Solving Partial Differential Equations with Numerical Methods based on Finite Differences

Francesca Ribas, Physics Department, EETAC

April 19, 2018

Abstract

This document contains a summary of some notions you learned in Ampliació de Matemàtiques 2 (sections 1 and 2) and also a tutorial of how to solve numerically some examples of problems that are described by Partial Differential Equations like those of the projects of Fluid Mechanics (section 3). This document is not a complete description of how to solve partial differential equations with finite difference methods.

1 Finite difference approximation of derivatives

Considering a function $f(x)$, its derivative at a point x can be numerically approximated from the value of f in the surrounding points. First, a uniform mesh of points (spaced a constant small distance Δx) and the values of f at the mesh points are defined:

$$x_0, \ldots, x_i - \Delta x, x_i, x_i + \Delta x, x_i + 2\Delta x, \ldots, x_n$$

$$f_0, \ldots, f_{i-1}, f_i, f_{i+1}, f_{i+2}, \ldots, f_n$$

with f_i referring to the value of f at point x_i, $f_i = f(x_i)$.

From the Taylor expansion for $f(x_i + \Delta x)$ and $f(x_i - \Delta x)$, three approximations for the first derivative can be found:

- Forward difference
$$\frac{df}{dx} = \frac{f_{i+1} - f_i}{\Delta x} + O(\Delta x) \quad . \tag{1}$$
- Backward difference
$$\frac{df}{dx} = \frac{f_i - f_{i-1}}{\Delta x} + O(\Delta x) \quad . \tag{2}$$
- Central difference
$$\frac{df}{dx} = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + O(\Delta x^2) \quad . \tag{3}$$

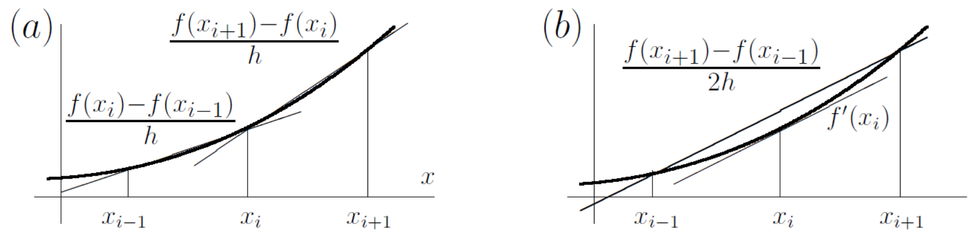
The forward and backward differences are approximations of first order because the error committed is of order Δx (in the formulas this is expressed as $O(\Delta x)$). The central difference is more precise ($O(\Delta x^2)$). The geometric interpretation of the three approximations of the first derivative is illustrated in Figure 1.

From the same Taylor expansion for $f(x_i + \Delta x)$ and $f(x_i - \Delta x)$, approximations for the second derivative can also be found, such as:

- Central difference
$$\frac{d^2 f}{dx^2} = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + O(\Delta x^2) \quad . \tag{4}$$

The previous approximations are extended easily to partial derivatives of functions of several variables such as $f(x, y, z, t)$ by changing d by ∂ in equations (1-4).

Ex1 Demonstrate equations (1-4) from the Taylor expansion for $f(x_i + \Delta x)$ and $f(x_i - \Delta x)$ (this is part of the contents of Amplicació de Matemàtiques 2).



(a)
$\frac{f(x_{i+1})-f(x_i)}{h}$
$\frac{f(x_i)-f(x_{i-1})}{h}$
x
x_{i-1}
x_i
x_{i+1}
(b)
$\frac{f(x_{i+1})-f(x_{i-1})}{2h}$
$f'(x_i)$
x_{i-1}
x_i
x_{i+1}

Figure 1: Sketch of the geometric interpretation of the forward and backward differences (panel a) and the central difference (panel b), the three approximations for the first derivative (equations 1-3). The h in the plots is the Δx of the text.

2 Overview of Partial Differential Equations

The classification of *Partial Differential Equations (PDEs)* is important for the numerical solution you choose. The general form of a second order lineal PDE for $f(x, y)$ is:

$$A(x,y)\frac{\partial^2 f}{\partial x^2} + 2B(x,y)\frac{\partial^2 f}{\partial x \partial y} + C(x,y)\frac{\partial^2 f}{\partial y^2} = D\left(x, y, f, \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \quad ,$$

where the variables x, y are coordinates. They have been chosen arbitrarily, they can also be z or the time coordinate t. The variable f is a function that can refer to any physical quantity, such as temperature T, a component of the velocity u, pressure p, stream function ψ, velocity potential ϕ, etc. Also, A, B, C and D are arbitrary functions.

- If $AC > B^2$ it is an *elliptic PDE.* An example is the Laplace equation for $f(x, y)$ $(A = C = 1, B = D = 0)$
$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0 \quad .$$
- If $AC < B^2$ it is a *hyperbolic PDE.* An example is the 1D wave equation for $f(x, t)$ $(A = 1, C = -c^2, B = D = 0)$
$$\frac{\partial^2 f}{\partial t^2} = c^2 \frac{\partial^2 f}{\partial x^2} \quad .$$
- If $AC = B^2$ it is a *parabolic PDE.* An example is the 1D heat equation for $f(x, t)$ $(A = \alpha, B = C = 0, D = \partial f/\partial t)$
$$\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial x^2} \quad .$$

For a complete description of a physical problem, the boundary conditions must also be provided and they can be of two different types.

- *Dirichlet conditions*: the value of the variable is imposed at the boundary. For example:
$$f(x = 0, t) = C_1(t) \quad , \quad f(x = L, t) = C_2(t) \quad .$$
- *Neumann conditions*: the value of the variable derivative is imposed at the boundary. For example:
$$\frac{\partial f}{\partial x}(x = 0, t) = C_3(t) \quad , \quad \frac{\partial f}{\partial x}(x = L, t) = C_4(t) \quad .$$

If the problem is unsteady, initial conditions are also necessary (e.g., $f(x, t = 0) = C_5(x)$).

3 Examples of numerical methods for PDEs

To solve a PDE with numerical methods based on finite differences, it is necessary to discretize i) the time and/or spatial coordinates (i.e., to create a grid), ii) the governing equations and iii) the boundary and initial conditions (if needed).

Numerical methods to solve PDEs are generally classified as:

- *Explicit mehods*: There is a direct formula for all unknown values in the grid as a function of previously computed values. They are the simplest methods but are are stable only for certain step sizes, the latter being sometimes governed by the so called Courant condition (and some other times they are never stable!).
- *Implicit methods*: There is no explicit formula at each point, only a set of simultaneous equations that must be solved over the whole grid. They are stable for all step sizes but they are more complex to implement and we will not use them in the project of Fluid Mechanics.

3.1 Explicit numerical method for a parabolic PDE

To illustrate these methods, in this section we will focus on the 1D parabolic equation for $f(t, y)$

$$\frac{\partial f}{\partial t} = \alpha \frac{\partial^2 f}{\partial y^2} \quad . \tag{5}$$

with the Dirichlet boundary conditions and an initial boundary condition

$$f(t, y = 0) = C_1 \quad , \; f(t, y = L) = C_2 \quad , \; f(t = 0, y) = C_3 \quad . \tag{6}$$

Here, C_1, C_2 and C_3 are positive constants. Equation (5) can represent the heat (or energy balance) equation if $f = T$ (temperature) and the Navier Stokes (or momentum balance) equation if $f = u$ (x-component of the velocity). This equation will be applied in two types of project. Notice that this problem has an analytical solution for the steady state (when $t \to \infty$ and $\partial f/\partial t = 0$).

We consider $f(t, y)$ for $0 \le t \le T$ and $0 \le y \le L$ (where T is the final simulated time and L is the total length) and discretize time t and spatial coordinate y using time step size $\Delta t = T/m$ and spatial step size $\Delta y = L/n$. We take $t_k = k\,\Delta t$ with $0 \le k \le m$, $y_j = j\,\Delta y$ with $0 \le j \le n$ and $f_j^k = f(t_k, y_j)$ (see figure 2). There are $n + 1$ spatial grid points, from which $y_1, ..., y_{n-1}$ are inside the domain and y_0 and y_n are at the boundaries. Also, there are $m + 1$ temporal grid points and t_0 is the initial time.

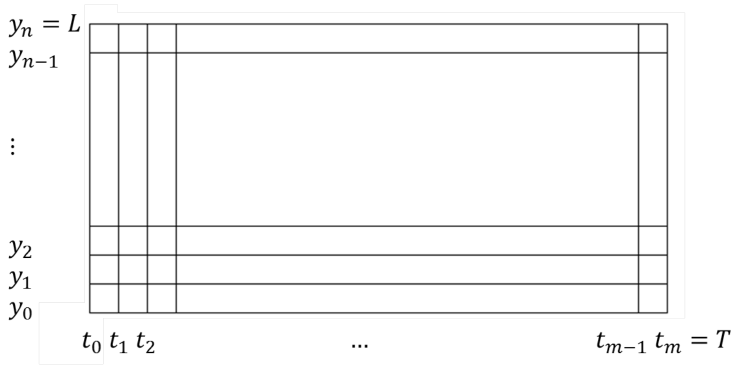


Figure 2: Sketch of the spatial and temporal grid for the example of 1D parabolic PDE.

Ex2 Making use of equations (1-4), discretize equation (5) and isolate f_j^{k+1} as a function of f_{j-1}^k, f_j^k and f_{j+1}^k. To obtain it, use the most precise approximations of the derivatives that you can choose. Also, write the discretized version of the boundary and initial conditions (6).

The following numerical scheme for a 1D parabolic equation corresponding to the discretization in Exercise 2 is very intuitive and can be implemented in any programming language.

- *Step 1*: Define constants, vectors (e.g., y_j, t_k) and matrices (e.g., f_j^k) to be used within the code.
- *Step 2*: Write down the values of the discretized coordinates (y_j, t_k), the discretized boundary conditions for f at y_0 and y_n and the discretized initial condition for f at t_0 (Exercise 2).
- *Step 3*: Write down a loop to compute the solution for f at the subsequent times (t_1, t_2, etc.) by imposing the discretized equation you found in Exercise 2.
- *Step 4*: Compute secondary magnitudes, if required, and plot the results.

Ex3 Implement a Python code to find the numerical solution of equation (5) and the boundary conditions (6), applying the four steps described above. Use the following physical parameter values, $\alpha = 0.01$, $C_1 = 0$, $C_2 = 2$, $C_3 = 0$, $L = 10$, $T = 2000$, and the following numerical parameter values, $n = 10$, $m = 40$, where all values are given in international units. Plot the solutions at times t_{10}, t_{20} and t_{30}. (Hint: Use the Python function *plt.plot(f,y)*, where the Python *matplotlib* has been imported as *plt*). Interpret the result and compare it with the analytical steady solution of the problem.

3.2 Explicit numerical method for an elliptic PDE

To illustrate these types of methods, in this section we will focus on the 2D elliptic equation for $f(x, y)$

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0 \quad . \tag{7}$$

with the Dirichlet boundary conditions

$$\begin{aligned} f(x=0,y) = K\,y \ \ , \ f(x=L_x,y) = K\,y \ \ , \\ f(x,y=0) = 0 \ \ , \ f(x,y=L_y) = K\,L_y \ \ . \end{aligned} \tag{8}$$

Here, K is a positive constant. Equation (7) is the 2D Laplace equation for 2D incompressible potential flows if $f = \psi$ (Stream function) or if $f = \phi$ (velocity potential). This equation will be applied in one type of project.

We consider $f(x, y)$ for $0 \le x \le L_x$ and $0 \le y \le L_y$ and discretize the two spatial coordinates x, y using spatial step sizes $\Delta x = L_x/m$ and $\Delta y = L_y/n$, so that: $x_i = i\,\Delta x$ with $0 \le i \le m$, $y_j = j\,\Delta y$ with $0 \le j \le n$ and $f_{i,j} = f(x_i, y_j)$ (see figure 3). There are $m+1$ times $n+1$ grid points.

Ex4 Making use of equations (1-4), discretize equation (7) and isolate $f_{i,j}$ as a function of $f_{i-1,j}$, $f_{i+1,j}$, $f_{i,j-1}$ and $f_{i,j+1}$. To obtain it, use the most precise approximations of the derivatives that you can choose. Also, discretize the boundary conditions (8).

The numerical scheme corresponding to the 2D elliptic equation and the discretization of Exercise 4 is less intuitive than that corresponding to the parabolic equation of the previous section. The trick consists of making an initial guess for the values of $f_{i,j}$ inside the domain. Then this "wrong" solution is refined by making several iterations in which the discretized version of the equation and the "correct" boundary conditions are imposed.

- *Step 1*: Define constants, vectors (e.g., x_i, y_j) and matrices (e.g., $f_{i,j}$) to be used within the code.
- *Step 2*: Write down the values of the discretized coordinates (x_i, y_j) and the discretized boundary conditions for f at x_0, x_m, y_0 and y_n (Exercise 4). Write an initial guess for $f_{i,j}$ inside the domain: a good starting point is using a constant value that falls within the known values of f at the boundaries.
- *Step 3*: Write down a loop to compute improved values of $f_{i,j}$ inside the domain by imposing the discretized equation you found in Exercise 4. In general, do as many iterations as needed to reach a "good result" (you must think about what does "good result" means).
- *Step 4*: Compute secondary magnitudes, if required, and plot the results.

Ex5 Implement a Python code to find the numerical solution of equation (7) and the boundary conditions (8), applying the four steps described above. Use the following physical parameter values, $K = 2$, $L_x = 100$, $L_y = 100$, and the following numerical parameter values, $m = 10$, $n = 10$, where all values are given in international units. As initial guess for $f_{i,j}$ inside the domain, use the average of the values at the lower and upper boundaries ($f_{i,0}$, $f_{i,n}$). Plot the solution for $f_{i,j}$ after 10, 20 and 30 iterations (Hint: Use the Python function *plt.contourf(x,y,f,20)*). Interpret the result and discuss if you think this is already a "good result" for $f(x,y)$.

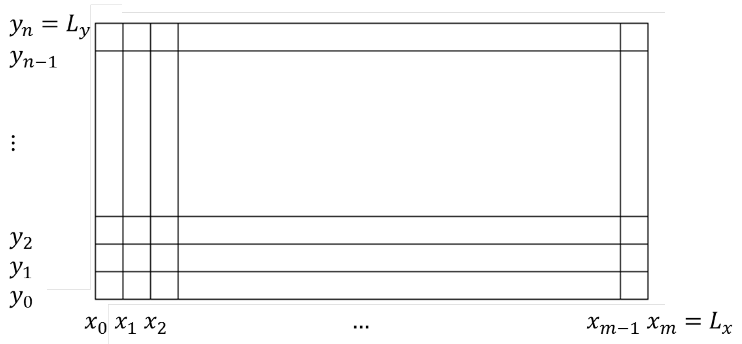


Figure 3: Sketch of the 2D spatial grid for the example of 2D elliptic PDE.