,k+1} = p_{i,N_y-1,k+1} \text{(make sure to compute } p_{i,N_y-1,k+1} \text{ beforehand)} \tag{61}$$

In the matlab code, each of these predictions will be a different line of code. In the case above the corners (green) use the same predictions than the north and south edges (orange), but if the boundary conditions were different the predictions would be a different line of code as well.

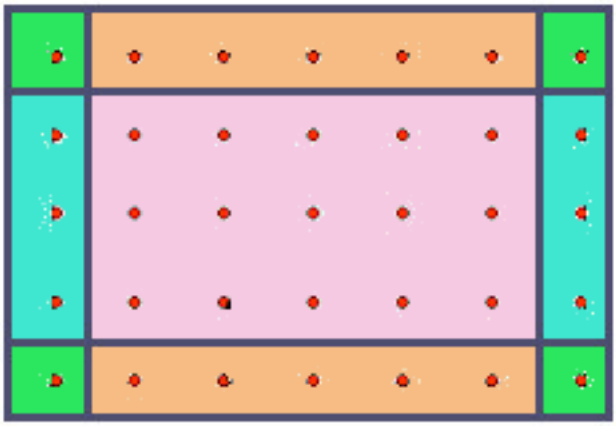

Figure 5: Areas of the spatial grid that may have difference numerical schemes (inner domain, edges and corners).

3.6 Integrating Solutions over time

The goal of a numerical simulation is most of the time to predict the evolution of the model in time. Assuming that we know all the conditions at a time t, we want to predict the conditions at $t + \Delta t$, and then repeat the process to predict following times.

The process can be summarized as follows, as shown in Figure 6:

1) Start from known initial conditions at $t = \Delta t$, $k = 1$. These are here $u_{i,j,1}$,$v_{i,j,1}$ and $p_{i,j,1}$ that are known at all i, j.

2) Use a numerical scheme to compute the conditions at $k = 2$.

3) Repeat the process to compute conditions at $k + 1$ knowing conditions at k. Repeat until end of simulation for $k = N_t$.

Beware, the amount of data generated can add up quickly, which takes time to compute and sometimes cannot be entirely stored. At first it is suggested to take N_x, N_y, N_t small (around 10-100 each) to test the code is working, then increase the values. Next, for a more efficient code it is better not to save the full conditions $u_{i,j,k}$, $v_{i,j,k}$, $p_{i,j,k}$ at all timesteps k, but for example each 10 or 100 timesteps (but thi

. For example, save a multidimensional array $u_{i,j,K}$, $v_{i,j,K}$, $p_{i,j,K}$ where $K = 100k$ (which means only save every 100 timesteps). You can also try increasing Δx, Δy and Δt (while keeping L_x, L_y, L_t the same) to make the computation time shorter, though always ensure to satisfy the stability criteria (CFL condition $\sqrt{G}\Delta t/\Delta x \leq 1$ in this model).

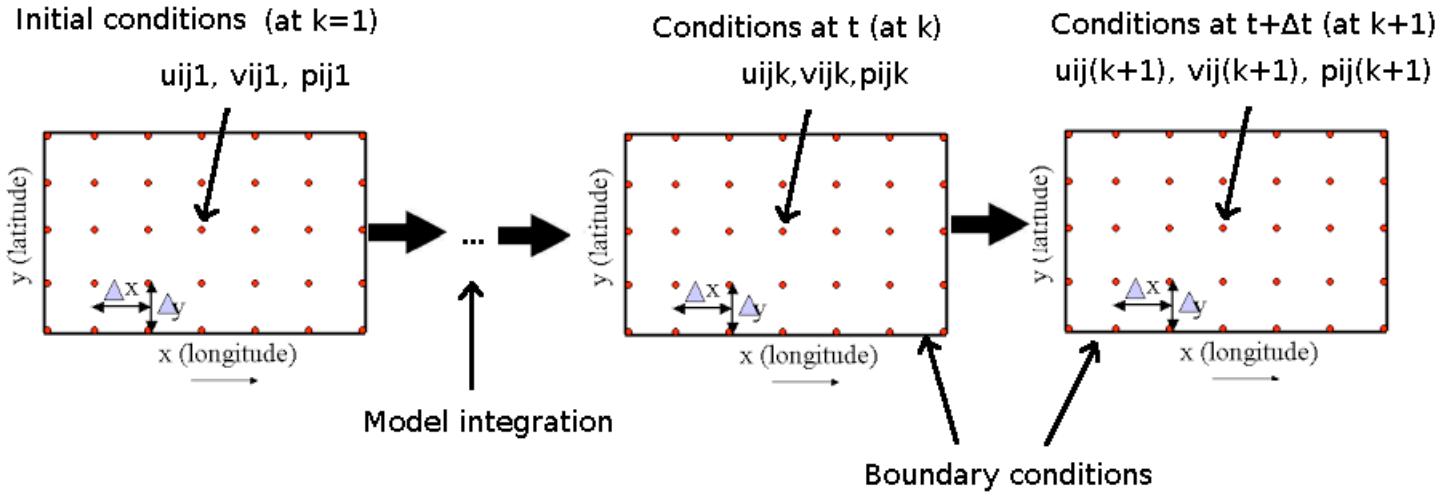

Figure 6: Sketch of a numerical simulation with the shalow-water model.

4 Summary

In this lecture on Computational Fluid Dynamics, the following elements are important to remember:

Finite Difference Methods:

Finite Difference methods are useful to represent a continuous system on a discrete grid. They take advantage of Taylor series:

$$\begin{aligned} f(t+\Delta t) &= f(t) + \Delta t \frac{df}{dt} \mid_t + O(\Delta t^2) \\ f(t+\Delta t) &= f(t) + \Delta t f \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + O(\Delta t^3) \\ f(t+\Delta t) &= f(t) + \Delta t \frac{df}{dt} \mid_t + \frac{\Delta t^2}{2} \frac{d^2 f}{dt^2} \mid_t + \frac{\Delta t^3}{6} \frac{d^3 f}{dt^3} \mid_t + O(\Delta t^4) \end{aligned} \tag{62}$$

to express derivatives in discrete form $f_k = f(t_k)$ with $t = k\Delta t$, for example:

$$\frac{df}{dt} \mid_k \approx \frac{f_{k+1} - f_k}{\Delta t} \text{ (Euler explicit)} \tag{63}$$

Many methods exist for treating derivatives in either space or time, and some important ones to remember all listed in Table 1 and Table 2.

Choosing Finite Difference Methods:

In order to choose a finite difference methods one has to consider the following criteria:

a) Cost to compute with arithmetic operations

b) Error r in representation (accuracy)

c) If numerical instability can happen

d) Other practical issues (aliasing, propagation of information...)

Usually, there is a trade-off to make between the factors and this depends on the physical problem that is represented. The error r is the error in representation of the continuous system by the discrete one. If the error is large or accumulates/magnifies over time then numerical instability

can happen. To avoid numerical stability several conditions must be met, and important ones are:

$$c\frac{\Delta t}{\Delta x} \leq 1 \text{ CFL condition (c is a propagation speed)} \tag{64}$$

$$\alpha\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \text{ Von Neumann condition } (\alpha \geq 0 \text{ is a diffusion coefficient}) \tag{65}$$

which can always be satisfied by decreasing Δt or increasing Δx. Some methods (e.g. implicit methods) can accept higher values but are more expensive to compute.

Building and Solving a Numerical Model:

The major components of a numerical model can be summarized as follows:

a) Starting Continuous system of equations

b) Model Grid

c) Discrete system obtained from finite difference methods

d) Boundary conditions at the limits of the domain

Then, from those conditions we can determine through time integration the numerical solutions for the model. We have given a complete tutorial to build and solve a numerical model for the shallow-water equations.

5 Exercices

1) Consider the equation

$$\frac{df}{dt} = af \tag{66}$$

where $a = 1$ or $a = -1$. Discretize the equation using an explicit Euler scheme. Compute numerical solutions in matlab for initial conditions $f(0) = 1$ and timestep $\Delta t = 0.1$, and $N_t = 100$. How do the solutions change depending on a ? See file odenum.m

2) Consider the linear transport equation:

$$\frac{\partial f}{\partial t} + c\frac{\partial f}{\partial x} = 0 \tag{67}$$

where $c = 1$ is a constant, and the domain is periodic ($f(0) = f(L)$). The inital conditions are $f(x, 0) = \cos(x2\pi/L)$. Discretize the equation using an explicit Euler scheme in time, and a backward scheme in space. Use $\Delta t = 0.1.$, $N_t = 100$, $\Delta x = 0.1$, $N_x = 100$. Plot the solution in matlab (see file pdenum.m).

We must have $\Delta t/\Delta x \leq c$ (CFL criterion) for the scheme to be adapted. What happens if $c = 2$? Lower Δt to avoid this problem.

Now take $c = -1$ and plot. What happens ? The reason for this is the forward scheme is not adapted, because propagation is the opposite way. Replace with a forward scheme and plot the solution (see file pdenum2.m)

3) Consider the heat equation:

$$\frac{\partial f}{\partial t} = \alpha\frac{\partial^2 f}{\partial x^2} \tag{68}$$

where $\alpha = 0.05$, and the domain is periodic ($f(0) = f(L)$). The inital conditions are $f(x, 0) = \cos(x2\pi/L)$. Discretize the equation using an explicit Euler scheme in time, and a centered scheme in space 2nd order. Use $\Delta t = 0.1.$, $N_t = 100$, $\Delta x = 0.1$, $N_x = 1000$. Plot the solution in matlab (see file pdenum4.m). What happens ? Try negative values of α and comment (these are unphysical and will lead to rapid growth of the magnitude $|f|$, try plotting $\log(|f|)$ to better see it).

4) Open boundary condition:

We provide here a strategy to implement an open boundary condition in the shallow-water model, called buffer layer. As an exercice, implement the open boundary condition in matlab for the shallow-water model. Test several values of α_b and L_b and verify that signals are evacuated outside of the model domain.

Assume that we implement the open boundary condition at $x = L$. In that case, we extend the model domain $\Delta x \leq x \leq L$ to $\Delta x \leq x \leq L_b$, and we modify the equations in the buffer layer as:

$$\begin{aligned} &\tfrac{\partial}{\partial t}u - fv = -\tfrac{\partial}{\partial x}p + \alpha\left(\tfrac{\partial^2 u}{\partial x^2} + \tfrac{\partial^2 u}{\partial y^2}\right) \\ &\tfrac{\partial}{\partial t}v + fu = -\tfrac{\partial}{\partial y}p + \alpha\left(\tfrac{\partial^2 v}{\partial x^2} + \tfrac{\partial^2 v}{\partial y^2}\right) \qquad \text{for } L \leq x \leq L_b \\ &\tfrac{\partial}{\partial t}p + G(\tfrac{\partial}{\partial x}u + \tfrac{\partial}{\partial y}v) = +\alpha\left(\tfrac{\partial^2 p}{\partial x^2} + \tfrac{\partial^2 p}{\partial y^2}\right) \end{aligned} \tag{69}$$

where the additional terms are diffusion terms with viscosity constant $\alpha \geq 0$ that dissipates the signals. For continuity with the model domain $\Delta x \leq x \leq L$ where $\alpha = 0$, we make α increase progressively up to an arbitrary value α_B. For example:

$$\alpha(x) = \alpha_B\left(\frac{x - L}{x - L_B}\right) \text{ for } L \leq x \leq L_B \tag{70}$$

for which $\alpha(L) = 0$ and $\alpha(L_B) = \alpha_B$. We also impose $u(L_B, y, t) = 0$, $v(L_B, y, t) = 0$ and $p(L_B, y, t) = 0$. The finite difference method is the same except for the second derivatives that we treat with a centered scheme:

$$\frac{\partial^2 u}{\partial x^2}\,|_{i,j,k} = \frac{u_{i+1,j,k} + 2u_{i,j,k} + u_{i-1,j,k}}{\Delta x^2}. \tag{71}$$