INTEGRATION OF PARTIAL DIFFERENTIAL EQUATIONS

I From a Partial Differential Equation to a Finite-Difference scheme 1

II Implementing Boundary Conditions 10

III Solving Finite-Difference equations for Initial Value Problems and Boundary Value Problems 15

IV Solving Finite-Difference equations for Boundary Value Problems 17

V Beyond Finite-Difference methods 21

In this chapter, we discuss the integration of Partial Differential Equations (PDEs).

Color code for the chapter:

- Definitions and important results are given in pink boxes.
- Examples are given in green boxes.
- Remarks and notions that can be skipped for the first read are in **black** boxes.
- Links are emphasized in blue.

The topic of numerical integration of partial differential equations is very broad and it is impossible to cover all possible methods in these lectures notes. We focus here on one class of methods to solve PDEs, namely, the **Finite-Difference** (FD) methods. For a very detailed presentation of finite-difference methods, look at J. W. Thomas, *Numerical Partial Differential Equations. Finite Difference Methods*. More involved methods, *e.g.*, finite-element methods or spectral methods will be briefly mentioned in a closing section.

Most of the algorithmic schemes we introduce in these notes will be exemplified by model linear PDEs in one or two dimensions of space, *e.g.*, the heat equation, the wave equation or the Poisson equation. For more complex or non-linear PDEs, you will have to adapt the methods presented in these notes.

We assume that the reader knows what a PDE is. However, no mathematical knowledge about PDEs is required to follow these lecture notes.

I. From a Partial Differential Equation to a Finite-Difference scheme

1. Introduction

In these lecture notes, we mostly discuss three archetypal equations:

- the heat equation

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = D\Delta\phi, \\ \phi(x,y,0) = \phi_{\mathrm{i}}(x,y), \end{cases} \tag{1}$$

with $\Delta\phi = \displaystyle\sum_{k=1}^{D} \frac{\partial^2 \phi}{\partial x_k^2}$ the Laplacian in D dimensions of space;

- the wave equation

$$\begin{cases} \dfrac{\partial^2 \phi}{\partial t^2} - c^2 \Delta \phi = 0, \\ \phi(x, y, 0) = \phi_{\mathrm{i}}(x, y), \\ \dfrac{\partial \phi}{\partial t}(x, y, 0) = \psi_{\mathrm{i}}(x, y), \end{cases} \tag{2}$$

  with $\Delta\phi$ the Laplacian in D dimensions;

- the Poisson equation

$$\Delta \phi = f, \tag{3}$$

  for a source term f.

All these PDEs must be supplemented with **boundary conditions** for the problem to be well posed (see the section about **Boundary Conditions**). They are thus all Boundary Value Problems (BVPs). The first two equations are also Initial Value Problems (IVPs) because they explicitly involve time and an initial condition must be provided in order for the problem to be well-posed.

> PDEs are usually classified according to the structure of their higher-order derivatives. The heat equation is an example of parabolic PDEs, the wave equation an example of hyperbolic PDEs, and the Poisson equation an example of elliptic PDEs. For our discussion, this partition is not important.

2. Discretizing time and space

In a similar way as ODEs, we aim to solve these PDEs by discretizing space and time. We first present the idea in the case of Cartesian coordinates, and we discuss then the case of polar coordinates.

a. Cartesian coordinates

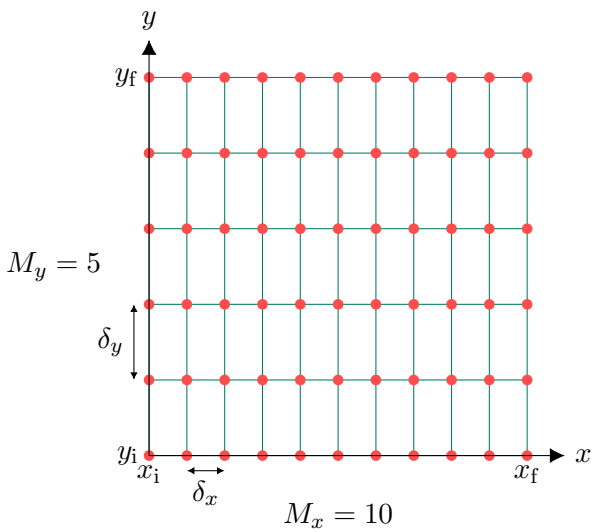

Figure 1: **Illustration of the spatial discretization in order to solve a PDE with a Finite-Difference method.** The rectangle $[x_{\mathrm{i}},\, x_{\mathrm{f}}] \times [x_{\mathrm{i}},\, y_{\mathrm{f}}]$ is decomposed into $(M_x + 1) \times (M_y + 1)$ nodes. The distance between two consecutive nodes is δ_x in the x-direction, and δ_y in the y-direction.

Imagine that you want to solve a PDE in 2 dimensions of space $(x,\, y) \in [x_{\mathrm{i}},\, x_{\mathrm{f}}] \times [y_{\mathrm{i}},\, y_{\mathrm{f}}]$. To discretize space you define nodes arranged in a lattice. On these nodes, you want to approximate the PDE solution (see Fig. 1). The nodes have coordinates $(x_j,\, y_k)$, where $x_j = x_{\mathrm{i}} + j\delta_x$ and $y_k = y_{\mathrm{i}} + k\delta_y$, with $j \in [\![0,\, M_x]\!]$ and $k \in [\![0,\, M_y]\!]$. The numbers of nodes $M_x + 1$ and $M_y + 1$ in each direction are such that $x_{\mathrm{f}} = x_{\mathrm{i}} + M_x \delta_x$ and $y_{\mathrm{f}} = y_{\mathrm{i}} + M_y \delta_y$.

Similarly, if time is involved in the problem you discretize the time interval $[t_\mathrm{i},\, t_\mathrm{f}]$ with a time step h, and you define instants $t_n = t_\mathrm{i} + nh$, with $n \in [\![0,\, N]\!]$ where you want to approximate the PDE solution. The estimate of the numerical solution at $(x_j,\, y_k,\, t_n)$ is denoted ϕ^n_{jk}. Based on this spatio-temporal discretization, we now approximate spatial and temporal derivatives.

Time derivatives. For the time derivative, only the first-order derivative is relevant. Indeed, for higher-order time derivatives, we can always transform the PDE into a system of coupled PDEs which only involve first-order time derivatives, as we did for ODEs. This is illustrated in the example below.

Consider the wave equation (2). If we define

$$\psi(x,y,t) = \frac{\partial \phi}{\partial t}(x,y,t),$$

then Eq. (2) is equivalent to the coupled system of first-order PDEs:

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = \psi, \\ \dfrac{\partial \psi}{\partial t} = c^2 \Delta \phi, \\ \phi(x,y,0) = \phi_\mathrm{i}(x,y) \\ \psi(x,y,0) = \psi_\mathrm{i}(x,y). \end{cases}$$

Just like for the time integration of ODEs, we can distinguish forward time derivatives and backward time derivatives.

Finite-Difference representation of time derivatives
The **Forward Time** (FT) representation of the time derivative is

$$\left.\frac{\partial \phi}{\partial t}\right|^n_{jk} = \frac{\phi^{n+1}_{jk} - \phi^n_{jk}}{h} + O(h). \tag{4}$$

This FD approximation is forward because we use the value of the function at the current time t_n and at the next time t_{n+1} to compute the slope at time t_n.

The **Backward Time** (BT) representation of the time derivative is

$$\left.\frac{\partial \phi}{\partial t}\right|^{n+1}_{jk} = \frac{\phi^{n+1}_{jk} - \phi^n_{jk}}{h} + O(h). \tag{5}$$

This FD approximation is backward because we use the value of the function at the current time t_{n+1} and at the previous time t_n to compute the slope at time t_{n+1}.

For both Finite-Difference (FD) representations, the truncated terms are of order $O(h)$.

Spatial derivatives. We similarly approximate spatial derivatives via different FD formulas.

Finite-Difference representation of first-order spatial derivatives
The **Forward Space** (FS) representation of the first-order spatial derivative is

$$\begin{cases} \left.\dfrac{\partial \phi}{\partial x}\right|^n_{jk} = \dfrac{\phi^n_{j+1,k} - \phi^n_{jk}}{\delta_x} + O(\delta_x), \\ \left.\dfrac{\partial \phi}{\partial y}\right|^n_{jk} = \dfrac{\phi^n_{j,k+1} - \phi^n_{jk}}{\delta_y} + O(\delta_y). \end{cases} \tag{6}$$

This FD approximation is forward because we use the value of the function at the current x-position x_j (respectively y-position y_k) and at the right x-position x_{j+1} (respectively y-position y_{k+1}) to compute the slope at x-position x_j (respectively y-position y_k).

The **Backward Space** (BS) representation of the first-order spatial derivative is

$$\begin{cases} \left.\dfrac{\partial \phi}{\partial x}\right|_{jk}^{n} = \dfrac{\phi_{jk}^{n} - \phi_{j-1,k}^{n}}{\delta_x} + O(\delta_x), \\ \left.\dfrac{\partial \phi}{\partial y}\right|_{jk}^{n} = \dfrac{\phi_{jk}^{n} - \phi_{j,k-1}^{n}}{\delta_y} + O(\delta_y). \end{cases} \tag{7}$$

This FD approximation is backward because we use the value of the function at the current x-position x_j (respectively y-position y_k) and at the left x-position x_{j-1} (respectively y-position y_{k-1}) to compute the slope at x-position x_j (respectively y-position y_k).

The **Centered Space** (CS) representation of the first-order spatial derivative is

$$\begin{cases} \left.\dfrac{\partial \phi}{\partial x}\right|_{jk}^{n} = \dfrac{\phi_{j+1,k}^{n} - \phi_{j-1,k}^{n}}{2\delta_x} + O(\delta_x^2), \\ \left.\dfrac{\partial \phi}{\partial y}\right|_{jk}^{n} = \dfrac{\phi_{j,k+1}^{n} - \phi_{j,k-1}^{n}}{2\delta_y} + O(\delta_y^2). \end{cases} \tag{8}$$

This FD approximation is centered because we use the value of the function at the right x-position x_{j+1} (respectively y-position y_{k+1}) and at the left x-position x_{j-1} (respectively y-position y_{k-1}) to compute the slope at the current x-position x_j (respectively y-position y_k).

For both FS and BS representations, the truncated terms are of order $O(\delta_x,\, \delta_y)$. For the CS representation, the truncated terms are instead of order $O(\delta_x^2,\, \delta_y^2)$.

The verification of the above formulas is a good exercise (do it before looking at the correction in the example below!).

For the FS representation, we do a Taylor expansion of ϕ at $(x_{j+1},\, y_k,\, t_n)$:

$$\phi(x_j + \delta_x, y_k, t_n) = \phi(x_j, y_k, t_n) + \delta_x \frac{\partial \phi}{\partial x}(x_j, y_k, t_n) + \frac{\delta_x^2}{2}\frac{\partial^2 \phi}{\partial x^2}(x_j, y_k, t_n) + O(\delta_x^3),$$
$$\phi_{j+1,k}^{n} = \phi_{jk}^{n} + \delta_x \left.\frac{\partial \phi}{\partial x}\right|_{jk}^{n} + \frac{\delta_x^2}{2} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{jk}^{n} + O(\delta_x^3),$$

leading to

$$\frac{\phi_{j+1,k}^{n} - \phi_{jk}^{n}}{\delta_x} = \left.\frac{\partial \phi}{\partial x}\right|_{jk}^{n} + \frac{\delta_x}{2} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{jk}^{n} + O(\delta_x^2) = \left.\frac{\partial \phi}{\partial x}\right|_{jk}^{n} + O(\delta_x).$$

The calculation is similar for the BS representation.

For the CS representation, we also do a Taylor expansion of ϕ at $(x_{j-1},\, y_k,\, t_n)$:

$$\phi(x_j - \delta_x, y_k, t_n) = \phi(x_j, y_k, t_n) - \delta_x \frac{\partial \phi}{\partial x}(x_j, y_k, t_n) + \frac{\delta_x^2}{2}\frac{\partial^2 \phi}{\partial x^2}(x_j, y_k, t_n) + O(\delta_x^3),$$
$$\phi_{j-1,k}^{n} = \phi_{jk}^{n} - \delta_x \left.\frac{\partial \phi}{\partial x}\right|_{jk}^{n} + \frac{\delta_x^2}{2} \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{jk}^{n} + O(\delta_x^3),$$

leading to

$$\frac{\phi_{j+1,k}^{n} - \phi_{j-1,k}^{n}}{2\delta_x} = \left.\frac{\partial \phi}{\partial x}\right|_{jk}^{n} + O(\delta_x^2).$$

Contrary to time derivatives, you might need to approximate higher-order spatial derivatives.

Finite-Difference representation of higher-order spatial derivatives
The **Centered Space** (CS) representation of the second-order spatial derivative is

$$\begin{cases} \left.\dfrac{\partial^2\phi}{\partial x^2}\right|_{jk}^n = \dfrac{\phi_{j+1,k}^n + \phi_{j-1,k}^n - 2\phi_{jk}^n}{\delta_x^2} + O(\delta_x^2), \\ \left.\dfrac{\partial^2\phi}{\partial y^2}\right|_{jk}^n = \dfrac{\phi_{j,k+1}^n + \phi_{j,k-1}^n - 2\phi_{jk}^n}{\delta_y^2} + O(\delta_y^2). \end{cases} \tag{9}$$

The verification of the above formulas is a good exercise (do it before looking at the correction in the example below!).

We do a Taylor expansion of ϕ at $(x_{j\pm1},\, y_k,\, t_n)$:

$$\phi(x_j \pm \delta_x, y_k, t_n) = \phi(x_j, y_k, t_n) \pm \delta_x \frac{\partial\phi}{\partial x}(x_j, y_k, t_n) + \frac{\delta_x^2}{2}\frac{\partial^2\phi}{\partial x^2}(x_j, y_k, t_n) \pm \frac{\delta_x^3}{6}\frac{\partial^3\phi}{\partial x^3}(x_j, y_k, t_n) + O(\delta_x^4),$$
$$\phi_{j\pm1,k}^n = \phi_{jk}^n \pm \delta_x \left.\frac{\partial\phi}{\partial x}\right|_{jk}^n + \frac{\delta_x^2}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{jk}^n \pm \frac{\delta_x^3}{6}\left.\frac{\partial^3\phi}{\partial x^3}\right|_{jk}^n + O(\delta_x^4),$$

leading to

$$\phi_{j+1,k}^n + \phi_{j-1,k}^n = 2\phi_{jk}^n + \delta_x^2 \left.\frac{\partial^2\phi}{\partial x^2}\right|_{jk}^n + O(\delta_x^4).$$

From this last equation, you easily get the result presented above.

If you need other higher-order spatial derivatives, you just have to combine several Taylor expansions, as it is done in the example above.

You can construct FD representations of spatial derivatives which are more refined than the above estimates. For that we need to use more nodes. We illustrate this with the example of a more accurate FS representation of first-order spatial derivatives. We do a Taylor expansion of ϕ at $(x_{j+1},\, y_k,\, t_n)$ and at $(x_{j+2},\, y_k,\, t_n)$:

$$\phi(x_j + \delta_x, y_k, t_n) = \phi(x_j, y_k, t_n) + \delta_x \frac{\partial\phi}{\partial x}(x_j, y_k, t_n) + \frac{\delta_x^2}{2}\frac{\partial^2\phi}{\partial x^2}(x_j, y_k, t_n) + O(\delta_x^3),$$
$$\phi_{j+1,k}^n = \phi_{jk}^n + \delta_x \left.\frac{\partial\phi}{\partial x}\right|_{jk}^n + \frac{\delta_x^2}{2}\left.\frac{\partial^2\phi}{\partial x^2}\right|_{jk}^n + O(\delta_x^3),$$

and

$$\phi(x_j + 2\delta_x, y_k, t_n) = \phi(x_j, y_k, t_n) + 2\delta_x \frac{\partial\phi}{\partial x}(x_j, y_k, t_n) + 2\delta_x^2\frac{\partial^2\phi}{\partial x^2}(x_j, y_k, t_n) + O(\delta_x^3),$$
$$\phi_{j+2,k}^n = \phi_{jk}^n + 2\delta_x \left.\frac{\partial\phi}{\partial x}\right|_{jk}^n + 2\delta_x^2 \left.\frac{\partial^2\phi}{\partial x^2}\right|_{jk}^n + O(\delta_x^3).$$

We can then get rid of the second-order derivative with the combination:

$$4\phi_{j+1,k}^n - \phi_{j+2,k}^n = 3\phi_{jk}^n + 2\delta_x \left.\frac{\partial\phi}{\partial x}\right|_{jk}^n + O(\delta_x^3),$$

leading to

$$\left.\frac{\partial\phi}{\partial x}\right|_{jk}^n = \frac{4\phi_{j+1,k}^n - \phi_{j+2,k}^n - 3\phi_{jk}^n}{2\delta_x} + O(\delta_x^2).$$

b. Polar coordinates

So far we have discussed Cartesian coordinates, but sometimes boundary conditions make other coordinate systems (polar, cylindrical, spherical, etc.) more suitable. We show how to find the FD representation of spatial derivatives in other coordinate systems, taking the example of the Laplacian in polar coordinates:

$$\Delta\phi = \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2}.$$

We see that there are two differences with respect to the Laplacian in Cartesian coordinates. First, for the term involving derivatives with respect to θ, there is a prefactor which varies in space. Second, for the term involving derivatives with respect to r, this is not the function ϕ which is derived.

To derive a FD representation of the Laplacian, we first discretize space as for Cartesian coordinates. The nodes have polar coordinates $(r_j,\, \theta_k)$, where $r_j = j\delta_r$ and $\theta_k = k\delta_\theta$, with $j \in [\![0,\, M_r]\!]$ and $k \in [\![0,\, M_\theta]\!]$. The nodes are regularly spaced in radius (with a step size δ_r) and in angle (with a step size δ_θ). The number of nodes in each direction are such that $r_\mathrm{f} = M_r\delta_r$ and $2\pi = M_\theta\delta_\theta$. The estimate of the numerical solution at $(r_j,\, \theta_k,\, t_n)$ is denoted ϕ^n_{jk}. We stress that all ϕ^n_{0k} are equal because they all correspond to the same point in space, namely, the origin: we simply note it ϕ^n_0.

To approximate the term involving derivatives with respect to θ, we can simply use the FD representation derived for Cartesian coordinates:

$$\frac{1}{r^2}\frac{\partial^2\phi}{\partial\theta^2}\bigg|^n_{jk} = \frac{1}{r_j^2}\left[\frac{\phi^n_{j,k+1} + \phi^n_{j,k-1} - 2\phi^n_{jk}}{\delta_\theta^2}\right] + O(\delta_\theta^2).$$

However, you can see that this term diverges for $j = 0$, which is problematic! We discuss this problem below.

> Note that the recipe we used here to obtain the FD representation can be generalized to terms like $a(x)\partial\phi/\partial x$ in Cartesian coordinates.

For the term involving derivates with respect with r, we have to be more cautious. We first use a CS representation of the exterior derivative:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right)\bigg|^n_{jk} = \frac{1}{r_j}\left[\frac{\left(r\dfrac{\partial\phi}{\partial r}\right)\bigg|^n_{j+1/2,k} - \left(r\dfrac{\partial\phi}{\partial r}\right)\bigg|^n_{j-1/2,k}}{\delta_r}\right] + O(\delta_r^2)$$

Note that we have considered imaginary nodes of index $j\pm1/2$ between node j and node $j\pm1$ in the r-direction. We then apply a second CS representation for the remaining derivatives:

$$\left(r\frac{\partial\phi}{\partial r}\right)^n_{j+1/2,k} = r_{j+1/2}\left(\frac{\phi^n_{j+1,k} - \phi^n_{jk}}{\delta_r}\right) + O(\delta_r^2),$$

where $r_{j+1/2} = (j+1/2)\delta_r$. Similarly,

$$\left(r\frac{\partial\phi}{\partial r}\right)^n_{j-1/2,k} = r_{j-1/2}\left(\frac{\phi^n_{jk} - \phi^n_{j-1,k}}{\delta_r}\right) + O(\delta_r^2),$$

where $r_{j-1/2} = (j-1/2)\delta_r$. We finally end up with

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\phi}{\partial r}\right)\bigg|^n_{jk} = \frac{1}{r_j}\left[\frac{r_{j+1/2}\left(\phi^n_{j+1,k} - \phi^n_{jk}\right) - r_{j-1/2}\left(\phi^n_{jk} - \phi^n_{j-1,k}\right)}{\delta_r^2}\right] + O(\delta_r^2).$$

You can note that, finally, the FD representation only involves the values of the function at real nodes $j-1$, j, $j+1$. However, the formula is again problematic when $j = 0$.

Finite-Difference representation of the Laplacian in polar coordinates
The CS representation of the Laplacian in polar coordinates is

$$\Delta\phi|^n_{jk} = \frac{1}{r_j}\left[\frac{r_{j+1/2}\left(\phi^n_{j+1,k} - \phi^n_{jk}\right) - r_{j-1/2}\left(\phi^n_{jk} - \phi^n_{j-1,k}\right)}{\delta_r^2}\right] + \frac{1}{r_j^2}\left[\frac{\phi^n_{j,k+1} + \phi^n_{j,k-1} - 2\phi^n_{jk}}{\delta_\theta^2}\right] + O(\delta_r^2, \delta_\theta^2). \tag{10}$$

If the integration domain includes the origin ($r_0 = 0$) then the above formula is not valid there.

To cure the problem for $j = 0$ (around the origin), you usually need to integrate the PDE in a small disk around the origin and use approximations of the integrals involved. This is illustrated in the remark/example below for the curious reader.

We illustrate the procedure to cure the problem at $j = 0$ in the case of the heat equation. We integrate Eq. (1) on a small disk of radius $\delta_r/2$ and between times t_n and t_{n+1}:

$$\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{\partial \phi}{\partial t}(r,\theta,t) = D \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \phi}{\partial r}\right)(r,\theta,t) + D \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{1}{r^2}\frac{\partial^2 \phi}{\partial \theta^2}(r,\theta,t).$$

The second integral in the right-hand side vanishes because of the 2π-periodicity of ϕ with respect to θ:

$$\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{\partial^2 \phi}{\partial \theta^2}(r,\theta,t) = \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \left[\frac{\partial \phi}{\partial \theta}(r,\theta,t)\right]_{\theta=0}^{\theta=2\pi} = 0.$$

We can also simplify the integral in the left-hand side by performing the time integration:

$$\begin{aligned}\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{\partial \phi}{\partial t}(r,\theta,t) &= \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta \left[\phi(r,\theta,t)\right]_{t=t_n}^{t=t_{n+1}} \\ &= \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \left[\phi(r,\theta,t_{n+1}) - \phi(r,\theta,t_n)\right].\end{aligned}$$

We can finally simplify the first integral in the right-hand side by simplifying by r and performing the integration with respect to r:

$$\begin{aligned}\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \phi}{\partial r}\right)(r,\theta,t) &= \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{2\pi} \mathrm{d}\theta \left[r\frac{\partial \phi}{\partial r}(r,\theta,t)\right]_{r=0}^{r=\delta_r/2} \\ &= \frac{\delta_r}{2} \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{2\pi} \mathrm{d}\theta\, \frac{\partial \phi}{\partial r}(\delta_r/2,\theta,t).\end{aligned}$$

We are thus left with the following equality:

$$\int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \left[\phi(r,\theta,t_{n+1}) - \phi(r,\theta,t_n)\right] = \frac{D\delta_r}{2} \int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{2\pi} \mathrm{d}\theta\, \frac{\partial \phi}{\partial r}(\delta_r/2,\theta,t).$$

We now approximate the two above integrals. For the integral in the left-hand side, we use the left rectangular rule for the integration over r, after the change of variable $s = r^2$:

$$\begin{aligned}\int_0^{\delta_r/2} \mathrm{d}r\, r \left[\phi(r,\theta,t_{n+1}) - \phi(r,\theta,t_n)\right] &= \frac{1}{2}\int_0^{\delta_r^2/4} \mathrm{d}s \left[\phi(\sqrt{s},\theta,t_{n+1}) - \phi(\sqrt{s},\theta,t_n)\right] \\ &= \frac{\delta_r^2}{8}\left[\phi(0,\theta,t_{n+1}) - \phi(0,\theta,t_n)\right].\end{aligned}$$

The right-hand side does not depend on θ because we evaluate the solution for $r = 0$ (origin), so we can easily perform the integration over θ, and finally express the integral in the left-hand side with the value of ϕ at the nodes:

$$\int_0^{\delta_r/2} \mathrm{d}r\, r \int_0^{2\pi} \mathrm{d}\theta\, \left[\phi(r,\theta,t_{n+1}) - \phi(r,\theta,t_n)\right] = 2\pi \times \frac{\delta_r^2}{8}\left(\phi_0^{n+1} - \phi_0^n\right) = \frac{\pi\delta_r^2}{4}\left(\phi_0^{n+1} - \phi_0^n\right).$$

We proceed similarly for the integral in the right-hand side. We first use a left rectangular method for the time integration time:

$$\int_{t_n}^{t_{n+1}} \mathrm{dt}\, \frac{\partial \phi}{\partial r}(\delta_r/2,\theta,t) = h\frac{\partial \phi}{\partial r}(\delta_r/2,\theta,t_n).$$

For the integration with respect to θ, we also apply the left rectangular method:

$$\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{2\pi} \mathrm{d}\theta \, \frac{\partial \phi}{\partial r}(\delta_r/2, \theta, t) = h \int_0^{2\pi} \mathrm{d}\theta \, \frac{\partial \phi}{\partial r}(\delta_r/2, \theta, t_n) = h\delta_\theta \sum_{k=0}^{M_\theta - 1} \frac{\partial \phi}{\partial r}(\delta_r/2, \theta_k, t_n).$$

We finally approximate the spatial derivative above with a CS representation, leading to

$$\int_{t_n}^{t_{n+1}} \mathrm{dt} \int_0^{2\pi} \mathrm{d}\theta \, \frac{\partial \phi}{\partial r}(\delta_r/2, \theta, t) = h\delta_\theta \sum_{k=0}^{M_\theta - 1} \frac{\phi_{1k}^n - \phi_0^n}{\delta_r}.$$

Combining the above results, we obtain that

$$\frac{\pi \delta_r^2}{4} \left(\phi_0^{n+1} - \phi_0^n\right) = \frac{Dh\delta_\theta}{2} \sum_{k=0}^{M_\theta - 1} \left(\phi_{1k}^n - \phi_0^n\right),$$

that we can rearrange as

$$\frac{\phi_0^{n+1} - \phi_0^n}{h} = D \times \frac{2\delta_\theta}{\pi \delta_r^2} \sum_{k=0}^{M_\theta - 1} \left(\phi_{1k}^n - \phi_0^n\right).$$

We recognize on the left-hand side the FD representation of the time derivative. Therefore the term in the right-hand side corresponds to the FD representation of the Laplacian for $j = 0$.

3. Building an integration scheme to solve Initial Value Problems

We have seen how to discretize space and time separately. For Boundary Value Problems, like the Poisson equation (3), the FD scheme is complete. But for Initial Value Problems, like the heat equation (1) or the wave equation (2), we need to combine the two to build an integration scheme. We illustrate few possible combinations in the case of the heat equation in two dimensions below. Of course, you can then easily adapt them to other PDEs.

a. Forward Time Centered Space scheme

Presentation of the scheme. The Forward Time Centered Space (FTCS) scheme is obtained by combining a FT representation of the time derivative with a CS representation of the Laplacian:

$$\left.\frac{\partial \phi}{\partial t}\right|_{jk}^n = D \, \Delta\phi|_{jk}^n$$

$$\frac{\phi_{jk}^{n+1} - \phi_{jk}^n}{h} + O(h) = D \left(\frac{\phi_{j+1,k}^n + \phi_{j-1,k}^n - 2\phi_{jk}^n}{\delta_x^2} + \frac{\phi_{j,k+1}^n + \phi_{j,k-1}^n - 2\phi_{jk}^n}{\delta_y^2} \right) + O(\delta_x^2, \delta_y^2).$$

We thus obtain the following recurrence relation.

FTCS scheme for the heat equation
The FTCS scheme for the 2D heat equation is given by the following recurrence relation:

$$\phi_{jk}^{n+1} = \phi_{jk}^n + \frac{Dh}{\delta_x^2} \left(\phi_{j+1,k}^n + \phi_{j-1,k}^n - 2\phi_{jk}^n\right) + \frac{Dh}{\delta_y^2} \left(\phi_{j,k+1}^n + \phi_{j,k-1}^n - 2\phi_{jk}^n\right). \tag{11}$$

This scheme is **explicit** because the equations giving ϕ_{jk}^{n+1} for all j and k are independent and only involve the values of the solution at time t_n.

The FTCS scheme is equivalent to the Forward Euler method for ODEs.

Solving the scheme. Because the scheme is explicit, it can be solved straightforwardly after boundary conditions have been provided (see the section about **Boundary Conditions**).

b. Backward Time Centered Space scheme

Presentation of the scheme. The Backward Time Centered Space (BTCS) scheme is obtained by combining a BT representation of the time derivative with a CS representation of the Laplacian:

$$\left.\frac{\partial \phi}{\partial t}\right|_{jk}^{n+1} = D\ \Delta\phi|_{jk}^{n+1}$$

$$\frac{\phi_{jk}^{n+1}-\phi_{jk}^{n}}{h} + O(h) = D\left(\frac{\phi_{j+1,k}^{n+1}+\phi_{j-1,k}^{n+1}-2\phi_{jk}^{n+1}}{\delta_x^2} + \frac{\phi_{j,k+1}^{n+1}+\phi_{j,k-1}^{n+1}-2\phi_{jk}^{n+1}}{\delta_y^2}\right) + O(\delta_x^2,\delta_y^2).$$

We thus obtain the following recurrence relation.

BTCS scheme for the heat equation
The BTCS scheme for the 2D heat equation is given by the following recurrence relation:

$$-\frac{Dh}{\delta_x^2}\left(\phi_{j+1,k}^{n+1}+\phi_{j-1,k}^{n+1}\right) - \frac{Dh}{\delta_y^2}\left(\phi_{j,k+1}^{n+1}+\phi_{j,k-1}^{n+1}\right) + \phi_{jk}^{n+1}\left(1+\frac{2Dh}{\delta_x^2}+\frac{2Dh}{\delta_y^2}\right) = \phi_{jk}^{n}. \tag{12}$$

This scheme is **implicit** because the equations giving ϕ_{jk}^{n+1} for all j and k are coupled.

The BTCS scheme is equivalent to the Backward Euler method for ODEs.

Solving the scheme. Because the scheme is implicit, it cannot be solved straightforwardly. This is because the equations giving ϕ_{jk}^{n+1} for all j and k are coupled. If the PDE is linear, you obtain a set of coupled linear equations that you can rewrite as a linear system. If the PDE is not linear, then you obtain a set of coupled non-linear equations. Solving these sets of coupled equations is discussed below.

c. Crank-Nicolson scheme

Presentation of the scheme. The Crank-Nicolson scheme is obtained by averaging the FT and BT schemes.

$$\begin{aligned}\frac{\phi_{jk}^{n+1}-\phi_{jk}^{n}}{h} + O(h) = &\frac{D}{2}\left(\frac{\phi_{j+1,k}^{n+1}+\phi_{j-1,k}^{n+1}-2\phi_{jk}^{n+1}}{\delta_x^2} + \frac{\phi_{j,k+1}^{n+1}+\phi_{j,k-1}^{n+1}-2\phi_{jk}^{n+1}}{\delta_y^2}\right)\\ &+ \frac{D}{2}\left(\frac{\phi_{j+1,k}^{n}+\phi_{j-1,k}^{n}-2\phi_{jk}^{n}}{\delta_x^2} + \frac{\phi_{j,k+1}^{n}+\phi_{j,k-1}^{n}-2\phi_{jk}^{n}}{\delta_y^2}\right) + O(\delta_x^2,\delta_y^2),\end{aligned}$$

leading to the following recurrence relation.

Crank-Nicolson scheme of the heat equation
The Crank-Nicolson scheme for the 2D heat equation is given by the following recurrence relation:

$$\begin{aligned}&-\frac{Dh}{2\delta_x^2}\left(\phi_{j+1,k}^{n+1}+\phi_{j-1,k}^{n+1}\right) - \frac{Dh}{2\delta_y^2}\left(\phi_{j,k+1}^{n+1}+\phi_{j,k-1}^{n+1}\right) + \phi_{jk}^{n+1}\left(1+\frac{Dh}{\delta_x^2}+\frac{Dh}{\delta_y^2}\right)\\ &= \phi_{jk}^{n}\left(1-\frac{Dh}{\delta_x^2}-\frac{Dh}{\delta_y^2}\right) + \frac{Dh}{2\delta_x^2}\left(\phi_{j+1,k}^{n}+\phi_{j-1,k}^{n}\right) + \frac{Dh}{2\delta_y^2}\left(\phi_{j,k+1}^{n}+\phi_{j,k-1}^{n}\right).\end{aligned} \tag{13}$$

This scheme is **implicit** because the equations giving ϕ_{jk}^{n+1} for all j and k are coupled.

The Crank-Nicolson scheme is equivalent to the Trapezoidal method for ODEs.

Solving the scheme. Because the scheme is implicit, it cannot be solved straightforwardly, like the BTCS scheme.

II. Implementing Boundary Conditions

Before discussing how to solve the various schemes introduced above, we need to complement the PDE with **Boundary Conditions** (BCs). In these notes, we discuss the most common ones: **periodic** BCs, **Dirichlet** BCs and **Neumann** BCs.

1. Periodic boundary conditions

a. Definition

We start by defining periodic boundary conditions.

Definition of periodic boundary conditions
Periodic boundary conditions in the x-direction mean that for all y, the point of coordinates $(x_\mathrm{i},\, y)$ is the same as the point of coordinates $(x_\mathrm{f},\, y)$ (you wrap the rectangle around itself). For the solution, this amounts to say that

$$\forall n \in [\![0,\, N]\!]\ \forall k \in [\![0,\, M_y]\!]\ \phi^n_{0k} = \phi^n_{M_x,k}. \tag{14}$$

Periodic boundary conditions in the y-direction mean that for all x, the point of coordinates $(x,\, y_\mathrm{i})$ is the same as the point of coordinates $(x,\, y_\mathrm{f})$ (you wrap the rectangle around itself). For the solution, this amounts to say that

$$\forall n \in [\![0,\, N]\!]\ \forall j \in [\![0,\, M_x]\!]\ \phi^n_{j0} = \phi^n_{j,M_y}. \tag{15}$$

Periodic BCs emerge naturally in the case of polar coordinates (the solutions are 2π-periodic functions of θ). Otherwise, they may be used to avoid finite-size effects and to mimic an infinite system.

Periodic boundary conditions are easy to implement. We illustrate this for the one-dimensional heat equation

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = D \dfrac{\partial^2 \phi}{\partial x^2}, \\ \phi(0,t) = \phi(1,t), \\ \phi(x,0) = \phi_\mathrm{i}(x), \end{cases}$$

first for the FTCS scheme, and then for the BTCS scheme.

b. FTCS scheme for the 1D heat equation with periodic BCs

The recurrence relation for the FTCS scheme is given by Eq. (11):

$$\phi^{n+1}_j = \phi^n_j + \frac{Dh}{\delta_x^2}\left(\phi^n_{j+1} + \phi^n_{j-1} - 2\phi^n_j\right).$$

Because of the periodic boundary conditions, we can restrict j to $[\![0,\, M_x - 1]\!]$. If we write explicitely the above recurrence relation, we get

$$\begin{cases} \phi^{n+1}_0 = \phi^n_0 + \dfrac{Dh}{\delta_x^2}\left(\phi^n_1 + \phi^n_{M_x-1} - 2\phi^n_0\right), \\ \phi^{n+1}_1 = \phi^n_1 + \dfrac{Dh}{\delta_x^2}\left(\phi^n_2 + \phi^n_0 - 2\phi^n_1\right), \\ \quad\vdots \\ \phi^{n+1}_{M_x-1} = \phi^n_{M_x-1} + \dfrac{Dh}{\delta_x^2}\left(\phi^n_0 + \phi^n_{M_x-2} - 2\phi^n_{M_x-1}\right). \end{cases} \tag{16}$$

c. BTCS scheme for the 1D heat equation with periodic BCs

We can repeat the procedure for the BTCS scheme. The recurrence relation is given by Eq. (12):

$$-\frac{Dh}{\delta_x^2}\left(\phi_{j+1}^{n+1}+\phi_{j-1}^{n+1}\right)+\phi_j^{n+1}\left(1+\frac{2Dh}{\delta_x^2}\right)=\phi_j^n.$$

Because of the periodic boundary conditions, we can restrict j to $[\![0,\, M_x-1]\!]$. If we write explicitely the above recurrence relation, we get the following linear system

$$\begin{pmatrix} 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 & 0 & \dots & 0 & -Dh/\delta_x^2 \\ -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 & 0 & \dots & 0 \\ & & \dots & \dots & & \\ 0 & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 \\ -Dh/\delta_x^2 & 0 & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 \end{pmatrix} \begin{pmatrix} \phi_0^{n+1} \\ \phi_1^{n+1} \\ \vdots \\ \phi_{M_x-2}^{n+1} \\ \phi_{M_x-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \phi_0^n \\ \phi_1^n \\ \vdots \\ \phi_{M_x-2}^n \\ \phi_{M_x-1}^n \end{pmatrix}. \tag{17}$$

2. Dirichlet boundary conditions

a. Definition

We start by defining Dirichlet boundary conditions.

Definition of Dirichlet boundary conditions
Dirichlet boundary conditions correspond to imposing the value of the solution on one boundary.

For example, for a boundary condition of the form $\phi(x_{\text{i}}, y, t) = g(y, t)$, this amounts to say that

$$\forall n \in [\![0,\, N]\!]\ \forall k \in [\![0,\, M_y]\!]\ \phi_{0k}^n = g_k^n, \tag{18}$$

where $g_k^n = g(y_k, t_n)$.

Dirichlet BCs are used when you want to impose the value of the solution on one boundary. Several physical examples are given below.

Examples of natural Dirichlet BCs include:

- fixing the value of the electrostatic potential on a conductor,
- fixing the amplitude of motion of the extremity of a string,
- fixing the temperature at the interface between two media.

Dirichlet boundary conditions are easy to implement. We illustrate this again for the one-dimensional heat equation (physically, this corresponds to imposing the temperature on the two boundaries)

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = D\dfrac{\partial^2 \phi}{\partial x^2}, \\ \phi(0,t) = a(t), \\ \phi(1,t) = b(t), \\ \phi(x,0) = \phi_{\text{i}}(x), \end{cases}$$

first for the FTCS scheme, and then for the BTCS scheme.

b. FTCS scheme for the 1D heat equation with Dirichlet BCs

The recurrence relation for the FTCS scheme is given by Eq. (11):

$$\phi_j^{n+1} = \phi_j^n + \frac{Dh}{\delta_x^2}\left(\phi_{j+1}^n + \phi_{j-1}^n - 2\phi_j^n\right).$$

Because of the Dirichlet boundary conditions, we can restrict j to $[\![1,\, M_x - 1]\!]$ because $\phi_0^n = a^n$ and $\phi_{M_x}^n = b^n$ [with $a^n = a(t_n)$ and $b^n = b(t_n)$]. If we write explicitely the above recurrence relation, we get

$$\begin{cases} \phi_1^{n+1} = \phi_1^n + \dfrac{Dh}{\delta_x^2}\left(\phi_2^n + a^n - 2\phi_1^n\right), \\ \phi_2^{n+1} = \phi_2^n + \dfrac{Dh}{\delta_x^2}\left(\phi_3^n + \phi_1^n - 2\phi_2^n\right), \\ \qquad \vdots \\ \phi_{M_x-1}^{n+1} = \phi_{M_x-1}^n + \dfrac{Dh}{\delta_x^2}\left(b^n + \phi_{M_x-2}^n - 2\phi_{M_x-1}^n\right). \end{cases} \tag{19}$$

c. BTCS scheme for the 1D heat equation with Dirichlet BCs

We can repeat the procedure for the BTCS scheme. The recurrence relation is given by Eq. (12):

$$-\frac{Dh}{\delta_x^2}\left(\phi_{j+1}^{n+1} + \phi_{j-1}^{n+1}\right) + \phi_j^{n+1}\left(1 + \frac{2Dh}{\delta_x^2}\right) = \phi_j^n.$$

Because of the Dirichlet boundary conditions, we can restrict j to $[\![1,\, M_x - 1]\!]$. If we write explicitely the above recurrence relation, we get the following linear system

$$\begin{pmatrix} 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 & 0 & \dots & 0 & 0 \\ -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 & 0 & \dots & 0 \\ & & \dots & \dots & & \\ 0 & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 \\ 0 & 0 & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 \end{pmatrix} \begin{pmatrix} \phi_1^{n+1} \\ \phi_2^{n+1} \\ \vdots \\ \phi_{M_x-2}^{n+1} \\ \phi_{M_x-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \phi_1^n + Dha^{n+1}/\delta_x^2 \\ \phi_2^n \\ \vdots \\ \phi_{M_x-2}^n \\ \phi_{M_x-1}^n + Dhb^{n+1}/\delta_x^2 \end{pmatrix}. \tag{20}$$

3. Neumann boundary conditions

a. Definition

We start by defining Neumann boundary conditions.

> **Definition of Neumann boundary conditions**
> Neumann boundary conditions correspond to imposing the value of the normal derivative of the solution on one boundary. For example, a Neumann boundary condition at $x = x_{\mathrm{i}}$ is of the form
>
> $$\frac{\partial \phi}{\partial x}(x_{\mathrm{i}}, y, t) = h(y, t). \tag{21}$$

Similarly, a Neumann boundary condition at $y = y_{\mathrm{f}}$ is of the form

$$\frac{\partial \phi}{\partial y}(x, y_{\mathrm{f}}, t) = h(x, t). \tag{22}$$

Neumann BCs are used when you want to impose the value of the normal derivative, *i.e.*, the flux, on one boundary. Several physical examples are given below.

Examples of natural Neumann BCs include:

- fixing the value of the electric field (gradient of the electrostatic potential) at the interface between two media,
- fixing the heat flux at the interface between two media.

Neumann boundary conditions require some manipulations that we explain now.

Implementation of Neumann boundary conditions
To impose the boundary condition given by Eq. (21), you have to define a **ghost cell** (see Fig. 2) labelled $j = -1$. Then by using a CS representation, you get

$$\frac{\phi^n_{1,k} - \phi^n_{-1,k}}{2\delta_x} = h^n_k,$$

with $h^n_k = h(y_k, t_n)$. The values $\phi^n_{-1,k}$ will also appear in the recurrence relations of the scheme in order to close the system of equations.

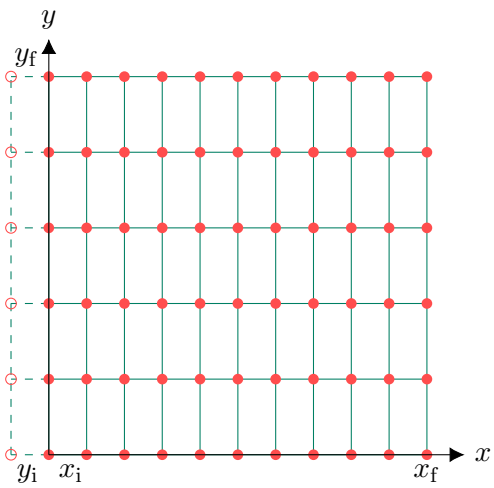

Figure 2: **Illustration of the ghost cell construction to define Neumann Boundary Conditions.** We use the same space discretization as in Fig. 1. If we want to impose a Neumann boundary condition at $x = x_{\mathrm{i}}$, we define ghost nodes of abscissa $x_{\mathrm{i}} - \delta_x$ (empty disks connected to real nodes by dashed lines).

We illustrate the implementation of Neumann BCs again for the one-dimensional heat equation (physically, this corresponds to imposing the heat flux on the two boundaries)

$$\begin{cases} \dfrac{\partial \phi}{\partial t} = D \dfrac{\partial^2 \phi}{\partial x^2}, \\ \dfrac{\partial \phi}{\partial x}(0, t) = q(t), \\ \phi(1, t) = b(t), \\ \phi(x, 0) = \phi_{\mathrm{i}}(x), \end{cases}$$

first for the FTCS scheme, and then for the BTCS scheme. Note that we have imposed one Neumann BC on the left, but a Dirichlet BC on the right (simpler case).

b. FTCS scheme for the 1D heat equation with Neumann BCs

The recurrence relation for the FTCS scheme is given by Eq. (11):

$$\phi_j^{n+1} = \phi_j^n + \frac{Dh}{\delta_x^2}\left(\phi_{j+1}^n + \phi_{j-1}^n - 2\phi_j^n\right).$$

Because of the Dirichlet boundary condition on the right extremity, we can restrict j to $[\![0,\, M_x - 1]\!]$ because $\phi_{M_x}^n = b^n$. We now write explicitely the above recurrence relation:

$$\begin{cases} \phi_0^{n+1} = \phi_0^n + \dfrac{Dh}{\delta_x^2}\left(\phi_1^n + \phi_{-1}^n - 2\phi_0^n\right), \\ \phi_1^{n+1} = \phi_1^n + \dfrac{Dh}{\delta_x^2}\left(\phi_2^n + \phi_0 - 2\phi_1^n\right), \\ \qquad \vdots \\ \phi_{M_x-1}^{n+1} = \phi_{M_x-1}^n + \dfrac{Dh}{\delta_x^2}\left(b + \phi_{M_x-2}^n - 2\phi_{M_x-1}^n\right). \end{cases}$$

We see that ϕ_{-1}^n (the value of the function at the ghost node) appears in these equations. We now combine the first equation of this system, with the Neumann boundary condition

$$\frac{\phi_1^n - \phi_{-1}^n}{2\delta_x} = q^n \Longrightarrow \phi_{-1}^n = \phi_1^n - 2\delta_x q^n,$$

where $q^n = q(t_n)$, to get

$$\phi_0^{n+1} = \phi_0^n + \frac{2Dh}{\delta_x^2}\left(\phi_1^n - \delta_x q^n - \phi_0^n\right).$$

Eventually, the recurrence relation can be written

$$\begin{cases} \phi_0^{n+1} = \phi_0^n + \dfrac{2Dh}{\delta_x^2}\left(\phi_1^n - \delta_x q^n - \phi_0^n\right), \\ \phi_1^{n+1} = \phi_1^n + \dfrac{Dh}{\delta_x^2}\left(\phi_2^n + \phi_0^n - 2\phi_1^n\right), \\ \qquad \vdots \\ \phi_{M_x-1}^{n+1} = \phi_{M_x-1}^n + \dfrac{Dh}{\delta_x^2}\left(b^n + \phi_{M_x-2}^n - 2\phi_{M_x-1}^n\right). \end{cases} \tag{23}$$

c. BTCS scheme for the 1D heat equation with Neumann BCs

We can repeat the procedure for the BTCS scheme. The recurrence relation is given by Eq. (12):

$$-\frac{Dh}{\delta_x^2}\left(\phi_{j+1}^{n+1} + \phi_{j-1}^{n+1}\right) + \phi_j^{n+1}\left(1 + \frac{2Dh}{\delta_x^2}\right) = \phi_j^n.$$

Because of the Dirichlet boundary condition on the right extremity, we can restrict j to $[\![0,\, M_x - 1]\!]$. Similarly as before, we have to combine the recurrence relation for $j = 0$

$$-\frac{Dh}{\delta_x^2}\left(\phi_1^{n+1} + \phi_{-1}^{n+1}\right) + \phi_0^{n+1}\left(1 + \frac{2Dh}{\delta_x^2}\right) = \phi_0^n.$$

with the Neumann BC (this time at instant t_{n+1})

$$\frac{\phi_1^{n+1} - \phi_{-1}^{n+1}}{2\delta_x} = q^{n+1} \Longrightarrow \phi_{-1}^{n+1} = \phi_1^{n+1} - 2\delta_x q^{n+1},$$

to get rid of ϕ_{-1}^{n+1} (which is unknown). We thus get

$$-\frac{2Dh}{\delta_x^2}\phi_1^{n+1} + \phi_0^{n+1}\left(1+\frac{2Dh}{\delta_x^2}\right) = \phi_0^n - \frac{2Dh}{\delta_x}q^{n+1}.$$

Therefore, we can write explicitely the recurrence relations as the following linear system

$$\begin{pmatrix} 1+2Dh/\delta_x^2 & -2Dh/\delta_x^2 & 0 & \dots & \dots & 0 \\ -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 & 0 & \dots & 0 \\ & & \dots & \dots & & \\ 0 & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 & -Dh/\delta_x^2 \\ 0 & \dots & \dots & 0 & -Dh/\delta_x^2 & 1+2Dh/\delta_x^2 \end{pmatrix} \begin{pmatrix} \phi_0^{n+1} \\ \phi_1^{n+1} \\ \vdots \\ \phi_{M_x-2}^{n+1} \\ \phi_{M_x-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \phi_0^n - 2Dhq^{n+1}/\delta_x \\ \phi_1^n \\ \vdots \\ \phi_{M_x-2}^n \\ \phi_{M_x-1}^n + Dhb^{n+1}/\delta_x^2 \end{pmatrix}. \tag{24}$$

4. Other boundary conditions

There are other kinds of boundary conditions, which involve the value of the function and of its spatial derivatives on the frontiers of the domain. They are referred to as mixed boundary conditions. They will not be discussed in these lecture notes.

5. Conclusion: the role of boundary conditions

The above discussion should have made clear that **a Finite-Difference scheme must always be complemented with Boundary Conditions in order to be solvable**. Otherwise, the problem has more unknowns than equations.

III. Solving Finite-Difference equations for Initial Value Problems and Boundary Value Problems

1. Method

We now discuss methods to solve IVPs and BVPs, like the heat equation (1) or the wave equation (2).

Solving explicit schemes. Examples of explicit schemes are given by Eqs. (16), (19) and (23). These equations are not coupled for ϕ_{lk}^{n+1} and can thus be solved very straightforwardly.

Solving implicit schemes. Examples of implicit schemes are given by Eqs. (17), (20) and (24). These schemes amount to solving coupled equations for ϕ_{lk}^{n+1} which are linear as soon as the PDE is linear. As a conclusion, we need to solve linear systems at each time step.

> The matrix involved in the linear systems given by Eqs. (17), (20) and (24) has an important property, it is sparse, meaning that it has a lot of zeros on each line. Specific and efficient methods to solve linear systems exist for sparse matrices, you can look at the documentation page of `scipy.sparse`. For example, the systems given by Eqs. (17), (20) and (24) are of size $O(M_x)$. Gaussian elimination costs $O(M_x^3)$ operations, while specific algorithms for sparse matrices only cost $O(M_x)$ operations.

2. Choice of the discretization steps

The choice of the discretization steps h, δ_x and δ_y remains to be discussed. Of course, we expect the integration scheme to provide a better approximation of the solution when h, δ_x, $\delta_y \to 0$ (if the scheme is convergent).

However, the smaller the discretization steps are, the larger the computation time. Therefore, we ask how large can we make the discretization steps to still get an acceptable estimate of the solution?

The major constraint is the **stability** of the scheme. Said differently, a numerical integrator of PDEs can become unstable if the discretization steps are taken too large. The analysis of the stability of PDE integrators is complex because of the boundary conditions which affect the form of the recurrence relations [see for instance Eqs. (17), (20) and (24)]. In the following, we propose a method to assess the stability of an integrator scheme for **linear PDEs** without taking into account the boundary conditions. This method, called the **von Neumann stability criterion**, however cannot provide a sufficient criterion for the scheme to be stable, but **only a necessary condition**.

von Neumann stability criterion
If a FD scheme solving a **linear PDE** is **stable** then injecting the von Neumann ansatz

$$\phi_{jk}^n = \varphi^n e^{\mathrm{i}(\xi j + \eta k)} \tag{25}$$

(with ξ, $\eta \in \mathbb{R}$ and $\varphi^n \in \mathbb{C}$) into the recurrence relation must lead to a **non-diverging amplitude**:

$$\left|\varphi^{n+1}\right| \leq \left|\varphi^n\right| \tag{26}$$

for all values of ξ and η. The ratio $\rho = \varphi^{n+1}/\varphi^n$ is called the **symbol** or the **amplification factor** of the scheme. The latter is stable if $|\rho| \leq 1$.

Again, be careful that the reciprocal is **not** true. In other words, the von Neumann stability criterion can be satisfied but the scheme can be unstable because of the boundary conditions.

We illustrate the method in an example below for the FTCS scheme to solve the 2D heat equation (try it by yourself first!).

We inject the von Neumann ansatz (25) into the FTCS scheme for the 2D heat equation [see Eq. (11)]:

$$\begin{aligned}\varphi^{n+1} e^{\mathrm{i}[\xi(j+1)+\eta k]} = \varphi^n e^{\mathrm{i}(\xi j+\eta k)} &+ \frac{Dh}{\delta_x^2}\left\{\varphi^n e^{\mathrm{i}[\xi(j+1)+\eta k]} + \varphi^n e^{\mathrm{i}[\xi(j-1)+\eta k]} - 2\varphi^n e^{\mathrm{i}(\xi j+\eta k)}\right\} \\ &+ \frac{Dh}{\delta_y^2}\left\{\varphi^n e^{\mathrm{i}[\xi j+\eta(k+1)]} + \varphi^n e^{\mathrm{i}[\xi j+\eta(k-1)]} - 2\varphi^n e^{\mathrm{i}(\xi j+\eta k)}\right\}.\end{aligned}$$

Simplifying by $e^{\mathrm{i}(\xi j+\eta k)}$, we get

$$\begin{aligned}\varphi^{n+1} &= \varphi^n\left[1 + \frac{Dh}{\delta_x^2}\left(e^{\mathrm{i}\xi} + e^{-\mathrm{i}\xi} - 2\right) + \frac{Dh}{\delta_y^2}\left(e^{\mathrm{i}\eta} + e^{-\mathrm{i}\eta} - 2\right)\right] \\ \varphi^{n+1} &= \varphi^n\left[1 + \frac{2Dh}{\delta_x^2}\left(\cos\xi - 1\right) + \frac{2Dh}{\delta_y^2}\left(\cos\eta - 1\right)\right] \\ \varphi^{n+1} &= \varphi^n\left[1 - \frac{4Dh}{\delta_x^2}\sin^2\left(\frac{\xi}{2}\right) - \frac{4Dh}{\delta_y^2}\sin^2\left(\frac{\eta}{2}\right)\right].\end{aligned}$$

We thus identify the symbol

$$\rho = 1 - \frac{4Dh}{\delta_x^2}\sin^2\left(\frac{\xi}{2}\right) - \frac{4Dh}{\delta_y^2}\sin^2\left(\frac{\eta}{2}\right)$$

As a result, for the scheme to be stable, we need $-1 \leq \rho \leq 1$. The inequality $\rho \leq 1$ is verified. The other inequality is verified if

$$\frac{2Dh}{\delta_x^2}\sin^2\left(\frac{\xi}{2}\right) + \frac{2Dh}{\delta_y^2}\sin^2\left(\frac{\eta}{2}\right) \leq 1.$$

This inequality has to be true for all values of ξ and η. The maximum value of the left-hand side is obtained for

$\xi = \eta = \pi$ and we thus need that

$$2Dh\left(\frac{1}{\delta_x^2}+\frac{1}{\delta_y^2}\right) \leq 1 \Longrightarrow h \leq \frac{\delta_x^2\delta_y^2}{2D(\delta_x^2+\delta_y^2)}.$$

We have not discussed the convergence properties of the integrator, namely, whether the numerical solution approaches the exact solution when the discretization steps go to 0. There exists a theorem, called the Lax equivalence theorem, which tells you that if a scheme is consistent and stable, then it is convergent. Here all the schemes presented are generalizations of schemes introduced for ODEs. They are thus all consistent (the recurrence relation approximates the PDE well). As a consequence, as long as they are stable, they converge.

IV. Solving Finite-Difference equations for Boundary Value Problems

In this section, we list several methods to solve stationary BVPs, that we illustrate in the particular case of the Poisson equation (3). For more details about all these methods, you can read J. W. Thomas, *Numerical Partial Differential Equations. Conservation Laws and Elliptic Equations*.

1. Finite-Difference representation of the Poisson equation

We start by building the FD representation of Eq. (3).

Finite-Difference representation of the Poisson equation
The FD representation of the 2D Poisson equation is

$$\frac{\phi_{j+1,k}+\phi_{j-1,k}-2\phi_{jk}}{\delta_x^2}+\frac{\phi_{j,k+1}+\phi_{j,k-1}-2\phi_{jk}}{\delta_y^2} = f_{jk} \tag{27}$$

using a second-order CS scheme, where $f_{jk} = f(x_j, y_k)$.

Do not forget that this equation must be complemented with **boundary conditions**. We have already seen how to implement boundary conditions, and here we only focus on Dirichlet boundary conditions:

$$\begin{cases} \Delta\phi = f, \\ \phi(x_\mathrm{i}, y) = a(y), \\ \phi(x_\mathrm{f}, y) = b(y), \\ \phi(x, y_\mathrm{i}) = c(x), \\ \phi(x, y_\mathrm{f}) = d(x). \end{cases}$$

Because of the Dirichlet boundary conditions, we can restrict ourselves to $j \in [\![1, M_x - 1]\!]$ and $k \in [\![1, M_y - 1]\!]$. We are thus left with $M_x \times M_y$ coupled linear equations that we can write as a system. For that, we need to rearrange all ϕ_{jk}'s into a column vector Φ. We decide to go line by line (although other rearrangements are possible), look at Fig. 1 to find your way with the indices:

$$\Phi = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \Phi_3 \\ \vdots \\ \Phi_{M_y-3} \\ \Phi_{M_y-2} \\ \Phi_{M_y-1} \end{pmatrix}, \tag{28}$$

where Φ_k for $k \in [\![1,\, M_y - 1]\!]$ are column vectors of size M_x:

$$\Phi_k = \begin{pmatrix} \phi_{1k} \\ \phi_{2k} \\ \vdots \\ \phi_{M_x-2,k} \\ \phi_{M_x-1,k} \end{pmatrix}.$$

The column vector Φ is of size $M_x M_y$, and Eq. (28) is a block representation. If we multiply Eq. (27) by δ_x, the recurrence relations become equivalent to the following system

$$R\Phi = S, \tag{29}$$

with R a square matrix of size $M_x M_y$ which is, in block representation,

$$R = \begin{pmatrix} T & J & 0 & \dots & \dots & \dots & 0 \\ J & T & J & 0 & \dots & \dots & 0 \\ 0 & J & T & J & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & J & T & J & 0 \\ 0 & \dots & \dots & 0 & J & T & J \\ 0 & \dots & \dots & \dots & 0 & J & T \end{pmatrix},$$

where T and J are squared matrices of size M_x, with $J = (\delta_x^2/\delta_y^2)I$ (I is the identity matrix) and

$$T = \begin{pmatrix} -2(1+\delta_x^2/\delta_y^2) & 1 & 0 & \dots & \dots & 0 \\ 1 & -2(1+\delta_x^2/\delta_y^2) & 1 & 0 & \dots & 0 \\ & & \dots & \dots & & \\ 0 & \dots & 0 & 1 & -2(1+\delta_x^2/\delta_y^2) & 1 \\ 0 & \dots & \dots & 0 & 1 & -2(1+\delta_x^2/\delta_y^2) \end{pmatrix}.$$

The column vector S contains both the source term f and the Dirichlet BCs and reads, in block representation,

$$S = \delta_x^2 \begin{pmatrix} F_1 \\ F_2 \\ F_3 \\ \vdots \\ F_{M_y-3} \\ F_{M_y-2} \\ F_{M_y-1} \end{pmatrix} - \frac{\delta_x^2}{\delta_y^2} \begin{pmatrix} C \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ D \end{pmatrix} - \begin{pmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ L_{M_y-3} \\ L_{M_y-2} \\ L_{M_y-1} \end{pmatrix}, \tag{30}$$

where F_k and L_k for $k \in [\![1,\, M_y - 1]\!]$ are column vectors of size M_x:

$$F_k = \begin{pmatrix} f_{1k} \\ f_{2k} \\ \vdots \\ f_{M_x-2,k} \\ f_{M_x-1,k} \end{pmatrix}, \qquad L_k = \begin{pmatrix} a_k \\ 0 \\ \vdots \\ 0 \\ b_k \end{pmatrix}$$

with $a_k = a(y_k)$ and $b_k = b(y_k)$. Similarly, C and D are column vectors of siz M_x, with

$$C = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{M_x-2} \\ c_{M_x-1} \end{pmatrix}, \qquad D = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_{M_x-2} \\ d_{M_x-1} \end{pmatrix},$$

with $c_j = c(x_j)$ and $d_j = d(x_j)$. The second vector in Eq. (30) stands for the top and bottom boundary conditions, the last vector for the left and right boundary conditions.

Conclusion. We are left with a linear system given by Eq. (29) to solve. In the following section, we propose several ways of solving this system.

2. Brute-force solving

Just as we did for IVPs and BVPs, we can directly solve the linear system given by Eq. (29). However, these systems can be very large, and the computation can be very slow. We thus propose iterative techniques which can be faster in the following paragraph.

> We can quantify how slow such a brute-force solving may be. The system (29) is of size $M_x M_y$ and solving it with Gaussian elimination requires $O(M_x^3 M_y^3)$ operations. Note though that the matrix is sparse and specific algorithms only cost $O(M_x^3 M_y)$ operations. For $M_x = M_y$, this still represents $O(M_x^4)$ operations!

3. Iterative solving

a. The Jacobi method

The first iterative method, the **Jacobi method**, is directly inspired from our previous study of IVPs and BVPs. Consider the heat equation

$$\frac{\partial \phi}{\partial t} = \Delta \phi - f,$$

in the presence of a source term f. We know that in the stationary state (*i.e.*, when $t \to +\infty$), $\phi(x, y, t)$ converges to the solution $\Delta \phi = f$, which is precisely the problem we have to solve. Asa consequence, the idea is to solve the IVP and BVP given above up to a time when the solution does not change much. For instance, if you use a FTCS scheme, the recurrence relation is

$$\phi_{jk}^{n+1} = \phi_{jk}^n + \frac{h}{\delta_x^2}\left(\phi_{j+1,k}^n + \phi_{j-1,k}^n - 2\phi_{jk}^n\right) + \frac{h}{\delta_y^2}\left(\phi_{j,k+1}^n + \phi_{j,k-1}^n - 2\phi_{jk}^n\right) - h f_{jk}.$$

How to choose the time step h, which is here a calculation trick? We have seen above that the FTCS scheme can be stable only if

$$h \leq \frac{\delta_x^2 \delta_y^2}{2(\delta_x^2 + \delta_y^2)}.$$

To fasten the algorithm, we have to take h the largest while respecting the above condition, and we thus propose to iterate with $h = \delta_x^2 \delta_y^2 / [2(\delta_x^2 + \delta_y^2)]$. Then the iteration scheme becomes:

$$\begin{aligned}\phi_{jk}^{n+1} &= \phi_{jk}^n + \frac{\delta_y^2}{2(\delta_x^2 + \delta_y^2)}\left(\phi_{j+1,k}^n + \phi_{j-1,k}^n - 2\phi_{jk}^n\right) + \frac{\delta_x^2}{2(\delta_x^2 + \delta_y^2)}\left(\phi_{j,k+1}^n + \phi_{j,k-1}^n - 2\phi_{jk}^n\right) - \frac{\delta_x^2 \delta_y^2}{2(\delta_x^2 + \delta_y^2)} f_{jk} \\ &= \frac{1}{2(\delta_x^2 + \delta_y^2)}\left[\delta_y^2(\phi_{j+1,k}^n + \phi_{j-1,k}^n) + \delta_x^2(\phi_{j,k+1}^n + \phi_{j,k-1}^n) - \delta_x^2 \delta_y^2 f_{jk}\right].\end{aligned}$$

Jacobi method for the Poisson equation

To solve the 2D Poisson equation (3), the **Jacobi method** consists in iterating the recurrence scheme

$$\phi_{jk}^{n+1} = \frac{1}{2(\delta_x^2 + \delta_y^2)}\left[\delta_y^2(\phi_{j+1,k}^n + \phi_{j-1,k}^n) + \delta_x^2(\phi_{j,k+1}^n + \phi_{j,k-1}^n) - \delta_x^2 \delta_y^2 f_{jk}\right] \tag{31}$$

for $n \in [\![0,\, N]\!]$ from an initial guess.

The value of N is set so that the relative change of the solution between two iterations at steps n and $n+1$

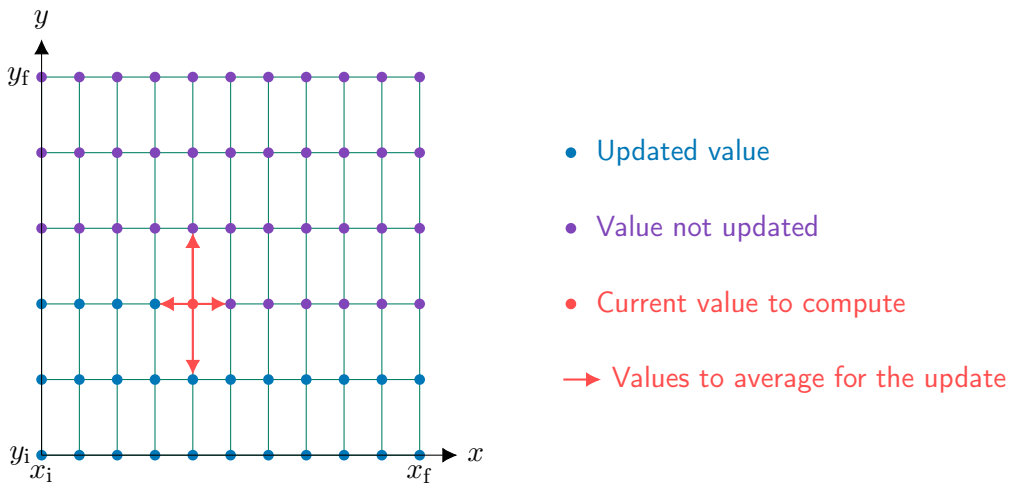

Figure 3: **Illustration of the Gauss-Seidel method to solve the Poisson equation.** The current node appears in pink, the already updated nodes in blue and the nodes which still need to be updated in purple. The arrows mark the data points to average to make the update.

is smaller than a given threshold $\epsilon \ll 1$:

$$\left\|\Phi^n - \Phi^{n+1}\right\| < \epsilon \left\|\Phi^n\right\|, \tag{32}$$

with Φ^n the column vector given by Eq. (28).

Said differently, the Jacobi method tells you that the updated values of ϕ_{jk} are obtained as the average of the old neighboring values, plus an extra term coming from the source term.

The Jacobi method is **explicit** and can be implemented straightforwardly.

b. The Gauss-Seidel method

The **Gauss-Seidel method** is very close to the Jacobi method, but instead of updating ϕ_{jk} with the average of the old neighboring values, you average over the neighboring nodes the new values if they are already known, or the old values otherwise, see Fig. 3. Of course, you still have to add the extra term coming from the source f.

Gauss-Seidel method for the Poisson equation
To solve the 2D Poisson equation (3), the **Gauss-Seidel method** consists in iterating the recurrence scheme

$$\phi_{jk}^{n+1} = \frac{1}{2(\delta_x^2 + \delta_y^2)} \left[\delta_y^2(\phi_{j+1,k}^{n/n+1} + \phi_{j-1,k}^{n/n+1}) + \delta_x^2(\phi_{j,k+1}^{n/n+1} + \phi_{j,k-1}^{n/n+1}) - \delta_x^2\delta_y^2 f_{jk}\right] \tag{33}$$

for $n \in [\![0,\, N]\!]$ from an initial guess. The superscript $n/n+1$ means that you should use the value at step $n+1$ if it has already been computed (lower line and left column), or the value at step n otherwise (upper line and right column), see Fig. 3.

The value of N is set so that the relative change of the solution between two iterations at steps n and $n+1$ is smaller than a given threshold $\epsilon \ll 1$: $\left\|\Phi^n - \Phi^{n+1}\right\| < \epsilon \left\|\Phi^n\right\|$, with Φ^n the column vector given by Eq. (28).

Said differently, the Gauss-Seidel method tells you that the updated values of ϕ_{jk} are obtained as the average over the neighboring nodes of the new values if they are already known, or the old values otherwise, plus an extra term coming from the source term.

The Gauss-Seidel method is **explicit** and can be implemented straightforwardly.

c. Successive overrelaxation method

The **successive overrelaxation method** is very close to the Gauss-Seidel method. The updated value of ϕ_{jk} is now obtained as a weighted average of the values at the step before and of the prediction coming from the Gauss-Seidel method.

Successive overrelaxation method for the Poisson equation
To solve the 2D Poisson equation (3), the **successive overrelaxation method** consists in iterating the recurrence scheme

$$\begin{cases} \hat{\phi}_{jk} = \dfrac{1}{2(\delta_x^2+\delta_y^2)}\left[\delta_y^2(\phi_{j+1,k}^{n/n+1}+\phi_{j-1,k}^{n/n+1})+\delta_x^2(\phi_{j,k+1}^{n/n+1}+\phi_{j,k-1}^{n/n+1})-\delta_x^2\delta_y^2 f_{jk}\right] \\ \phi_{jk}^{n+1} = (1-\omega)\phi_{jk}^n + \omega\hat{\phi}_{jk} \end{cases} \tag{34}$$

for $n \in [\![0,\, N]\!]$ from an initial guess. The superscript $n/n+1$ means that you should use the value at step $n+1$ if it has already been computed (lower line and left column), or the value at step n otherwise (upper line and right column), see Fig. 3. The weight ω must verify $0 < \omega < 2$ in order for the method to be convergent.

The value of N is set so that the relative change of the solution between two iterations at steps n and $n+1$ is smaller than a given threshold $\epsilon \ll 1$: $\left\|\Phi^n - \Phi^{n+1}\right\| < \epsilon \left\|\Phi^n\right\|$, with Φ^n the column vector given by Eq. (28).

How to choose the weight ω? First, you should take it different from 1, otherwise you recover the Gauss-Seidel method. In practice, the larger the faster: therefore **you should take** $\omega \lesssim 2$.

V. Beyond Finite-Difference methods

In these lectures notes, we have only discussed Finite-Difference methods. Other methods to solve PDEs exist, among them the **Finite-Element methods** and the **Spectral methods**. Both methods consist in decomposing the solution of the PDE on a basis of functions. They can be more robust when you have to solve complex non-linear PDEs. We will not discuss them in these notes, but you should know that there are Python librairies implementing these methods. For more details, you can consult the documentation pages of FEniCS (Finite-Element methods) or Dedalus (Spectral methods).