Chapter 4

# Introduction to numerical integration of PDEs

**Disclaimer: the content of this chapter will not be covered in lectures, it is non-examinable!**

Until now in this module, we have looked at the problem of finding analytical solutions to PDEs. We studied a variety of resolution methods for a quite large number of different PDEs. Now a common thread throughout these Chapters was that the applicability of these methods was limited to rather canonical problems, i.e. sufficiently nice equations in simple domains. Even when we were able to devise an analytical solution, this sometimes took the form on an infinite series which is unpractical. Furthermore, nonlinear equations or equations in complicated domains cannot, in general, be solved analytically. Unfortunately, most real-world problems are nonlinear and/or need to be solved in complex geometries. So is this the end of the road?

Not quite! In this Chapter, we introduce very succinctly the ideas behind the numerical solutions of PDEs. Numerical integration of PDEs is based on replacing the continuous variables by discrete variables. The continuum problem represented by the PDE is discretized and transformed in a problem in finitely many variables. The price we must then pay is that we can only hope to find approximations to the exact solution at a discrete set of points.

In practice, this is not such a big limitation. In many situations, there is no need to know the solution with infinite accuracy. For instance, when solving a heat conduction problem, it is rarely required to obtain an answer with an accuracy better than, say, one hundredth of a degree. Said differently, your exact analytical solutions provide you in general with more information than is useful!

So the main idea of a numerical method is to replace the PDE, formulated for an unknown real-valued function, by a discrete equation involving a finite number of variables. This discrete problem is called a **numerical scheme**. Through this process, one replaces the PDE by an algebraic equation. For instance, when the PDE is linear, we obtain a system of linear algebraic equations to solve. In very general terms, the accuracy of your solution generally depends on the number of discrete variables you are using (i.e. the number of algebraic equations). Therefore, when seeking an accurate approximation, you may have to solve quite large algebraic systems and at this point, the knowledge that you have acquired in numerical analysis would prove useful!

There are several techniques for converting a PDE into a discrete problem. The most popular numerical methods are called: (1) the **finite difference method** (FDM) and (2) the **finite element method**. When compared to the long history of analytical solution methods for PDEs, the field of numerical solutions of PDEs is comparatively young. There are on-going debates about which of these two methods is the best.

One way to look at this question is that both methods can be used for most problems, including equations with constant or nonconstant coefficients, and even nonlinear equations. That being said, as the finite difference method is far simpler to introduce

and to program, this is the method that we will focus on in this short section. The power of the finite element method becomes quite clear when one wants to solve problems in complicated geometries.

**Remark.** *In this section, we will try to remain succinct as no matter how much you may read about numerical methods, the best way to fully understand and appreciate these methods is by trying to implement them. So in this section, I would encourage you to devote most of your time working through the method (see Problem Sheet #7).*

*Finally, if you want to explore these concepts further, you may find some of the Year 3/4 electives interesting including: Computational PDEs (MATH96021), Finite Elements (MATH96063) or Scientific Computation (MATH96055).*

## 4.1 Finite differences

As its name indicates, the finite difference method relies on approximating the derivatives appearing in your PDE problem by **finite differences**. To illustrate this, we consider a PDE problem whose solution is a smooth function in two variables $u(x, y)$, defined over the domain $D = [0, a] \times [0, b]$. We partition the domain using a discrete grid of points (also called a **mesh**). In the case of a **uniform mesh** in both variables, this mesh is defined by the set of points

$$\{(x_i, y_j) = (i\Delta x, j\Delta y), \quad 0 \leq i \leq N, \quad 0 \leq j \leq M\} \tag{4.1}$$

where the grid spacing in both dimensions is given by $\Delta x = a/N$ and $\Delta y = b/M$ with $N$ and $M$ integers. The larger their values, the finer the mesh-size. In the finite difference method, we are interested in the values that $u$ takes at these points; we can approximate the solution $u(x, y)$ at the mesh points by writing

$$u_{i,j} = u(x_i, y_j) \tag{4.2}$$

The whole idea behind the finite difference method is to transform our initial PDE problem in a set of algebraic equations for the $u_{i,j}$ which we will compute numerically. In doing so, there are several questions we need to worry about:

- how do we find equations for the approximate solutions $u_{i,j}$? These are called **numerical schemes**. As we will see, discretizing a PDE turns out to be a nontrivial task. Equations of different types should be handled numerically differently.
- If I have obtained a set of equations, how do I know that the computed solution is a good approximation to the exact solution?
- What does one mean by ”good”? How good is good and how can I quantify the error?
- Can I analyze the numerical scheme to ensure that I converge to the exact solution as I take $\Delta x, \Delta y \to 0$, i.e. as the mesh becomes finer and finer?

These questions form the essence of the numerical analysis of PDEs and we will only be scratching the surface here!

### 4.1.1 Discretization scheme

Our PDE is an equation which involves the partial derivatives of our function $u(x, y)$, so before we get into the actual discretization of PDEs, let us consider how we would approximate a derivative of a function $u(x)$ if we only know values of the function at

discrete points $x_i = i\Delta x$, with $i \in [0, N]$. Using a Taylor's expansion of $u$ around the point $x_i$, we can compute $u(x_{i+1})$ as follows

$$u(x_{i+1}) = u(x_i) + u'(x_i)\Delta x + \frac{1}{2}u''(x_i)(\Delta x)^2 + \frac{1}{6}u'''(x_i)(\Delta x)^3 + \cdots \tag{4.3}$$

It follows that to linear order in $\Delta x$, we get

$$u'(x_i) = \frac{u(x_{i+1}) - u(x_i)}{\Delta x} + \mathcal{O}(\Delta x) \tag{4.4}$$

So we have here obtained a first approximation for the derivative of $u$ with respect to $x$ which is called the **forward difference formula (FDF)**

$$u'(x_i) \approx \frac{u_{i+1} - u_i}{\Delta x} \equiv D_+ u_i \tag{4.5}$$

Similarly, one can derive a **backward difference formula (BDF)**

$$u'(x_i) \approx \frac{u_i - u_{i-1}}{\Delta x} \equiv D_- u_i \tag{4.6}$$

where the notations $D_+u_i$ and $D_-u_i$ are classical notations for the FDF and BDF for a uniform mesh size $\Delta x$. Note that what we just did directly carries over if $u$ is function of several variables; in this case, what we have defined are approximations to the partial derivative of $u(x, y)$ with respect to $x$. An important question is how accurate are our FDF and BDF. As $\Delta x$ becomes smaller and smaller, how does the error between the approximate and exact derivatives decrease?

### 4.1.2 Truncation error

This leads us to introduce what is called the **truncation error** of an approximation. Take for instance the case of the forward difference formula, the argument is similar for the case of the backward difference formula. The expression we obtained says that if you are given $\{u_i\}_{i=0,\ldots,N}$ then an approximation to the derivative is given by $D_+u_i$. Now we can replace $u_{i+1}$ by its exact value which is $u(x_{i+1}) = u(x_i + \Delta x)$. The truncation error is the difference between the exact derivative and the formula for $D_+$ applied to $u$, i.e.

$$T_{\mathrm{FDF}}(\Delta x) = \left| u'(x_i) - \frac{u_{i+1} - u_i}{\Delta x} \right| \tag{4.7}$$

Once again, Taylor's theorem tells us that

$$u(x \pm \Delta x) = u(x) \pm \Delta x u'(x) + \frac{(\Delta x)^2}{2}u''(x) \pm \frac{(\Delta x)^3}{6}u'''(x) + \mathcal{O}\left((\Delta x)^4\right) \tag{4.8}$$

Hence, we have

$$\frac{u_{i+1} - u_i}{\Delta x} = \frac{1}{\Delta x}\left(u(x_i) \pm \Delta x u'(x_i) + \frac{(\Delta x)^2}{2}u''(x_i) + \cdots - u(x_i)\right) \tag{4.9}$$

$$= u'(x_i) + \frac{\Delta x}{2}u''(x_i) + \cdots \tag{4.10}$$

It follows that the truncation error is given by

$$T_{\mathrm{FDF}}(\Delta x) = \left| \frac{\Delta x}{2}u''(x_i) + \cdots \right| \tag{4.11}$$

Hence, we can say that the truncation error is

$$T_{\mathrm{FDF}}(\Delta x) \leq C\Delta x \tag{4.12}$$

where $C$ is a positive constant. Note that, in general, we do not know the value of the constant $C$ but what is important is that $T_{\mathrm{FDF}}(\Delta x) = \mathcal{O}(\Delta x)$. To minimize the truncation error and thus obtain a more faithful approximation for the derivative, we need $\Delta x$ to be very small. Now since $\Delta x = \mathcal{O}(1/N)$, it means that we would require $N$ to be very large. For instance, if you double the number of grid points, then the error can go down by a factor of $2$. This is called **linear convergence**.

Working with large $N$ is computationally expensive (as you have more equations to solve) and requires more memory. Here is an interesting question: can we do better than linear? Can we introduce a finite difference approximation for our derivative that is more accurate than the FDF or BDF we just introduced? After all, we have the sequence $\{u_i\}_{i=0,\ldots,N}$ anyway, so would different combinations of its elements lead to better approximations?

### 4.1.3 Centered finite differences

To see that, consider the Taylor's expansion of $u(x_i + \Delta x)$ and subtract from it the expansion for $u(x_i - \Delta x)$ obtaining

$$u(x_i + \Delta x) - u(x_i - \Delta x) = 2\Delta x u'(x_i) + \frac{(\Delta x)^3}{3}u'''(x_i) + \mathcal{O}\left((\Delta x)^5\right) \tag{4.13}$$

Rearranging the terms in (4.13), we can solve for $u'(x_i)$ and obtain a better approximation of the derivative

$$u'(x_i) \approx \frac{u_{i+1} - u_{i-1}}{2\Delta x} \equiv Du_i \tag{4.14}$$

This is called the **centered difference formula (CDF)**. It can easily be shown that the central difference is the average of the forward and backward difference formulas

$$Du_i = \frac{1}{2}(D_+u_i + D_-u_i) \tag{4.15}$$

The truncation error of the CF formula is now

$$T_{\mathrm{CDF}}(\Delta x) = \mathcal{O}\left((\Delta x)^2\right) \tag{4.16}$$

which follows directly from (4.13). With a very minor modification, we have obtained a scheme with quadratic error, which means that if I double the number of grid points, the error goes down by a factor of 4. Figure 4.1 shows a comparison of the three discretization scheme we just introduced.

### 4.1.4 Higher-order schemes

Naturally, you may ask whether we can go further. In other words, can we improve the error and get quadratic convergence for instance? The answer is yes! As you may have already guessed, this will require us to bring in information from grid points further away from the grid point $i$. Note that central differences always lead to a better accuracy. As an example, here is a fourth-order difference for the first-order derivative $u'$

$$u'(x_i) \approx \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} \equiv D_4u_i \tag{4.17}$$

The truncation error for this discretization scheme is then

$$T_{D_4}(\Delta x) = \mathcal{O}\left((\Delta x)^4\right) \tag{4.18}$$

which is the reason why it is called a fourth-order scheme. When halving the step-size, we reduce the error by a factor of $16$.

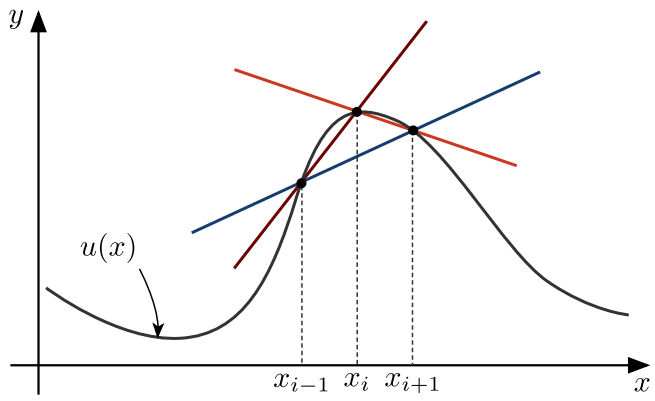


**Figure 4.1** Comparison of the discretization schemes for first-order derivatives: forward difference (orange), backward difference (red) and centered difference (blue).

### 4.1.5 Global vs local error

Our approximation of $u'$ was done at any one of the grid points $x_i$. So, for each grid point, one can calculate the local error associated with the approximation for any of the discretization schemes. Rather than checking the errors for each grid point, it is much more convenient to add all these local errors over the interval. Defining this global error also ensures that the approximation is uniformly valid over the domain and alerts you of programming errors. Let us see how to do it for the forward differences (note that the procedure is analogous for the other discretization schemes). We define

$$e(t) \equiv \sum_{i=0}^{M-1} T_i(\Delta x) = \sum_{i=0}^{M-1} \left| u'(x_i) - \frac{u_{i+1} - u_i}{\Delta x} \right| \tag{4.19}$$

where $T_i(\Delta x)$ is the local truncation error for this particular scheme. Now this simple definition does not give me the truncation error over the interval directly. Indeed, we are here adding $M$ terms but since $\Delta x = \mathcal{O}(1/N)$, i.e. $N = \mathcal{O}(1/\Delta x)$, we would expect $e(t)$ to be of order $1/\Delta x$ times the actual truncation error! The correct way to define the error is, therefore, as follows

$$E(t) \equiv \Delta x \sum_{i=0}^{M-1} T_i(\Delta x) = \Delta x \sum_{i=0}^{M-1} \left| u'(x_i) - \frac{u_{i+1} - u_i}{\Delta x} \right| \tag{4.20}$$

with similar expressions in the case of other discretization schemes.

### 4.1.6 Higher-order derivatives

Using a similar method, one can also derive second-order central differences for the second derivatives of $u$ and obtain

$$u''(x_i) \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} \tag{4.21}$$

We can also generalize very easily what we have just introduced here to functions of multiple variables. Let us go back to our initial function $u(x, y)$ of two independent variables. The second-order central finite difference scheme for first-order partial derivatives

will be given by

$$\frac{\partial u}{\partial x}(x_i, y_j) \approx \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x} \tag{4.22}$$

$$\frac{\partial u}{\partial y}(x_i, y_j) \approx \frac{u_{i,j+1} - u_{i,j+1}}{2\Delta y} \tag{4.23}$$

and those for second-order derivatives

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \approx \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{(\Delta x)^2} \tag{4.24}$$

$$\frac{\partial^2 u}{\partial y^2}(x_i, y_j) \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{(\Delta y)^2} \tag{4.25}$$

$$\frac{\partial^2 u}{\partial x \partial y}(x_i, y_j) \approx \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y} \tag{4.26}$$

In summary, using Taylor's theorem, we can derive finite difference schemes of an order for derivatives of any order. For instance, for central finite difference, we summarize in Table 4.1 the coefficients appearing in derivatives of orders 1 and 2 for several orders of accuracy (with uniform grid).

| Deriv. | Acc. | -4 | -3 | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | | | | -1/2 | 0 | 1/2 | | | |
| | 4 | | | 1/12 | -2/3 | 0 | 2/3 | -1/12 | | |
| | 6 | | -1/60 | 3/20 | -3/4 | 0 | 3/4 | -3/20 | 1/60 | |
| | 8 | 1/280 | -4/105 | 1/5 | -4/5 | 0 | 4/5 | -1/5 | 4/105 | -1/280 |
| 2 | 2 | | | | 1 | -2 | 1 | | | |
| | 4 | | | -1/12 | 4/3 | -5/2 | 4/3 | -1/12 | | |
| | 6 | | 1/90 | -3/20 | 3/2 | -49/18 | 3/2 | -3/20 | 1/90 | |
| | 8 | -1/560 | 8/315 | -1/5 | 8/5 | -205/72 | 8/5 | -1/5 | 8/315 | -1/560 |

**Table 4.1** Coefficients appearing in central finite differences for derivatives of order 1 and 2, for several orders of accuracy and with uniform grid spacing.

**Remark.** *The coefficients for higher order derivatives and for forward and backward differences can be found in a very useful Wikipedia page!*

## 4.2 Numerical integration of the diffusion equation

At this point, we are able to discretize any derivative. So it may seem like the construction of a numerical scheme, i.e. converting a PDE to a discrete problem is quite a simple task: all one has to do is replace each derivative appearing in the PDE by finite difference approximation and a numerical scheme pops up! Well, it turns out that it is not so simple. Several difficulties frequently arise during the process and we shall discuss some of these in this section.

Let us illustrate this on the following one-dimensional diffusion problem

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \quad 0 < x < \pi, \ t > 0 \tag{4.27}$$

$$u(0,t) = u(\pi,t) = 0, \quad t > 0 \tag{4.28}$$

$$u(x,0) = f(x), \quad 0 \le x \le \pi \tag{4.29}$$

where we assume $f(0) = f(\pi) = 0$. Let's an integer $N > 2$ and a positive number $\Delta t$, and set $\Delta x = \pi/(N-1)$. We define a grid in space $\{x_i = i\Delta x\}$ on the interval $[0, \pi]$ and a grid in time $\{t_n = n\Delta t\}$ on the interval $[0, T]$. We will denote $u_{i,n} = u(x_i, t_n)$.

### 4.2.1 FCTS explicit scheme

One of the simplest finite difference schemes for this problem is based on a first-order forward difference for the time derivative and a central difference for the spatial derivative leading to

$$\frac{u_{i,n+1} - u_{i,n}}{\Delta t} = D\frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{(\Delta x)^2} \quad 1 \leq i \leq N-2,\ n \geq 0 \tag{4.30}$$

This numerical scheme is called **FTCS** for **F**orward in **T**ime, **C**entered in **S**pace. Note that the boundary values are determined by the boundary conditions (4.28), i.e.

$$u_{0,n} = u_{N-1,n} = 0, \quad n \geq 0 \tag{4.31}$$

This scheme is particularly simple as using simple algebraic manipulations, we can obtain an explicit expression for the discrete solution at each point at time $n+1$ in terms of the solutions at time $n$, namely

$$u_{i,n+1} = u_{i,n} + \alpha\left(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}\right) \quad 1 \leq i \leq N-2,\ n \geq 0 \tag{4.32}$$

where $\alpha = D\Delta t/(\Delta x)^2$. Finally, the initial condition for the PDE becomes an initial condition for the difference equation

$$u_{i,0} = f(x_i) \tag{4.33}$$

Without too much effort, we derived a simple algorithm for the numerical solution of the diffusion equation. So one may ask where might the problem be! The problem is that, unless we are careful in our choice of $\Delta t$ and $\Delta x$, the difference scheme (4.32) is not stable. This means that a small perturbation to the initial conditions will grow very fast in time. The representation of numbers in the computer being always finite, every numerical solution will inevitably include some **round-off error**. Therefore, it is necessary to ensure that our numerical scheme is stable against small perturbations (including these round-off errors).

### 4.2.2 Stability of numerical schemes

We denote $\mathbf{v}$ the vector of unknowns, i.e. the values of the solution on the grid points where the value of $u$ is unknown. Indeed, note that the solution $u$ is known at the grid points where the boundary and initial conditions are given. We write our discrete problem as $T(\mathbf{v}) = \mathbf{w}$, where $\mathbf{w}$ is a vector containing the known parameters of the problem (e.g. initial condition or boundary conditions).

**Definition 4.2.1: Stability**

Let $T(\mathbf{v}) = \mathbf{w}$ be a numerical scheme. Let $\mathbf{v}^i$ be two solutions, i.e. $T(\mathbf{v}^i) = \mathbf{w}^i$, for $i = 1, 2$. The scheme $T$ is **stable** if

$$\forall \varepsilon > 0, \exists\, \delta : |\mathbf{w}^1 - \mathbf{w}^2| < \delta \Rightarrow |\mathbf{v}^1 - \mathbf{v}^2| < \varepsilon \tag{4.34}$$

In other words, a small change in the problem's data implies a small change in the solution.

For the FTCS scheme for the diffusion equation, the external conditions are given by the initial conditions $f(x)$ and the boundary conditions $u(0,t) = u(\pi,t) = 0$. To examine stability of the scheme for an arbitrary perturbation, we can choose an initial condition of the form $f(x) = \sin kx$ for some positive integer $k$. Indeed, we have seen in the previous

Chapter that periodic solutions to the diffusion equation on a finite domain are written as a linear combinations of such sinusoidal modes.Therefore, the stability with respect to any such initial condition implies stability for arbitrary perturbations. Conversely, if there exists an integer $k$ for which the scheme is unstable, then this implies instability with respect to any arbitrary perturbation.

Given the form of the solutions of the diffusion equation that we obtained in Section 3.3.2, we seek a solution for (4.32) of the form

$$u_{i,n} = A(n) \sin ki\Delta x \tag{4.35}$$

Substituting this into (4.32) leads to the following difference equation for $A(n)$

$$A(n+1) = A(n)\left[1 + \alpha \frac{\sin(k(i+1)\Delta x) - 2\sin(ki\Delta x) + \sin(k(i-1)\Delta x)}{\sin ki\Delta x}\right] \tag{4.36}$$

Now using the identity

$$\frac{\sin(k(i+1)\Delta x) - 2\sin(ki\Delta x) + \sin(k(i-1)\Delta x)}{\sin ki\Delta x} = -4\sin^2\left(\frac{k\Delta x}{2}\right) \tag{4.37}$$

we can simplify the equation for $A(n)$, $n \geq 1$ and get

$$A(n+1) = \left[1 - 4\alpha \sin^2\left(\frac{k\Delta x}{2}\right)\right] A(n) \tag{4.38}$$

Therefore $\{A(n)\}_{n\geq 1}$ is a geometric sequence and

$$A(n) = \left[1 - 4\alpha \sin^2\left(\frac{k\Delta x}{2}\right)\right]^n A(0) \tag{4.39}$$

and as a consequence

$$u_{i,n} = \left[1 - 4\alpha \sin^2\left(\frac{k\Delta x}{2}\right)\right]^n \sin(ki\Delta x) \tag{4.40}$$

Since $1 - 4\alpha \sin^2(k\Delta x/2) < 1$, it follows that a necessary and sufficient condition for stability of the difference equation (4.32), i.e. for a perturbation not to grow out of bounds, is

$$1 - 4\alpha \sin^2(k\Delta x/2) > -1 \tag{4.41}$$

Therefore, we conclude that we cannot choose $\Delta t$ and $\Delta x$ arbitrarily; they must satisfy the **stability condition**

$$\Delta t \leq \frac{1}{2D}(\Delta x)^2 \tag{4.42}$$

This is, therefore, rather bad news! Remember that in order to obtain a "good" approximation of the derivative, one usually wants to select a small value of $\Delta x$. So although this numerical scheme is very easy to program, the stability condition requires that we choose a **very small** timestep $\Delta t$; this implies potentially very large number of time steps to compute to obtain the solution as some finite time. The conclusion is that I have obtained (with limited effort) a numerical scheme for the diffusion equation with the unfortunate limitation of tiny time steps. So one may wonder if with a little more effort, we may be able to derive other numerical schemes which may not have this limitation or at least not such a drastic one.

### 4.2.3 Richardson method

You may suspect that the first-order time difference is to blame here. To try to test this hypothesis, let us consider a similar numerical scheme, except that the time difference is now of second-order (i.e. a centered difference in time)

$$\frac{u_{i,n+1}-u_{i,n-1}}{2\Delta t}=D\frac{u_{i+1,n}-2u_{i,n}+u_{i-1,n}}{(\Delta x)^2},\quad 1\leq i\leq N-2,\ n\geq 0 \tag{4.43}$$

You should notice immediately that the values of $u_{i,n-1}$ are not defined at all for $n=0$. This is a common technical difficulty; one usually overcomes this problem by using the FTCS scheme just for one time step, and then proceeding with this new scheme (4.43).

Surprisingly, it turns out that this promising numerical scheme is unstable for **any** choice of time step! To see this, we can construct as we did above a solution for the difference equation of the form $u_{i,n}=A(n)\sin ki\Delta x$. In this case, the sequence $A(n)$ satisfies the difference equation

$$A(n+1)=A(n-1)-8\alpha\sin^2\left(\frac{k\Delta x}{2}\right)A(n) \tag{4.44}$$

The solution to this recurrence relation is given by $A(n)=A(0)r^n$, where $r$ is a root of the following quadratic equation

$$r^2+2\beta r-1=0 \tag{4.45}$$

where we have defined $\beta=4\alpha\sin^2(k\Delta x/2)$. Notice that one of the roots of this polynomial is

$$r=-\beta\pm\sqrt{\beta^2+1} \tag{4.46}$$

so the root of this polynomial with negative sign always satisfies $|r|>1$, which implies that the scheme is always unstable! In trying to improve our stability, we have made things much worse as there is now value of $\Delta t$ for which (4.43) is stable.

The problem of finding efficient and stable numerical schemes has attracted over the years intense activity leading to the development of numerous numerical schemes. However, the point of this section is not to catalog these numerical schemes...

### 4.2.4 Implicit schemes

Notice that the FTCS scheme (4.32) is particularly simple to implement in your favorite programming language. Indeed if we denote $\mathbf{u}_n$ the vector containing the value that our solution takes at all spatial grid points, we were able to write down an explicit expression for the solution $\mathbf{u}_{n+1}$ at time $n+1$ in terms of the solution $\mathbf{u}_n$. Such a scheme is called an **explicit scheme**. Practically speaking, we can easily write our difference equation in matrix

$$\mathbf{u}_{n+1}=A\mathbf{u}_n \tag{4.47}$$

where $A$ is a tridiagonal matrix. As a consequence, advancing our solution in time corresponds to a simple matrix multiplication. In this explicit scheme, we have used forward differences in time, let us explore what sort of scheme we obtain if we use backward differences instead.

Using a **B**ackward difference in **T**ime and a **C**entral difference in **S**pace leads to the so-called **BTCS**, i.e.

$$\frac{u_{i,n+1}-u_{i,n}}{\Delta t}=D\frac{u_{i+1,n+1}-2u_{i,n+1}+u_{i-1,n+1}}{(\Delta x)^2}\quad 1\leq i\leq N-2,\ n\geq 0 \tag{4.48}$$

which leads to the following difference equation

$$-u_{i,n} = -u_{i,n+1} + \alpha \left( u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1} \right) \quad 1 \leq i \leq N-2,\ n \geq 0 \tag{4.49}$$

where $\alpha = D\Delta t/(\Delta x)^2$ or equivalently, we can write

$$u_{i,n} = -\alpha u_{i+1,n+1} + (1+2\alpha) u_{i,n+1} - \alpha u_{i-1,n+1} \quad 1 \leq i \leq N-2,\ n \geq 0 \tag{4.50}$$

In contrast with the explicit schemes, such a numerical scheme is called an **implicit scheme**.

Let us analyze the stability of the BTCS scheme. We can proceed exactly like we did for the FTCS scheme and seek a solution of the form $u_{i,n} = A(n) \sin ki\Delta x$. Substituting this in the implicit scheme (4.48), we write the following difference equation for $A(n)$

$$A(n) = A(n+1) \left[ 1 - \alpha \frac{\sin(k(i+1)\Delta x) - 2\sin(ki\Delta x) + \sin(k(i-1)\Delta x)}{\sin ki\Delta x} \right] \tag{4.51}$$

using as before the identity

$$\frac{\sin(k(i+1)\Delta x) - 2\sin(ki\Delta x) + \sin(k(i-1)\Delta x)}{\sin ki\Delta x} = -4\sin^2\left(\frac{k\Delta x}{2}\right) \tag{4.52}$$

we obtain that $\{A(n)\}_{n\geq 1}$ is a geometric sequence and

$$A(n) = \left[ 1 + 4\alpha \sin^2\left(\frac{k\Delta x}{2}\right) \right]^{-n} A(0) \tag{4.53}$$

with

$$\left| 1 + 4\alpha \sin^2\left(\frac{k\Delta x}{2}\right) \right|^{-1} < 1 \tag{4.54}$$

So we conclude that this implicit scheme is stable for any choice of $\Delta t$ and $\Delta x$! We say in this case that the numerical scheme is **unconditionally stable**. A *rule of thumb* in numerical analysis is that implicit schemes are better behaved than explicit schemes and should often be preferred (although one should not conclude that implicit schemes are always valid). The better behavior manifests itself, for example, in higher efficiency or higher accuracy.

If we denote once again $\mathbf{u}_n$ the vector containing the value that our solution takes at all spatial grid points, our implicit scheme can be written in matrix form

$$\mathbf{u}_n = A\mathbf{u}_{n+1} \tag{4.55}$$

and so

$$\mathbf{u}_{n+1} = A^{-1}\mathbf{u}_n \tag{4.56}$$

As a consequence, the advantages of implicit schemes are counterbalanced by the fact that at each time step we need to solve an algebraic system of equations with $N-2$ unknowns. For instance, in the case of linear PDEs like the diffusion equation, this leads to the inversion of a potentially very large matrix. This explains that a major theme in numerical analysis is to derive efficient methods for solving large algebraic systems which can be nonlinear when you deal with nonlinear PDEs.

Finally, note that while it is unconditionally stable, the BTCS scheme is first-order accurate in time and second-order accurate in space; this is commonly denoted $\mathcal{O}\left(\Delta t, (\Delta x)^2\right)$. Is it possible to preserve the unconditional stability but to improve accuracy?

The Crank-Nicolson scheme (introduced in 1947) does exactly this. This implicit scheme is commonly written

$$\frac{u_{i,n+1} - u_{i,n}}{\Delta t} = D\left(\frac{u_{i+1,n} - 2u_{i,n} + u_{i-1,n}}{2(\Delta x)^2} + \frac{u_{i+1,n+1} - 2u_{i,n+1} + u_{i-1,n+1}}{2(\Delta x)^2}\right) \tag{4.57}$$

i.e.

$$-\alpha u_{i+1,n+1} + 2(1+\alpha)u_{i,n+1} - \alpha u_{i-1,n+1} = \alpha u_{i+1,n} + 2(1-\alpha)u_{i,n} + \alpha u_{i-1,n} \tag{4.58}$$

where we have defined $\alpha$ as before. Once again, as all the terms on the RHS are known, the equations (4.58) form a tridiagonal linear system

$$A\mathbf{u}_{n+1} = \mathbf{b}_n \tag{4.59}$$

where $\mathbf{b}_n$ is a vector of known quantities (i.e. involving the solution at time step $n$) and $\mathbf{u}_{n+1}$ is the vector of unknowns, i.e. the value of the solution at the grid points at time step $n+1$.

**Remark.** *Using the same procedure as above. Can you show that this scheme is unconditionally stable?*

Up until now, there is one detail we have swept under the rug. That is how to impose boundary conditions in these numerical schemes. In what follows, we will show how to impose Dirichlet and Neumann boundary conditions in the FTCS scheme. A similar procedure would be used to impose boundary conditions in other numerical schemes.

### 4.2.5 Imposing Dirichlet boundary conditions in the FTCS scheme

Consider that we are trying to solve our 1D diffusion equation on the real interval $[0, a]$. Say that we have split our interval $[0, a]$ in $N$ sub-intervals, which means that we have $N+1$ grid points in space with indices ranging in $[0, N]$.

Let us start by trying to impose Dirichlet boundary conditions to our 1D diffusion problem; in doing so, we impose the value of the solution on the edges of the domain, namely

$$u(0,t) = u_L(t) \tag{4.60}$$
$$u(a,t) = u_R(t) \tag{4.61}$$

where $u_L$ and $u_R$ are potentially time-dependent functions. With these boundary conditions, we know the values of the solution $u_{0,n}$ and $u_{N,n}$ at all time steps $n$. This means that we do not need to compute the solution at this grid points, we only need to compute the solution for grid points with indices $[1, N]$. These are commonly called the **interior points** (or interior domain). We only need to implement our numerical scheme for these interior points. The boundary grid points are then **exterior to the domain**; these points are sometimes called "false points".

Conceptually, the easiest thing to do here is to write the FTCS scheme for all the grid points which are not on the edge of the interior domain, i.e.

$$u_{i,n+1} = u_{i,n} + \alpha(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}), \quad i \in [2, N-2] \tag{4.62}$$

Indeed, on the boundary of the interior domain, i.e. for indices $i = 1$ and $i = N-1$, we need to write the FTCS scheme using the boundary condition as

$$u_{1,n+1} = u_{1,n} + \alpha\left[u_{2,n} - 2u_{1,n} + u_L(t_n)\right] \tag{4.63}$$
$$u_{N-1,n+1} = u_{N-1,n} + \alpha\left[u_R(t_n) - 2u_{N-1,n} + u_{N-2,n}\right] \tag{4.64}$$

We can conclude that this numerical scheme composed of $N-1$ linear equations involving $N-1$ unknowns can be written explicitly in matrix form as

$$\mathbf{u}_{n+1} = A\mathbf{u}_n + \mathbf{b}_n \tag{4.65}$$

where $\mathbf{u}_n$ is a $(N-1)$-dimensional vector whose entries are the values of the solution for all grid points $i \in [1, N-1]$ at time step $n$. The matrix $A$ is a tridiagonal matrix whose diagonal and off-diagonal elements can be read directly from the linear system of equations to give

$$A = \begin{pmatrix} 1-2\alpha & \alpha & 0 & 0 & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & 0 \\ 0 & \alpha & 1-2\alpha & \alpha & 0 \\ 0 & 0 & \alpha & 1-2\alpha & \alpha \\ 0 & 0 & 0 & \alpha & 1-2\alpha \end{pmatrix} \tag{4.66}$$

(where we have used here without loss of generality $N-1=5$) and $\mathbf{b}_n$ is a vector involving the Dirichlet boundary conditions given by

$$\mathbf{b}_n = \begin{pmatrix} \alpha u_L(t_n) \\ 0 \\ 0 \\ 0 \\ \alpha u_R(t_n) \end{pmatrix} \tag{4.67}$$

### 4.2.6 Imposing Neumann boundary conditions in the FTCS scheme

Finally, using the same geometry, let us see how we can deal with Neumann boundary conditions. When imposing Neumann boundary conditions, we impose the value of the derivative of the solution at the boundaries, namely

$$\frac{\partial u}{\partial x}(0,t) = v_L(t) \tag{4.68}$$

$$\frac{\partial u}{\partial x}(a,t) = v_R(t) \tag{4.69}$$

Here, we are not setting the value of the solution at the grid points $i=0$ and $i=N$; it thus needs to be computed, it needs to be part of our numerical scheme, i.e. we need to obtain algebraic equations for $u_{0,n}$ and $u_{N,n}$. In the FTCS scheme, we use central differences for the space derivatives which means in particular that the first-order derivative of the solution at the boundary $x=0$ (respectively, $x=a$) would involve the value of the solution at $x_{-1}$ (respectively, at $x_{N+1}$) which are points outside of our domain. To solve this problem, we will use here the concept of "false points" again and define $x_{-1}$ and $x_{N+1}$ as being exterior to the domain on which we are trying to solve this problem.

Now, at $x_0 = 0$, we want to impose the Neumann boundary condition

$$\frac{\partial u}{\partial x}(0,t) = v_L(t) \approx \frac{u_{1,n} - u_{-1,n}}{2\Delta x} \tag{4.70}$$

where we approximated the derivative of $u$ using a central difference formula. This allows us to write an expression for $u_{-1,n}$ which only involves either known quantities or values of the solutions at grid points actually inside the domain

$$u_{-1,n} = u_{1,n} - 2\Delta x v_L(t_n) \tag{4.71}$$

Similarly, at $x_N = a$, we impose the Neumann boundary condition and write

$$\frac{\partial u}{\partial x}(a,t) = v_R(t) \approx \frac{u_{N+1,n} - u_{N-1,n}}{2\Delta x} \tag{4.72}$$

which after rearranging terms allows us to write

$$u_{N+1,n} = u_{N-1,n} + 2\Delta x v_R(t_n) \tag{4.73}$$

Now that we have a way to express the value of the solution at these exterior grid points as a function of values on the interior of the domain (and the boundary conditions), we can write the FTCS scheme first on the interior points as

$$u_{i,n+1} = u_{i,n} + \alpha(u_{i+1,n} - 2u_{i,n} + u_{i-1,n}), \quad i \in [1, N-1] \tag{4.74}$$

and secondly, on the boundaries of the domain as

$$\begin{aligned} u_{0,n+1} &= u_{0,n} + \alpha\left[u_{-1,n} - 2u_{0,n} + u_{1,n}\right] && (4.75)\\ &= u_{0,n} + \alpha\left[2u_{1,n} - 2u_{0,n}\right] - 2\alpha\Delta x v_L(t_n) && (4.76)\\ u_{N,n+1} &= u_{N,n} + \alpha\left[u_{N-1,n} - 2u_{N,n} + u_{N+1,n}\right] && (4.77)\\ &= u_{N,n} + \alpha\left[2u_{N-1,n} - 2u_{N,n}\right] + 2\alpha\Delta x v_R(t_n) && (4.78) \end{aligned}$$

where we have used equations (4.71) and (4.73) to express the unknown values $u_{-1,n}$ and $u_{N+1,n}$

Finally, we can conclude that the FTCS scheme with Neumann boundary conditions can be written in matrix form as

$$\mathbf{u}_{n+1} = A\mathbf{u}_n + \mathbf{b}_n \tag{4.79}$$

with $A$ the following tridiagonal matrix

$$A = \begin{pmatrix} 1-2\alpha & 2\alpha & 0 & 0 & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & 0 \\ 0 & \alpha & 1-2\alpha & \alpha & 0 \\ 0 & 0 & \alpha & 1-2\alpha & \alpha \\ 0 & 0 & 0 & 2\alpha & 1-2\alpha \end{pmatrix} \tag{4.80}$$

and $\mathbf{b}_n$ the following $(N+1)$ dimensional vector

$$\mathbf{b}_n = \begin{pmatrix} -2\alpha\Delta x\ v_L(t_n) \\ 0 \\ 0 \\ 0 \\ 2\alpha\Delta x\ v_R(t_n) \end{pmatrix} \tag{4.81}$$

(where we have used here without loss of generality $N+1=5$).

Note that in this final case, the linear system we obtained has size $(N+1)\times(N+1)$ as we have not excluded the boundary grid points: we need to compute the solutions at these grid points here as we are not setting their values but the values of the derivative at the boundary.

### 4.2.7 Some concluding remarks

Hopefully, this very short introduction to numerical methods for PDEs tickled your curiosity. Obviously, one could explore this numerical integration of PDEs further and entire modules and countless reference texts are dedicated to this topic; however, sometimes less is more! We have now introduced all the concepts that we may need to tackle the problems that we will introduce in the next section on reaction-diffusion equations.