An introductive course to some numerical approximation methods for ordinary and partial differential equations

Franz Chouly

An introductive course to some numerical approximation methods for ordinary and partial differential equations

Franz Chouly

Institut de Mathématiques de Bourgogne
franz.chouly@u-bourgogne.fr

May 4, 2021

Introduction

These lectures notes were written for students of the first year of the international Master 'Maths for Physics' at the University of Burgundy in 2021, as a part of the course called 'Numerical Methods'. It was designed for a total of 22 hours of classes. The main objective of this course is to provide a first insight into numerical methods to solve mathematical problems inspired from physics and engineering. It focuses both on the mathematical fundations and justifications of numerical methods and on practical aspects related to their implementation. We start to study ordinary differential equations (ODEs), and present some methods for their discretization, *i.e.*, to approximate the time-derivative. Then we focus on partial differential equations (PDEs), and present a well-known method, the Finite Element Method (FEM), to approximate the solution of elliptic PDEs. The last part will be devoted to time-dependent PDEs, such as the heat equation, where we combine both ideas coming from the two previous chapters. I thank deeply all the students for their remarks associated to this course, and for their enthusiasm despite the very special context, and I thank also my colleagues at the IMB and ICB for their support.

Contents

1 Discretization of ODEs

This part concerns ordinary differential equations and their numerical approximation. Ordinary differential equations are omnipresent to describe the behavior of many systems in physics and applied science. For instance (see [12]), if you are interested in computing the velocity v of a small ball of mass m when it is dropped from a given height, you need to solve:

$$v'(t) = g - \frac{k}{m}v(t),\ t > 0, \quad v(0) = v_0,$$

where t is the time variable, g is the gravity constant, k is the resistance of the air, and v_0 is the initial value of the velocity. If you are interested in predicting the motion of a spherical pendulum, you need to solve:

$$\begin{aligned} 0 &= -\sin(\theta(t)) + \sin(\theta(t))\cos(\theta(t))(\dot{\varphi}(t))^2 - \ddot{\theta}(t) \\ 0 &= -\sin^2(\theta(t))\ddot{\varphi}(t) - 2\sin(\theta(t))\cos(\theta(t))\dot{\varphi}(t)\dot{\theta}(t), \quad t > 0 \\ & \varphi(0) = \varphi_0,\ \dot{\varphi}(0) = \dot{\varphi}_0,\ \theta(0) = \theta_0,\ \dot{\theta}(0) = \dot{\theta}_0, \end{aligned}$$

where (θ, φ) represent the Euler angles of a pendulum of lenght 1 (with mass and gravity constant 1, also). Initial conditions are given, respectively the initial position (θ_0, φ_0) and velocity $(\dot{\theta}_0, \dot{\varphi}_0)$ It is well known that a lot of differential equations modelling physical problems do not admit, up to what is known today, closed-form solutions. For instance the first ODE we presented may have one, but the second one surely not. So we need some techniques to approximate their solution efficiently and accurately.

1.1 Introduction and simplest schemes

Let us start to study this simple *initial-value problem*:

$$\left\{ \begin{array}{rcl} y'(t) &=& f(t, y(t)), \\ y(0) &=& y^0. \end{array} \right. \tag{1}$$

Here $y : \mathbb{R}^+ \to \mathbb{R}$ is the (unknown) solution, a function that depends on the (time) variable t, y' is the first-order derivative of y, and $f : \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ a given (known) function of two variables. The (real) scalar y^0 is known. The first equation of (1) is a scalar first-order ordinary differential equation (ODE), and the second one is the initial condition, at time $t = 0$. We suppose that f is kind enough, for instance globally Lipschitz, so that (1) admits one unique solution y, well-defined and continuously differentiable on the whole interval $\mathbb{R}^+$. This setting is simplistic, of course, but will be sufficient to introduce almost all the ideas related to numerical approximation. Most of the techniques and proofs that we will see in the reminder extend easily to more general setting (ODE well-posed only within a bounded interval of $\mathbb{R}^+$, second-order ODE, systems of ODE, etc).

Suppose that f, though kind enough to make (1) is well-posed, is not so kind, however, so that it is easy to find an explicit, closed-form solution. So if we are

really interested in solving this, one realistic option is to solve it approximately using a computer. The main problem, at this stage, is that a computer is very efficient to perform quickly basic arithmetical operations, but not clever enough to understand the concepts of function and derivative. So we can not give it directly (1) and tell him : 'Please, solve this !'. We need to preprocess it for him, so that he has to deal only with sequences of numbers. The simplest way to do this is to slice the half-real line into small slabs, of size τ, where $\tau > 0$ is a chosen parameter, called the *time–step*. As a result we define the sequence of real values $t_n := n\tau$, for any $n \in \mathbb{N}$:

$$0 = t_0 < t_1 (= \tau) < \ldots < t_n (= n\tau) < \ldots$$

To each value of n, we also associate a real unknown y_n, and we will work enough so that at the end, the computer provides an explicit value of y_n such that

$$y_n \simeq y(t_n),$$

and we will precise later on the meaning of $\simeq$. The idea is that, if τ is small enough, the sequence (y_n) is fine enough to represent approximately the function y. Of course, plenty of values of y are missing, but could be, if necessary, recovered, using, for instance, linear interpolation between two successive values y_n and y_{n+1}.

Now, we need to approximate the value of the derivative y' using the sequence (y_n). The basic idea is to make use of the *finite difference*:

$$y'(t_n) \simeq \frac{y_{n+1} - y_n}{\tau}.$$

Let us just remark for the moment, that, if (y_n) represents accurately the function y, and if τ is very small, we should approximate correctly the derivative. The initial condition is simply transposed as

$$y_0 = y^0.$$

From now on, three possible options arise naturally for the discretization of (1), that depend each one of the treatment of the right-and side of the first equation $(f(t, y(t)))$:

1. The first option: we take $f(t, y(t))$ at time–step n, which leads to the equation, for all $n \geq 0$:

$$\frac{y_{n+1} - y_n}{\tau} = f(t_n, y_n). \tag{2}$$

   This is called the *forward Euler scheme*.

2. The second option: we take $f(t, y(t))$ at time–step $n + 1$, which leads to the equation, for all $n \geq 0$:

$$\frac{y_{n+1} - y_n}{\tau} = f(t_{n+1}, y_{n+1}). \tag{3}$$

   This is called the *backward Euler scheme*.

3. The third option: we take a barycentric combination of time–steps n and $n+1$. This leads to the following equation, for all $n \geq 0$:

$$\frac{y_{n+1} - y_n}{\tau} = \frac{1}{2}(f(t_n, y_n) + f(t_{n+1}, y_{n+1})). \tag{4}$$

This is called the *Crank-Nicolson scheme*.

Before entering the mathematical analysis of these procedures, let us just give a first insight into their implementation. We illustrate the implementation aspects using Octave, which is an open-source clone of Matlab, that has a graphical user interface and a script language very convenient for scientific computing. It can be downloaded here:

`https://www.gnu.org/software/octave/index`

There is also an online version of Octave, available here:

`https://octave-online.net`

We recommend to download the package on your computer, which makes its use more comfortable. Nevertheless, the above online version can be helpful in case of emergency for instance. Of course, other alternatives can be chosen if preferred, for instance the commercial software Matlab (if you are rich enough), the open-source french software Scilab (from the INRIA, `https://www.scilab.org`) or Python-based solutions such as SciPy, with nice interfaces such as Spyder or Jupyter notebooks (`https://www.scipy.org/about.html`).

Let us start with the implementation of the first scheme (2), forward Euler. Note that the relationship (2) can be rewritten:

$$y_{n+1} = y_n + \tau f(t_n, y_n). \tag{5}$$

This provides a simple recurrence formula to compute the approximation y_{n+1}, at time t_{n+1}, for the previous approximated solution y_n at time t_n. There is nothing more to do. Since the approximated solution at a given discrete time is inferred explicitely from the previous solution, this scheme is called *explicit* (and is sometimes referred as explicit Euler time-marching scheme).

For the backward Euler scheme (3), it can be also rewritten as follows:

$$y_{n+1} - \tau f(t_{n+1}, y_{n+1}) = y_n. \tag{6}$$

This provides also a recurrence formula, but this is not as simple as for forward Euler. Indeed, the approximation y_{n+1} is not obtained explicitely, and, instead, needs to be found as the zero of the function $I - \tau f(t_{n+1}, \cdot) - y_n$ (I denotes the identity). This operation can be done by many ways, for instance using Newton's method, but at the price of an extra effort, and an extra computational expense. For this reason, this scheme is called *implicit* (and is often referred to

as implicit Euler time–marching scheme). Despite of this difficulty, this scheme remains interesting in many situations. We will see the precise reason for this later on.

The Crank-Nicolson scheme (4) can be rewritten as:

$$y_{n+1} - \frac{1}{2}\tau f(t_{n+1}, y_{n+1}) = y_n + \frac{1}{2}\tau f(t_n, y_n). \tag{7}$$

Remark that this scheme remains still implicit. Remark also that any other convex combination between the time–steps n and $n+1$ could be meaningful, but Crank-Nicolson refers to the values $(\frac{1}{2}, \frac{1}{2})$ (otherwise we talk about θ–schemes). At first glance, this scheme seems to have no advantage above the backward Euler scheme, but we will see later on it has some.

A final remark about implementation is that, though all these schemes are very simple to implement, their main drawback is that, to obtain the solution at a given final time, you need to compute iteratively all the intermediate solutions, which can be long for some problems. In fact, for ODEs, there is no straightforward way to solve them quickly using parallel architectures, and some special schemes are needed if you want to achieve this (see for instance [6] or [7]).

Now, let us provide another viewpoint on the above schemes. Let us integrate the first equation of (1) between time t_n and time t_{n+1}:

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))\mathrm{d}t. \tag{8}$$

In some sense, we transform our initial problem to a problem of another kind: the computation of an integral. Now let us try, instead of approximating the derivative y' in (1), to approximate the integral in the right-hand side of (8). There are many *quadrature rules* that can be used, and let us try with the simplest ones. For instance, if we use the *left rectangle rule*

$$\int_a^b \varphi(t)\mathrm{d}t \simeq (b-a)\varphi(a)$$

we get from (8):

$$y(t_{n+1}) - y(t_n) \simeq \tau f(t_n, y(t_n)) \tag{9}$$

from which we recover the forward Euler scheme, by substituting y_n to $y(t_n)$ for each value of n. You can also show, as an exercice, that the backward Euler scheme, resp. the Crank-Nicolson scheme, is recovered identically from the right rectangle rule, resp. the trapezoidal rule.

Now, to conclude this first part, let us give some very first notions about numerical stability of the above schemes. To this purpose, we consider the following variant of (1):

$$\left\{ \begin{array}{rcl} y'(t) & = & -\alpha\, y(t), \\ y(0) & = & 1. \end{array} \right. \tag{10}$$

Here $\alpha > 0$ is a real parameter. The above equation has the following explicit solution:

$$y(t) = \exp(-\alpha t).$$

This solution remains *bounded*, in the sense that its modulus remains smaller than 1:

$$|y(t)| \leq 1, \qquad \forall t \in \mathbb{R}^+.$$

Let us approximate now this solution using the forward Euler scheme. Following equation (5), we obtain:

$$y_{n+1} = y_n - \alpha\tau y_n = (1 - \alpha\tau)\, y_n. \tag{11}$$

Here, there is no need to use a computer to obtain the discrete solution, which is simply given by the formula below, for any value of n:

$$y_n = (1 - \alpha\tau)^n.$$

So the modulus of the discrete solution is:

$$|y_n| = |1 - \alpha\tau|^n.$$

We expect, that, at least, the discrete solution remains bounded, but we observe here that this is not always the case. A necessary and sufficent condition, in this situation, is that:

$$|1 - \alpha\tau| \leq 1$$

or, equivalently:

$$-1 \leq 1 - \alpha\tau \leq 1$$

which can be reformulated as

$$\tau \leq \frac{2}{\alpha},$$

(where we used that α and τ are positive). So the discrete solution (y_n) remains bounded only if the time–step τ is small enough. We say in this case that the forward Euler scheme is *conditionally stable*: stable if τ is small enough, unstable otherwise. Of course, we want to avoid numerical unstability, since, in this situation, the discrete solution approximates very poorly the exact one. Using the same argument, one can assess that the backward Euler scheme and the Crank-Nicolson scheme are *unconditionnaly stable* (these are good exercices).

1.2 One-step methods

Now that we have a first insight into numerical approximation of ODEs, and before going into more complex (and widely used) methods, let us be more precise and go into the mathematical analysis of the simplest methods. To this purpose, let us introduce a general class of approximation methods, called *one–step* methods, as follows:

$$y_{n+1} = y_n + \tau\Phi_\tau(t_n, y_n), \tag{12}$$

for $n \in \mathbb{N}$, and where $\Phi_\tau : \mathbb{R}^+ \times \mathbb{R} \to \mathbb{R}$ is a given function, that may depend on the time–step τ, and that is supposed at least continuous. The simplest example of one-step method is the forward Euler scheme (5), and this case corresponds to

$$\Phi_\tau(t, y) = f(t, y).$$

We have seen previously that y_n is only an approximation of $y(t_n)$, and we will try now to say something general and precise about this approximation. For this purpose, we introduce the *global error*:

$$e_n := y(t_n) - y_n$$

at each step n. We introduce also the *truncation error* T_n as

$$T_n := \frac{y(t_{n+1}) - y(t_n)}{\tau} - \Phi_\tau(t_n, y(t_n)),$$

where y is the solution to Problem (1). The truncation error is the basic notion, and represents roughly how much the one-step method approximates the ODE. The first interesting point is that we can infer a bound on the global error from the truncation error (see [16]):

Theorem 1 *Suppose that, for the one-step method* (12)*, the function* Φ_τ *is Lipschitz: there exists* $L > 0$ *such that, for all* $t \in \mathbb{R}_+$ *and* $u, v \in \mathbb{R}$*, there holds:*

$$|\Phi_\tau(t, u) - \Phi_\tau(t, v)| \leq L|u - v|, \tag{13}$$

then there holds, for all $n \in \mathbb{N}$*:*

$$|e_n| \leq \frac{1}{L}\left(e^{Lt_n} - 1\right) \max_{k=0,\ldots,n} |T_k|. \tag{14}$$

Proof. First let us rewrite differently the definition of the truncation error:

$$y(t_{n+1}) = y(t_n) + \tau\Phi_\tau(t_n, y(t_n)) + \tau T_n$$

and then we substract (12):

$$y(t_{n+1}) - y_{n+1} = y(t_n) - y_n + \tau\Phi_\tau(t_n, y(t_n)) - \tau\Phi_\tau(t_n, y_n) + \tau T_n$$

so we obtain an equation on the global error:

$$e_{n+1} = e_n + \tau\left(\Phi_\tau(t_n, y(t_n)) - \Phi_\tau(t_n, y_n)\right) + \tau T_n.$$

Now we use the triangle inequality and the Lipschitz condition (13) to bound the left hand-side above:

$$|e_{n+1}| \leq |e_n| + \tau L\,|y(t_n) - y_n| + \tau|T_n|,$$

which can be rewritten:

$$|e_{n+1}| \leq (1 + \tau L)|e_n| + \tau|T_n|.$$

Now, let us show by induction that, for all $n \geq 0$, there holds:

$$|e_n| \leq \frac{\Theta_n}{L}\left((1+\tau L)^n - 1\right) \tag{15}$$

where $\Theta_n := \max_{k=0,\ldots,n} |T_k|$. Indeed, for $n = 0$, there holds:

$$e_0 = y(0) - y_0 = y^0 - y^0 = 0,$$

so that (15) holds for $n = 0$. Suppose now that (15) is true for n. From this relationship and the bound above, we deduce:

$$\begin{aligned} |e_{n+1}| &\leq (1+\tau L)|e_n| + \tau|T_n| \\ &\leq (1+\tau L)\frac{\Theta_n}{L}\left((1+\tau L)^n - 1\right) + \tau\Theta_n \\ &\leq \frac{\Theta_n}{L}(1+\tau L)^{n+1} - (1+\tau L)\frac{\Theta_n}{L} + \tau\Theta_n \\ &\leq \frac{\Theta_{n+1}}{L}(1+\tau L)^{n+1} - 1) \end{aligned}$$

where we used the (obvious) bound $\Theta_n \leq \Theta_{n+1}$. We then use the relationship

$$1 + \tau L \leq \exp(\tau L),$$

(remember $\exp(\cdot)$ is a convex function) to infer from (15):

$$|e_n| \leq \frac{\Theta_n}{L}\left(\exp(n\tau L) - 1\right),$$

and remarking that $\tau n = t_n$, we obtain (14). This ends the proof. □

Let us now apply this result to the forward Euler scheme:

Corollary 1 *Let us consider the forward Euler method* (2) *to discretize Problem* (1)*, and make the assumption that f is globally Lipschitz: there exists $L > 0$ such that, for all $t \in \mathbb{R}_+$ and $u, v \in \mathbb{R}$, there holds:*

$$|f(t,u) - f(t,v)| \leq L|u - v|, \tag{16}$$

then there holds, for all $n \in \mathbb{N}$:

$$|e_n| \leq \frac{\tau}{2L}\left(e^{Lt_n} - 1\right) \max_{t\in[0;t_n]} |y''(t)|. \tag{17}$$

Proof. It suffices to apply Theorem 1 to the forward Euler scheme, recalling that $\Phi_\tau = f$, so L is the Lipschitz constant of f. Let us estimate the consistency error, which expression is, for the forward Euler method:

$$T_n = \frac{y(t_{n+1}) - y(t_n)}{\tau} - f(t_n, y(t_n)).$$

Then we use the first equation from (2):

$$y'(t_n) = f(t_n, y(t_n))$$

and the Taylor-Lagrange expansion of y at time t_n:

$$y(t_{n+1}) = y(t_n) + \tau y'(t_n) + \frac{\tau^2}{2} y''(\xi_n),$$

with $\xi_n \in [t_n; t_{n+1}]$. We then deduce:

$$T_n = \frac{\tau y'(t_n) + \frac{\tau^2}{2} y''(\xi_n)}{\tau} - y'(t_n) = y'(t_n) + \frac{\tau}{2} y''(\xi_n) - y'(t_n).$$

Therefore:

$$T_n = \frac{\tau}{2} y''(\xi_n).$$

This ends the proof. □

As a result, the forward Euler scheme (2) converges in $\mathcal{O}(\tau)$. We say this is a *first-order* scheme, or a scheme of order one. More generally, we say that a scheme is of order p ($p \geq 0$) when convergence is in $\mathcal{O}(\tau^p)$. The above statement is important. Not only it tolds us that the approximation error vanishes when τ decreases, but it provides us an information with respect to the convergence rate (or speed). For a scheme of order one, notably, this means that dividing by two the time step results in a global error divided by two. This is not so bad, and this makes this scheme already applicable for numerical solving of ODEs. Nonetheless, in terms of accuracy, it remains still among the worst schemes one can think about. This is the reason why we will see in the next part of this chapter some schemes that allow to increase the accuracy. These schemes are not as simple as forward Euler, but they are not that much complicated and the gain in accuracy is sometimes huge.

Before seing other schemes, let us state a few additional remarks corresponding to Theorem 1 and Corollary 1. Particularly, note that, as expected, the bound on the global error increases, exponentially, with time. As a result, the numerical solution gets less and less accurate as time is evolving. This is something you can observe generally in practice. This is a problem for long-time simulations (think about simulation in celestial dynamics, astrophysics, or for spatial technology), which has motivated the need of very accurate schemes. Note also that the bound on the global error depends on the solution itself and its variations: in Corollary 1, it depends on y'', so, as it can be expected 'genuinely', an ODE for which the solution has great/irregular variations with time can be more difficult to approximate correctly.

1.3 Runge-Kutta (RK) methods

Runge-Kutta (RK) methods are still one-step methods, but for which f is evaluated at some intermediate points. If chosen correctly, these evaluation allow

to recover a better accuracy than the forward Euler method. For instance the *modified Euler method* is defined by the recurrence formula

$$y_{n+1} = y_n + \tau f\left(t_n + \frac{1}{2}\tau, y_n + \frac{1}{2}\tau f(t_n, y_n)\right).$$

It can be shown, as an exercice, that this method is *second-order* accurate: the bound in the global error is in $\mathcal{O}(h^2)$, this time. The gain in accuracy is substantial: dividing by two the time step, the approximation error is divided by four. Plenty of RK methods exist (see for instance the books of Butcher [4], and Hairer, Norsett & Wanner [12, 13] for an extensive study of these methods). Let us just mention the most popular one, the *classical fourth-order RK method* (see [16]):

$$y_{n+1} = y_n + \frac{1}{6}\left(\tilde{y}_n^1 + 2\tilde{y}_n^2 + 2\tilde{y}_n^3 + \tilde{y}_n^4\right),$$

where

$$\begin{aligned}
\tilde{y}_n^1 &= f(t_n, y_n), \\
\tilde{y}_n^2 &= f(t_n + \frac{1}{2}\tau, y_n + \frac{1}{2}\tau\tilde{y}_n^1), \\
\tilde{y}_n^3 &= f(t_n + \frac{1}{2}\tau, y_n + \frac{1}{2}\tau\tilde{y}_n^2), \\
\tilde{y}_n^4 &= f(t_n + \tau, y_n + \tau\tilde{y}_n^3).
\end{aligned}$$

An (awkward) calculation allows to assess that this method is fourth-order: the approximation error is in $\mathcal{O}(h^4)$. This is very good, still with a method that is quite simple. Remark that the above RK methods are both *explicit*: y_{n+1} is obtained directly, without the need to solve an equation involving f.

1.4 Linear multistep methods

A drawback of above RK methods is that they may require too many evaluations of f to improve the accuracy. Therefore, another strategy to improve the accuracy is to use previous computations y_{n-1}, y_{n-2}, ..., and not only y_n, to predict y_{n+1}. The interesting point, here, is that these previous computations have already been made, and stored somewhere on your favorite computer. This is the main idea of *linear multistep methods*, which are, for instance, behind the function `lsode` in Octave. Following [16], a *linear k-step method* is defined according to the formula below:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \tau \sum_{j=0}^{k} \beta_j f(t_{n+j}, y_{n+j}). \tag{18}$$

In (18) the coefficients $\alpha_0, \ldots, \alpha_k$ and $\beta_0, \ldots, \beta_k$ are real values. Each different choice corresponds to a different method. We will assume that: 1) either α_0 or β_0 is different from 0, 2) $\alpha_k \neq 0$, so as to avoid degenerate cases. Remark that

if $\beta_k = 0$, the method is *explicit*: y_{n+k} is obtained directly from previous values of the approximate solution. If $\beta_k \neq 0$, the method is *implicit*: an algebraic equation needs to be solved at each time-step to recover y_{n+k}. To alleviate the notation, we will use the same convention as in the book of Süli & Mayers [16] and set:

$$f_n := f(t_n, y_n).$$

Let us present first some popular examples of linear multistep methods:

- For the choice

  $$\beta_0 = 1, \quad \beta_1 = 0, \quad \alpha_0 = -1, \quad \alpha_1 = 1,$$

  we get

  $$y_{n+1} - y_n = \tau f_n,$$

  which is the (explicit) forward Euler scheme (2).

- For the choice

  $$\beta_0 = 0, \quad \beta_1 = 1, \quad \alpha_0 = -1, \quad \alpha_1 = 1,$$

  we get

  $$y_{n+1} - y_n = \tau f_{n+1},$$

  which is the (implicit) backward Euler scheme (3).

- For the choice

  $$\beta_0 = \frac{1}{2}, \quad \beta_1 = \frac{1}{2}, \quad \alpha_0 = -1, \quad \alpha_1 = 1,$$

  we get

  $$y_{n+1} - y_n = \frac{\tau}{2}(f_n + f_{n+1}),$$

  which is the (implicit) Crank-Nicolson scheme (4).

- For the choice

  $$\beta_0 = -\frac{9}{24}, \ \beta_1 = \frac{37}{24}, \beta_2 = -\frac{59}{24}, \ \beta_3 = \frac{55}{24}, \beta_4 = 0,$$

  and

  $$\alpha_0 = 0, \ \alpha_1 = 0, \ \alpha_2 = 0, \ \alpha_3 = -1, \ \alpha_4 = 1,$$

  we get

  $$y_{n+4} - y_{n+3} = \frac{\tau}{24}(-9f_n + 37f_{n+1} - 59f_{n+2} + 55f_{n+3}),$$

  which is the (explicit) Adams-Bashforth four-step scheme.

- For the choice

$$\beta_0 = \frac{1}{24},\ \beta_1 = -\frac{5}{24}, \beta_2 = \frac{19}{24},\ \beta_3 = \frac{9}{24},$$

and

$$\alpha_0 = 0,\ \alpha_1 = 0,\ \alpha_2 = -1,\ \alpha_3 = 1,$$

we get

$$y_{n+3} - y_{n+2} = \frac{\tau}{24}(f_n - 5f_{n+1} + 19f_{n+2} + 9f_{n+3}),$$

which is the (implicit) Adams-Moulton three-step scheme.

Note first that this formalism allows to recover the three different schemes presented in 1.1. The last two schemes are new. You already have guessed that the difficulty in the design of sophisticated linear multistep methods is to set appropriately the values of the coefficients (α_j) and (β_j). They must satisfy conditions that make the scheme stable and accurate (consistent and convergent). These last properties are vital: if they are not satisfied, the method is of no use in practice. Let us say a little bit more about these properties.

1.5 Stability and accuracy

Let us consider a linear multistep scheme to solve the initial-value problem (1). Let us now suppose that this problem is well-posed on a bounded time-interval $[0, T]$, and let N be the natural number such that $T = N\tau$ (we suppose that the time-step divides exactly T). We need k values $y_0, \ldots, y_{k-1}$ to initiate the computation of the approximate solution using the recurrence formula (18). The first thing to note is that, due to (limited) machine precision, at least, the initial values $y_0, \ldots, y_{k-1}$ will not be used exactly to compute the approximated solution (y_n), but instead we will only have approximate values: let us call them $\hat{y}_0, \ldots, \hat{y}_{k-1}$. They should satisfy, of course:

$$\hat{y}_0 \simeq y_0, \qquad \hat{y}_{k-1} \simeq y_{k-1},$$

with a precision of, for instance, ten digits (or more). The precision for the initial condition can be very good, but the final approximate solution y_n at a time t_n may be obtained from these initial values after thousands (or more) calculations. This means that the final solution can be very inaccurate, due to the accumulation of round-off errors. To limit such effects, we want to design linear multistep schemes that verify the following property, known as *zero-stability*:

The linear k-step method (18) is said *zero-stable* if there exists a constant $K > 0$ such that, for any small enough value of the time-step τ, and for any sequence (y_n) generated from (18) and initial conditions $y_0, \ldots, y_{k-1}$, and another sequence $(\hat{y}_n)$, still generated using (18), and initial conditions $\hat{y}_0, \ldots, \hat{y}_{k-1}$, there holds

$$|y_n - \hat{y}_n| \leq K \max_{0 \leq j \leq k-1} |y_j - \hat{y}_j|, \tag{19}$$

for any natural number n smaller than N.

Let us introduce now a very useful, and nice, result to assess in practice if a multistep method is zero-stable (or not). This is in fact an algebraic equivalent of zero-stability and is called *root condition*. For this purpose, let us define the first and second characteristic polynomials associated to (18), as, respectively:

$$P_\alpha(z) = \sum_{j=0}^{k} \alpha_j z^j, \qquad P_\beta(z) = \sum_{j=0}^{k} \beta_j z^j,$$

for $z \in \mathbb{C}$. To start with, we need the following lemma:

Lemma 1 *Let us consider the recurrence relation*

$$\alpha_k u_{n+k} + \alpha_{k-1} u_{n+k-1} + \ldots + \alpha_1 u_{n+1} + \alpha_0 u_n = 0, \tag{20}$$

for n a natural number, with the associated characteristic polynomial

$$P(z) = \alpha_k z^k + \alpha_{k-1} z^{k-1} + \ldots + \alpha_1 z + \alpha_0,$$

for $z \in \mathbb{C}$. We suppose $\alpha_0 \neq 0$ and $\alpha_k \neq 0$. Let us write P in factorized form

$$P(z) = \alpha_k (z - z_1)^{m_1} (z - z_2)^{m_2} \ldots (z - z_{l-1})^{m_{l-1}} (z - z_l)^{m_l},$$

where $z_1, z_2, \ldots, z_{l-1}, z_l$ are the l distinct (complex) roots of P, of associated multiplicities $m_1, m_2, \ldots, m_{l-1}, m_l$. Then, if a sequence (u_n) satisfies (20)*, it can be written as follows*

$$u_n = \sum_{j=1}^{l} P_j(n) z_j^n, \tag{21}$$

for any natural number n. In the above formula, the P_j are polynomials of degree $m_j - 1$.

We can admit this Lemma for the moment. The proof can be found in the book of Süli & Mayers [16, Lemma 12.1]. We provide now our main result known as the *Root Condition*, which is a necessary and sufficent condition for zero-stability [16].

Theorem 2 (Root Condition) *Let us consider the linear k-step method* (18) *to solve the initial-value problem* (1)*. Then the method* (18) *is zero-stable if, and only if, the roots of its first characteristic polynomial P_α are all within the domain*

$$\mathcal{S}_1 := \{z \in \mathbb{C} \,|\, |z| \leq 1\}$$

with, additionally, each root z_j that satisfies $|z_j| = 1$ being simple.

Proof. Let us prove that the Root Condition is a necessary condition.

Let us take the particular case $f = 0$ in (1), so that we solve $y' = 0$ with initial condition y_0, which solution is a constant one: $y(t) = y_0$ for $t \geq 0$. In this case, the method (18) reads

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0. \tag{22}$$

We then apply the above Lemma 1 to infer that the approximate solution (y_n) can be written as

$$y_n = \sum_{j=1}^{l} P_j(n) z_j^n.$$

Therefore, if there exists a root z_j of the first characteristic polynomial P_α that satisfies $z_j \notin \mathcal{S}_1$, this means that $|z_j| > 1$ and therefore

$$|z_j|^n \to +\infty$$

where n tends to $+\infty$. As a result, there exists a set of initial values $y_0, \ldots, y_{k-1}$ such that $|y_n|$ tends to $+\infty$ when n tends to $+\infty$ (i.e., when $\tau \to 0$, since $n \leq N$ and $N\tau = T$). The other case we need to consider is the case of a root z_j on the boundary of $\mathcal{S}_1$, ($|z_j| = 1$) of multiplicity greater than 1. Since its associated polynomial P_j is of multiplicity $m_j - 1 \geq 1$, then it grows with n as $n^{m_j - 1}$, and once again, there exists a set of initial values $y_0, \ldots, y_{k-1}$ such that $|y_n|$ tends to $+\infty$ when n tends to $+\infty$.

Let us take now another sequence $\hat{y}_0 = 0, \hat{y}_1 = 0, \ldots, \hat{y}_{k-1} = 0$ such that the sequence $(\hat{y}_n)$ generated from these initial values and (22) be identically equal to 0. It appears then that there is no way that the property (19) be satisfied for the two sequences (y_n) and $(\hat{y}_n)$, when the time-step τ is small enough. Therefore we proved that if the root condition is not satisfied, zero-stability does not hold any more.

The proof that the Root Condition is a sufficent condition is much more technical, and can be admitted. For a detailed proof, see for instance the book of Gautschi, Section 6.3 [10]. □

Let us now apply this result to assess that some of the presented linear multistep methods are zero-stable. First, let us consider the forward Euler method (2). The associated first characteristic polynomial is

$$P_\alpha(z) = z - 1,$$

which has one unique simple root $z_1 = 1$, that verifies $|z_1| = 1$. So Theorem 2 ensures that the forward Euler method is zero-stable. The same property holds for the backward Euler scheme (3) and the Crank-Nicolson scheme (4). For the Adams-Bashforth four-step scheme, the first characteristic polynomial is

$$P_\alpha(z) = z^4 - z^3 = z^3(z - 1),$$

with a triple root $z_1 = 0$ and a simple root $z_2 = 1$, so the Root Condition holds, and Theorem 2 ensures that it is zero-stable. You can treat the case of the Adams-Moulton three-step scheme as an exercice.

Let us provide also a counter-example: the following linear multi-step method

$$y_{n+3} + y_{n+2} - y_{n+1} - y_n = 2\tau(f_{n+2} + f_{n+1}),$$

is not zero-stable, though it can look nice at first glance. Indeed, the first characteristic polynomial is

$$P_\alpha = z^3 + z^2 - z - 1 = z(z^2 - 1) + z^2 - 1 = (z + 1)(z^2 - 1) = (z + 1)^2(z - 1),$$

has a double root $z_1 = -1$ on the boundary of $\mathcal{S}_1$, so the Root Condition is not satisfied, and Theorem 2 ensures that it can not be zero-stable.

In fact, as you may have noticed in the proof of Theorem 2, the Root Condition is linked with solving the ODE $y' = 0$, with zero right-hand side. This is why we talk about 'zero'-stability. In some sense, a multistep scheme is zero-stable if small perturbations in the initial data imply small perturbations in the whole (discrete) trajectory, and this notion of zero-stability can be viewed as a sensibility to initial small perturbations.

For a multistep method to be useful, it not only needs to be (zero-)stable, it should also be accurate. So the other crucial notion for this purpose is that of *consistency*. Let us take y the unique solution to the initial-value problem (1) and the recurrence formula (18) for a linear k-step scheme. We define the corresponding *truncation error* as follows:

$$T_n := \frac{\sum_{j=0}^k \alpha_j y(t_{n+j}) - \tau \sum_{j=0}^k \beta_j f(t_{n+j}, y(t_{n+j}))}{\tau \sum_{j=0}^k \beta_j}. \tag{23}$$

Though it looks more complicated, one can realize it is a quite straightforward adaptation of the definition for one-step scheme, with still the same idea behind: the exact solution y should satisfy approximately the recurrence relation (18). Note however the normalization by the quantity

$$\tau \sum_{j=0}^k \beta_j (= P_\beta(1)),$$

that should be supposed different from 0. We now define the consistency property as follow: the multistep scheme (18) is *consistent* if, for any $\varepsilon > 0$, there exists $\tau_\varepsilon > 0$, such that, for any value of the time-step τ that satisfies $\tau \leq \tau_\epsilon$, and for any natural number n, there holds

$$|T_n| \leq \varepsilon.$$

Let us now use some Taylor expansions to rewrite the numerator of (18):

$$\begin{aligned}
&\sum_{j=0}^{k} \alpha_j y(t_{n+j}) - \tau \sum_{j=0}^{k} \beta_j f(t_{n+j}, y(t_{n+j})) \\
= \;& \sum_{j=0}^{k} \alpha_j y(t_{n+j}) - \tau \sum_{j=0}^{k} \beta_j y'(t_{n+j}) \\
= \;& \sum_{j=0}^{k} \alpha_j \left(y(t_n) + (j\tau) y'(t_n) + \mathcal{O}(\tau^2) \right) \\
& -\tau \sum_{j=0}^{k} \beta_j \left(y'(t_n) + (j\tau) y''(t_n) + \mathcal{O}(\tau^2) \right) \\
= \;& y(t_n) \sum_{j=0}^{k} \alpha_j + \tau \left(y'(t_n) \sum_{j=0}^{k} (j\alpha_j - \beta_j) \right) + \mathcal{O}(\tau^2).
\end{aligned}$$

where we used first that y is the solution to (1), and then we applied Taylor expansions of y and y' at time t_n (supposing that y is smooth enough). Finally, we just put together all the terms of the same order in τ. So (necessary and sufficent) conditions for consistency are

$$\sum_{j=0}^{k} \alpha_j = 0, \qquad \sum_{j=0}^{k} j\alpha_j = \sum_{j=0}^{k} \beta_j,$$

which can be rewritten also

$$P_\alpha(1) = 0, \qquad P'_\alpha(1) = P_\beta(1) (\neq 0).$$

Remark particularly that any consistent method is associated with a simple root of the first characteristic polynomial, that belongs to the boundary of $\mathcal{S}_1$ (this is, luckily, still compatible with zero-stability).

As an example, still we consider the forward Euler method. It is indeed consistent, since

$$\sum_{j=0}^{k} \alpha_j = -1 + 1 = 0, \qquad \sum_{j=0}^{k} j\alpha_j = 0 + 1 = 1 = 1 + 0 = \sum_{j=0}^{k} \beta_j.$$

The same calculations can be done as exercices for the other linear multistep methods provided previously. In fact, we can go further and find the largest natural number p such that $T_n = \mathcal{O}(\tau^p)$. Algebraic conditions on the coefficients (α_j) and (β_j) can be derived following the same path as previously. It only requires Taylor expansions to order higher than 2 when we reformulate the truncation error (23) (see the book of Süli & Mayers [16], Section 12.8, for more details). As for one-step methods, we then talk about methods of order

p, and once again, the interest to derive methods of high order is to increase the accuracy of the discrete solution (y_n), with only a slight increase in the computational cost.

We end this section with an important result that allows us to understand why both notions of zero-stability and consistence are fundamental (and sufficient) to assess that the approximated solution (y_n) obtained with the multistep scheme (18) is accurate, in the sense it gets closer to $y(t_n)$ as τ becomes smaller. This is a well-known result in the theory of numerical ODEs, due to Germund Dahlquist. The proof is difficult and will be omitted for this introductory course.

Theorem 3 (Dahlquist's Equivalence Theorem) *Let us consider the initial-value problem* (1)*, with f globally Lipschitz, and the multistep scheme* (18)*, that is supposed to be consistent. Let us suppose that the initial values $y_0 \dots y_{k-1}$ are chosen consistently. Therefore, the multistep scheme* (18) *is zero-stable, if, and only if, it converges. Moreover, if the exact solution y has continuous derivatives up to order p and the truncation error satisfies $T_n = \mathcal{O}(\tau^p)$, for $p \geq 1$, then the global error $e_n = y(t_n) - y_n$ satisfies*

$$e_n = \mathcal{O}(\tau^p).$$

The above Theorem 3 ensures the convergence of the five multistep methods we have presented. For the forward Euler method (2) especially, we recover the result obtained thanks to the Corollary 1. We are done now with this first tour about numerical approximation of ODEs. Before entering in the (wonderful) world of numerical PDEs, we just give a few extra comments on this topic.

1.6 Extra remarks

This first part about numerical approximation of ODEs is just a small introduction, but can be enough to have a first idea on the topic, and for the reminding part of this course. We will revisit this topic at the end, when we will study PDEs with evolution in time. There are many issues we have not discussed, and if you are interested, there are many good books that deal with the topic (among many others, let us quote [4, 9, 12, 13, 16]). In fact, numerical analysis of ODEs has been object of intensive research along the twentieth century, and many methods have been invented and analysed. This is still an active research topic, though.

To give you a few hints about other issues, first, there is also an important result by Germund Dahlquist, the so-called *Dalquist barrier*, which tells you, roughly speaking, that the condition of zero-stability limits the multistep methods in terms of accuracy [16]. Also, another issue are *stiff* systems of ODEs, which present some specific difficulties in terms of numerical approximation, and for which we need a stronger notion of stability (*A-stability*) and specific multistep schemes (BDF schemes for instance). A good and simple introduction to this issue can be found, still, in the book of Süli & Mayers [16, Section 12.11], and one can have a look at [13] for an extensive treatment.

Also, a fascinating topic is about geometric numerical integration and numerical schemes that preserve some important properties for some classes of dynamical systems (symplectic numerical integration for instance). An introduction to these issues can be found in the book of 'Geometric Numerical Integration' of Ernst Hairer, Gerhard Wanner and Christian Lubich [11].

2 FEM for elliptic PDEs

Let us move now from ODEs to Partial Differential Equations (PDEs). PDEs allow to model a great range of physical phenomena very accurately, since common physical laws, such as the conservation of mass, momentum, energy, can be expressed using PDEs. Basically, and as you know, PDEs relate some unknown quantities (the temperature, the motion of material points, etc) to their derivative in time and space. The most famous PDEs are known sometimes since the eighteen or nineteen centuries, as well as closed-form solutions for very specific configurations. Various approximation techniques have been invented from the end of the nineteen century, so as to calculate effective approximate solutions. Now, the most popular methods are the *finite difference method*, the *finite element method*, the *finite volume method* and the *spectral method*. An introductive presentation to such methods is given for instance in the book of Martin Gander and Felix Kwok [8].

As an introduction to this topic, we will focus here on the finite element method, and on the most simple elliptic partial differential equation (a linear, scalar, PDE). We first present it, and then reformulate it to find what is called its *weak formulation*. Indeed, the finite element method does not discretize directly the differential operators. In fact, it relies on the construction of an appropriate *finite element space*, which is a finite dimensional vector space, that approximates the true vector space in which we seek the solution (which is an infinite dimensional vector space, endowed with a Hilbert (or Banach) structure). At the end of the whole process, we will simply obtain an algebraic linear system, that can be implemented and solved using a computer. Finally, we will answer the question : does the finite element method provide an accurate approximation of the exact solution ?

2.1 Model problem: reaction-diffusion

Let us introduce a first model problem, as simple as possible. A PDE provides a relationship between the different derivatives, in time and space, of an unknown. Let us forget about time for the moment and consider only PDEs in space, that correspond to *stationnary* problems (the solution does not evolve in time). In general (but not always) the solution of the PDE is defined only on a subdomain of the whole space. Let us denote by Ω this subdomain. We will suppose that Ω is an open subset of $\mathbb{R}^d$, $d \geq 1$ (in practice, $d = 1, 2$ or 3). We will suppose that Ω is non-empty and bounded. We will denote by $\Gamma := \partial\Omega$ the boundary of Ω. For $d = 1$ we will take Ω as a finite union of open intervals. For $d = 2$ (respectively $d = 3$) the boundary Γ will be supposed polygonal (respectively polyhedral). This last assumption can be alleviated in fact, but will avoid some technical difficulties at a specific moment. We do not make any extra assumption about Ω, and, for instance, it does not need to be connected or convex.

We want to solve the following problem:

$$c\, u - \Delta u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \Gamma. \tag{24}$$

In (24) $u : \Omega \to \mathbb{R}$ is the unknown solution, $c \geq 0$ is a constant, $f : \Omega \to \mathbb{R}$ is a (known) *source term*. The operator Δ is the Laplace operator, defined as

$$\Delta v := \text{div } (\nabla v) = \left(\frac{\partial^2}{\partial x_1^2} + \cdots + \frac{\partial^2}{\partial x_d^2} \right) v,$$

for any $v : \Omega \to \mathbb{R}$ smooth enough. Note that we used the notation $x_1, \ldots, x_d$ for the coordinate system in $\mathbb{R}^d$. The above equation $cu - \Delta u = 0$ is known as the *reaction-diffusion* equation, and is among the simplest partial differential equations. It is a linear, scalar, elliptic partial differential equation. For $c = 0$, we recover the Laplace equation (or Poisson's problem), that has many physical interpretations: for instance the unknown u can represent the (steady) temperature induced by the source term f, or the vertical displacement of a thin membrane induced by f, which in this case represents a surfacic density of loads. For $c > 0$ there is a reaction term cu can have a special meaning for some problems in chemistry or biology, and also will be useful when we will study time-evolving PDEs.

Remark also that in Problem (24) there is a second equation on the boundary Γ. This is a *boundary condition*. Since it involves the unknown u directly, it is called a Dirichlet (or essential) boundary condition. There are other types of boundary conditions (Neumann, Robin), that we should study later on. This condition is in general motivated by physical considerations : for instance, if u represents the displacement of a membrane, it means that the membrane is fixed at its boundary ($u = 0$ means 'no vertical displacement'), or if u represents a temperature, it means that the temperature is fixed (to 0 K) on the boundary.

2.2 Weak formulation

The first important idea of the FEM is to reformulate Problem (24) as a *weak problem*, which is equivalent in some specific sense. To this purpose we pick an arbitrary *test function* $v : \Omega \to \mathbb{R}$, and multiply it to the first equation of (24):

$$c\, uv - (\Delta u)v = fv,$$

and we integrate over the whole domain Ω:

$$c \int_\Omega uv - \int_\Omega (\Delta u)v = \int_\Omega fv.$$

We then apply the Green formula to the second term:

$$c \int_\Omega uv + \int_\Omega \nabla u \cdot \nabla v - \int_\Gamma (\nabla u \cdot n)v = \int_\Omega fv.$$

Here n denotes the outward unit normal to the boundary Γ. Finally, we impose an extra condition on the test function v. It should vanish on the boundary Γ. This causes the boundary term above to vanish, and we get finally:

$$c \int_\Omega uv + \int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv.$$

Before we go further, I guess that now, many questions should arise. Why did we do that ? Are the above calculations mathematically meaningful ? Why such a complicated way to proceed ? We will try to provide answers to these questions in the sequel.

First of all, just suppose that u and v are smooth enough so that above calculations are valid. Then, let us try to provide, at least, a precise meaning to the last equation. For this purpose, we need two functional spaces, first let us introduce the *Sobolev space*

$$H^1(\Omega) := \{v \in L^2(\Omega) \mid (\nabla v) \in L^2(\Omega; \mathbb{R}^d)\},$$

where $L^2(\Omega)$ is the (Lebesgue) space of square-integrable (real-valued) functions, and $L^2(\Omega; \mathbb{R}^d)$ is the space of square-integrable vector-valued functions. Take care that we need to take the gradient of v, ∇v in the sense of distributions (or in the weak sense) since functions in $L^2(\Omega)$ are not necessarily differentiable (in the classic sense). We introduce for functions v, w in $H^1(\Omega)$ a scalar product, defined as

$$(v, w)_{1,\Omega} := (v, w)_{0,\Omega} + (\nabla v, \nabla w)_{0,\Omega},$$

where

$$(v, w)_{0,\Omega} := \int_\Omega vw$$

denotes the standard scalar product in $L^2(\Omega)$. The corresponding norm will be denoted by $\|\cdot\|_{1,\Omega}$:

$$\|v\|_{1,\Omega} := \sqrt{(v, v)_{1,\Omega}}$$

for $v \in H^1(\Omega)$. It can be easily proven that, endowed with this scalar product, $H^1(\Omega)$ is a Hilbert space, *i.e.*, a vector space with a scalar product, in which every Cauchy sequence converges (completeness). This is in fact a property inherited from the structure of $L^2(\Omega)$ [14, 15] We define now $H^1_0(\Omega)$ which is the closure of the space of test functions $\mathcal{D}(\Omega)$ in $H^1(\Omega)$, for the norm $\|\cdot\|_{1,\Omega}$. It can be proven that

$$H^1_0(\Omega) := \{v \in H^1(\Omega) \mid v = 0 \text{ on } \Gamma\}.$$

This is a non-trivial result, and an important consequence of the *trace theorem* (see as well the references cited above), since it is not at all obvious at first glance that functions in $H^1(\Omega)$ are properly defined on the boundary Γ. Particularly, the trace theorem needs regularity assumptions on the boundary, and if Γ is a Lipschitz (or strictly) polygonal boundary, this theorem can be applied. The above characterization of $H^1_0(\Omega)$ is much more convenient than its original definition, and will serve to treat the Dirichlet boundary conditions. We can now define the bilinear form

$$a(v, w) := c \int_\Omega vw + \int_\Omega \nabla v \cdot \nabla w,$$

for every functions v and w in $H^1(\Omega)$. We check that indeed, $a(\cdot, \cdot)$ is meaningful when the functions are restricted to this space: applying the Cauchy-Schwarz

inequality, we assess indeed the Lebesgue integrals are finite. As well, we suppose that $f \in L^2(\Omega)$, and for every w in $H^1(\Omega)$, we define the linear form

$$L(w) := \int_\Omega fw,$$

and we check once again that this definition is meaningful.

We have worked enough now to define the *weak formulation* (or variational formulation) of Problem (24), which is

$$\text{Find } u \in H_0^1(\Omega) \text{ such that } a(u,v) = L(v), \text{ for any } v \in H_0^1(\Omega). \tag{25}$$

Remark that the boundary condition $u = 0$ on Γ has been integrated in the definition of the space $H_0^1(\Omega)$ (we say it is an *essential* boundary condition). Since the bilinear form $a(\cdot,\cdot)$ is symmetric, this weak formulation is the first-order optimality condition associated to the minimisation problem

$$\text{Find } u \in H_0^1(\Omega) \text{ that minimizes } \mathcal{J} \text{ over } H_0^1(\Omega), \tag{26}$$

where

$$\mathcal{J}(v) := \frac{1}{2}a(v,v) - L(v).$$

2.3 Lagrange finite element space on simplices

Of course, Problem (24) (or (25)) can be solved analytically only for very special cases, where for instance Ω is square-shaped and the expression of f is simple. Otherwise, there is little hope to find an exact solution (and the situation is even worse for more complex PDEs, with more complex boundary conditions). So we introduce an approximation technique known as the Lagrange Finite Element Method. The first step, as in the previous chapter about ODEs, is to decompose the domain Ω into small, simple cells. This process, in the FEM terminology, is called, *meshing*. So we introduce a *simplicial mesh* as a collection of simplices in $\mathbb{R}^d$ that partition the domain Ω. We call each simplex K and $\mathcal{K}_h$ the collection of all the simplices in Ω (we will precise in a few minutes the meaning of the index h). The mesh needs to satisfy two conditions:

1. It should cover exactly the domain Ω:

$$\overline{\Omega} := \bigcup_{K \in \mathcal{K}_h} K,$$

   where the simplices K are supposed to be closed.

2. The intersection of two distinct simplices K and K' is either empty, or a simplex of dimension lower than d, that is common to both K and K'.

For each simplex K in the mesh, h_K denotes the maximal distance between two points inside K (it is the 'size' of K), and

$$h := \max_{K \in \mathcal{K}_h} h_K$$

is the global *size* of the mesh. What we expect now is that, if the mesh size is small enough, we will recover an accurate enough approximated solution (as for the time-step τ when we studied ODEs).

Before going further, let us be more explicit about these notions for problems in one or two dimensions. First in one dimension ($d = 1$) and for Ω that is an open interval, the different simplices $K_1, \ldots, K_{N+1}$ are simply closed intervals, that cover Ω, and intersect each other only at both ends. We denote

$$x_i = K_i \cap K_{i+1}$$

the *nodes* of the mesh. Remark that if all the simplices are of the same size, we recover a situation similar as in the previous chapter.

Now, for problems in two dimensions, $d = 2$, simplices K are triangles. The intersection between two triangles is either empty, or a common vertex, or a common edge. The vertices of the triangles are also called the *nodes* of the mesh.

Now, from the mesh $\mathcal{K}_h$, we will build a finite dimensional subspace of $H_0^1(\Omega)$, that we will call V_h, as follows. We define, now for every value of the dimension d, the nodes of the mesh as the vertices of the simplices. The *interior nodes* are the nodes that do not belong to the boundary Γ. Let us denote them by $x_1, \ldots, x_N$. Let us define a family of *basis functions*

$$\varphi_1, \ldots, \varphi_N$$

such that

1. Each φ_i is globally continuous on $\overline{\Omega}$ and its restriction to each simplex K is a piecewise affine function (a polynomial function of degree at most 1):

   $$\varphi_i|_K \in \mathbb{P}_1(K),$$

   where $\mathbb{P}_1(K)$ is the space of (multivariate) polynomials of degree 1.

2. Each φ_i is equal to 1 at node x_i, and equal to 0 for the other nodes:

   $$\varphi_i(x_j) = \delta_{ij},$$

   where δ_{ij} is the Kronecker symbol.

From the two above properties, one can check directly that each φ_i has a *compact support* located on the *patch* ω_i of simplices that share the same vertex x_i. So this function vanishes everywhere except in simplices that are in the neighbourhood of the node x_i.

One can also check that the functions (φ_i) are linearly independent. Then we set

$$V_h := \operatorname{span}(\varphi_1, \ldots, \varphi_N).$$

In dimension 1, (φ_i) is the so-called family of hat functions, and in dimension 2, each φ_i is a pyramid of basis ω_i and height 1. One can also check that V_h is the

space of continuous functions that are affine on each simplex K, and that this is a proper subspace of $H_0^1(\Omega)$: because of this property, we talk about *conforming* finite elements. The terminology 'Lagrange' finite elements comes from the fact that it relies on piecewise polynomial Lagrange interpolation between the nodes of the mesh. Remark finally that any function v_h in V_h satisfies automatically the boundary condition

$$v_h = 0 \text{ on } \Gamma.$$

Once the space V_h is built, the discrete problem that serves to approximate Problem (25) simply reads:

$$\text{Find } u_h \in V_h \text{ such that } a(u_h, v_h) = L(v_h), \text{ for any } v_h \in V_h. \tag{27}$$

This is our Finite Element Method. You see, this is easy, we just took Problem (25) and added the subscript h everywhere. Seriously speaking, the main difference with (25) is that now, we find an approximate solution in a simple, finite dimensional, vector space V_h. The main interest of (27) is the following: it can be recasted in matrix form. Indeed, first write

$$u_h = \sum_{j=1}^{N} U_j \varphi_j$$

and set $\mathbf{U}$ the column vector of size N that contains all the scalar unknowns (U_j). Remark, that, due to the definition of the basis (φ_i), we get the relationship

$$U_j = u_h(x_j),$$

for any index j. We say that the unknowns are *nodal* unknowns, since they simply represent the value of u_h at interior nodes. Take $v_h = \varphi_i \in V_h$ for an arbitrary index i, and we get from (27):

$$a(u_h, \varphi_i) = L(\varphi_i).$$

Then use the above expression for u_h:

$$a\left(\sum_{j=1}^{N} U_j \varphi_j, \varphi_i\right) = L(\varphi_i).$$

By linearity

$$\sum_{j=1}^{N} U_j a\left(\varphi_j, \varphi_i\right) = L(\varphi_i). \tag{28}$$

There remains to introduce the (symmetric) *stiffness matrix* $\mathbf{K} = (K_{ij})$ with

$$K_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j \left(= \int_\Omega \nabla\varphi_j \cdot \nabla\varphi_i = K_{ji} \right)$$

as well as the (symmetric) *mass matrix*

$$M_{ij} = \int_\Omega \varphi_i \varphi_j \left(= \int_\Omega \varphi_j \varphi_i = M_{ji} \right)$$

for $i, j = 1, \ldots, N$. We define the *load vector* $\mathbf{F} = (F_i)$ with the relationship

$$F_i = L(\varphi_i)$$

for $i = 1, \ldots, N$. Since there holds, by definition, $a(\varphi_i, \varphi_j) = cM_{ij} + K_{ij}$, formula (28) becomes

$$\sum_{j=1}^{N} (c\ M_{ij} + K_{ij})\, U_j = F_i.$$

So we finally reformulate the discrete Problem (27) in matrix form:

$$(c\ \mathbf{M} + \mathbf{K})\, \mathbf{U} = \mathbf{F}. \tag{29}$$

At this point, we get something that can be treated by a computer. So the whole story about FEM, in practice, consists, in 1) building the matrices $\mathbf{M}$ and $\mathbf{K}$ as well as the vector $\mathbf{F}$ (*assembly procedure*), 2) solving the linear system (29), 3) recovering u_h from the vector $\mathbf{U}$ of nodal unknowns (*postprocessing*).

2.4 Practical implementation

There have been plenty of available finite element codes for decades. They differ in various aspects, but are more or less based on the same structure and ideas. To start with, and to stay within the (now friendly) Octave world, we will play a little bit with the Octave scripts of Martin Gander and Felix Kwok, available here:

`https://www.unige.ch/~gander/siamprograms.php`

If you have a look at all the scripts, and at the detailed explanations, you will realize that solving a PDE with FEM is rather more complicated than solving an ODE. Let us start just giving the main ideas.

1st step: mesh generation. The first thing is to get a mesh $\mathcal{K}_h$ of the domain Ω. A simple and convenient way to handle it is to store it as a matrix of mesh nodes `N`, which contains the coordinates of all the nodes, ordered by columns, and another matrix of simplices `T` that contains the information about connectivity. In fact, each line of `T` contains the indices of the nodes that belong to the same simplex. It can also contain additional information, such as special physical properties, or information about the boundary. To generate a mesh that corresponds to various shapes, simply use the function `NewMesh` that returns, at least, `N` and `T`. There are also functions to plot, refine and smooth a mesh.

2nd step: assembly of the linear system (29). This is the most important part and, in some sense, the very core of every finite element program. In fact,

the most efficient, and now widespread way to compute the matrices involved in (29) is to go through all the simplices, and add the corresponding contribution to the global matrices. This way of thinking comes from the following observation. Recall that, for the stiffness matrix, we have

$$K_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j = \sum_{K\in\mathcal{K}_h} \int_K \nabla\varphi_i \cdot \nabla\varphi_j.$$

So it suffices to compute only integrals on simplices K that involve restrictions on K of the basis functions. Remark that for a wide range of values for i and j, these integral vanish. The precise way to do it is described in details in the corresponding chapter in the book of Martin Gander and Felix Kwok [8].

The function `FEPoisson` does the whole job, and calls subroutines `ComputeElementMassMatrix` and `ComputeElementStiffnessMatrix`, that compute elementary matrices (associated to each simplex) that will contribute into the global matrix.

3rd step: solving of the linear system (29). In general, one just call a library for solving linear systems. In Octave, this is done automatically and efficiently through the instruction \. In general, direct methods such as LU or Cholesky decomposition or variants are used for small systems, and iterative methods such as CG or GMRES are used for large systems [2, 9]. Still the function `FEPoisson` does this part.

4th step: postprocessing. Once $\mathbf{U}$ is recovered, one can provide the user some various information about the solution. At least, it should be visualized. This is the object of the function `PlotSolution`.

2.5 Existence of solutions

Now that we have a first insight into the method and its behavior, let us try to justify it, particularly to justify that u_h is a good approximation of u if h is small enough and if the simplices have a *nice shape* (we will put a precise definition behind these fuzzy words soon). The first step is to make sure that the exact solution u exists, and is unique, so that we approximate something well-defined. For ODEs, we had the Cauchy-Lipschitz (or Picard) Theorem that ensured this fundamental point. Here, we need an equivalent. Before we prove this main theorem of existence, we need a preliminary result, which is known as *Poincaré's inequality.*

Theorem 4 *Let Ω be an open, connected and bounded set in $\mathbb{R}^d$ with polygonal boundary. There exists $c_P > 0$ such that, for every $v \in H^1_0(\Omega)$:*

$$c_P \|v\|_{0,\Omega} \leq \|\nabla v\|_{0,\Omega}. \tag{30}$$

As a consequence, we deduce that $\|\nabla \cdot\|_{0,\Omega}$ is a norm on $H^1_0(\Omega)$, equivalent to the H^1–norm.

Proof. We proceed by contradiction. Suppose that, for every $n \geq 1$, there exists $v_n \in H_0^1(\Omega)$ such that

$$\frac{1}{n}\|v_n\|_{0,\Omega} \geq \|\nabla v_n\|_{0,\Omega}.$$

Define $w_n := v_n/\|v_n\|_{0,\Omega}$, so there holds

$$\frac{1}{n} \geq \|\nabla w_n\|_{0,\Omega},$$

with $\|w_n\|_{0,\Omega} = 1$. As a result, there holds

$$\|w_n\|_{1,\Omega}^2 = \|w_n\|_{0,\Omega}^2 + \|\nabla w_n\|_{0,\Omega}^2 \leq 1 + \frac{1}{n^2} \leq 2,$$

so that the sequence (w_n) is bounded in the H^1–norm. As a result, we use the Rellich-Kondrachov Theorem, and deduce there is a subsequence $(w_{\varphi(n)})$ that converges strongly in $L^2(\Omega)$. Moreover, since

$$\|\nabla w_{\varphi(n)}\|_{0,\Omega} \leq \frac{1}{\varphi(n)},$$

we deduce that $\nabla w_{\varphi(n)}$ tends to 0 for $n \to +\infty$, and thus $(w_{\varphi(n)})$ is a Cauchy sequence in $H_0^1(\Omega)$. Since $H_0^1(\Omega)$ is a Hilbert (complete) space, we deduce that there exists a limit $w^\star$ such that

$$w_{\varphi(n)} \to w^\star \qquad \text{(strongly) in } H_0^1(\Omega).$$

There holds

$$\|\nabla w^\star\|_{0,\Omega} = \lim_{n\to+\infty} \|\nabla w_{\varphi(n)}\|_{0,\Omega} = 0,$$

from which we deduce that $w^\star$ is a constant on Ω. Furthermore, since $w^\star = 0$ on $\partial\Omega$, we conclude that $w^\star = 0$. On the other hand, we have

$$\|w^\star\|_{0,\Omega} = \lim_{n\to+\infty} \|w_{\varphi(n)}\|_{0,\Omega} = 1,$$

which contradicts $w^\star = 0$. To prove the last assertion, note first that we have, by definition, for $v \in H_0^1(\Omega)$:

$$\|\nabla v\|_{0,\Omega} \leq \|v\|_{1,\Omega}.$$

Then we use (30) to prove that

$$\|v\|_{1,\Omega}^2 = \|v\|_{0,\Omega}^2 + \|\nabla v\|_{0,\Omega}^2 \leq \frac{1}{c_P^2}\|\nabla v\|_{0,\Omega}^2 + \|\nabla v\|_{0,\Omega}^2 \leq \left(\frac{1}{c_P^2} + 1\right)\|\nabla v\|_{0,\Omega}^2.$$

This ends the proof. □

Beware that the above Theorem makes use of three non-trivial results in functional analysis:

1. The Rellich-Kondrachov Theorem, that ensures, below the appropriate assumptions on Ω, that the injection $H^1(\Omega) \hookrightarrow L^2(\Omega)$ is compact (particularly, Ω needs to be bounded).

2. The kernel of the (weak) gradient operator ∇ consists of the constant distributions on a connected domain (a result that is difficult to establish when $d \geq 1$).

3. The Trace Theorem, which ensures that $H_0^1(\Omega)$ is the kernel of the trace operator.

There is a direct (simplest ?) proof of Poincaré's inequality, which is obtained thanks to explicit calculation of the Sobolev norms [14, 15]. Nevertheless, the above proof has the advantage that it can be generalized easily for more complex boundary conditions (not only homogeneous Dirichlet). Some generalizations can be found in the litterature with the name of Poincaré-Friedrichs inequalities, Peetre-Tartar lemma, etc.

We now state our main theorem of existence:

Theorem 5 *Let Ω be an open bounded, connected, set in $\mathbb{R}^d$ with polygonal boundary. Let $f \in L^2(\Omega)$, for any $c \geq 0$, there exists one unique solution $u \in H_0^1(\Omega)$ to Problem (25), that is also the unique minimizer of $\mathcal{J}$ on $H_0^1(\Omega)$. Moreover, the following* a priori *estimate holds: there exists $C > 0$ such that*

$$\|u\|_{1,\Omega} \leq C \|f\|_{0,\Omega}. \tag{31}$$

Proof. We use the previous result, Theorem 4, which ensures that the bilinear form $a(\cdot,\cdot)$ defines a scalar product on $H_0^1(\Omega)$, even for $c = 0$, equivalent to the canonical scalar product $(\cdot,\cdot)_{1,\Omega}$. Since $H_0^1(\Omega)$ is a Hilbert space, we can apply the Riesz-Fréchet representation theorem, to assess that there is one unique vector $u \in H_0^1(\Omega)$ such that

$$a(u,\cdot) = L(\cdot).$$

Moreover, there holds, for any v in $H_0^1(\Omega)$:

$$\mathcal{J}(v) - \mathcal{J}(u) = \frac{1}{2} a(v-u, v-u).$$

This ensures that u is the unique minimizer of $\mathcal{J}$ on $H_0^1(\Omega)$. Let us derive finally the *a priori* error estimate. Remember first that

$$\|u\|_{1,\Omega}^2 \leq \left(\frac{1}{c_P^2} + 1\right) \|\nabla u\|_{0,\Omega}^2 \leq \left(\frac{1}{c_P^2} + 1\right) a(u,u) = \left(\frac{1}{c_P^2} + 1\right) L(u).$$

Using the Cauchy-Schwarz inequality, we have also

$$L(u) \leq \|f\|_{0,\Omega} \|u\|_{1,\Omega}.$$

Therefore the estimate (31) is deduced. Remark that the constant C does not depend of c. □

In the above proof, we have established that there exists $\alpha > 0$ such that, for any $v \in H_0^1(\Omega)$, there holds:

$$\alpha \|v\|_{1,\Omega}^2 \leq a(v, v).$$

When such property is satisfied, we say that the bilinear form $a(\cdot, \cdot)$ is *elliptic* (or *coercive*) on $H_0^1(\Omega)$. Having such a property (or a variant) is a fundamental point to prove existence and uniqueness when one looks at PDEs. From the viewpoint of convex analysis, using the ellipticity of $a(\cdot, \cdot)$, we deduce that the functional $\mathcal{J}$ is strongly convex on $H_0^1(\Omega)$ (and we can recover the existence and uniqueness of solutions following this path).

Let us now establish the *strong-weak* equivalence, in other terms let us prove that Problem (24) and Problem (25) are equivalent in some sense.

Proposition 1 *Any solution $u \in H_0^1(\Omega)$ to Problem* (25) *satisfies also the Problem* (24)*, where the first equation has to be understood in the distributional sense. Reciprocally, if $u \in H_0^1(\Omega)$ satisfies* (24) *(in the sense of distributions), it is also a solution to* (25)*.*

Proof. Let $u \in H_0^1(\Omega)$ be the solution to Problem (25), and pick $\varphi \in \mathcal{D}(\Omega)$. There holds

$$\langle \Delta u, \varphi \rangle = -\langle \nabla u, \nabla \varphi \rangle$$

by definition of the Laplace operator, in the sense of distributions (the brackets $\langle \cdot, \cdot \rangle$ denote the duality product in $\mathcal{D}'(\Omega)$). But, since $u \in H_0^1(\Omega)$, ∇u is in fact a regular distribution, which means that

$$\langle \nabla u, \nabla \varphi \rangle = \int_\Omega \nabla u \cdot \nabla \varphi$$

Now, from Problem (25), we deduce

$$\langle \nabla u, \nabla \varphi \rangle = \int_\Omega f\varphi - c \int_\Omega u\varphi$$

so

$$\langle (cu - \Delta u - f), \varphi \rangle = 0,$$

which means that

$$cu - \Delta u - f = 0$$

in $\mathcal{D}'(\Omega)$. Moreover, since $u \in H_0^1(\Omega)$, there also holds $u = 0$ on the boundary Γ. As a result, u is a solution to Problem (24). The other assertion can also be proven using the density of $\mathcal{D}(\Omega)$ into $H_0^1(\Omega)$, as we will show now. Let us suppose that $u \in H_0^1(\Omega)$ satisfies

$$cu - \Delta u = f$$

in $\mathcal{D}'(\Omega)$. Then pick $v \in H_0^1(\Omega)$ and a sequence of test functions (φ_n) that tends to v in the H^1-norm. First, note that, for any $n \geq 0$, we can bound

$$|a(u, v) - a(u, \varphi_n)| \leq (c + 1)\|u\|_{1,\Omega}\|v - \varphi_n\|_{1,\Omega}$$

where we used the Cauchy-Schwarz inequality. Similarly, we bound

$$|L(v) - L(\varphi_n)| \leq \|f\|_{0,\Omega}\|v - \varphi_n\|_{1,\Omega}.$$

There just remains to write

$$a(u,v) - L(v) = (a(u,v) - a(u,\varphi_n)) + (a(u,\varphi_n) - L(\varphi_n)) + (-L(v) + L(\varphi_n)).$$

Since $cu - \Delta u = f$ in $\mathcal{D}'(\Omega)$, the second term $a(u,\varphi_n) - L(\varphi_n)$ vanishes, and we conclude using the triangle inequality, the above bounds, and letting $n \to +\infty$. □

Following the same proof as above, we deduce this next result for the FE solution u_h:

Theorem 6 *Let Ω be an open bounded, connected, set in $\mathbb{R}^d$ with polygonal boundary. Let V_h a finite dimensional space such that $V_h \subset H_0^1(\Omega)$. Let $f \in L^2(\Omega)$, for any $c \geq 0$, there exists one unique solution $u_h \in V_h$ to Problem (27), that is also the unique minimizer of $\mathcal{J}$ on V_h. Moreover, the following* a priori *estimate holds: there exists $C > 0$ such that*

$$\|u_h\|_{1,\Omega} \leq C\|f\|_{0,\Omega}. \tag{32}$$

Moreover, from the ellipticity of $a(\cdot,\cdot)$ on $V_h(\subset V)$, we deduce that, in (29), the global matrix $(c\,\mathbf{M} + \mathbf{K})$ is symmetric, positive and definite (even for $c = 0$).

2.6 Accuracy of the FE approximation

Now we are ensured that both solutions u and u_h exist, and are not multiple, we can say something about the accuracy of the FE approximation, and we will show that the norm

$$\|u - u_h\|_{1,\Omega}$$

will decrease when the size h of the mesh decreases, provided some specific assumption on the geometry of the simplices. We will start with an abstract result, known as Cea's Lemma.

Lemma 2 *Let V be a Hilbert space and V_h a closed vector space such that $V_h \subset V$. Let $a(\cdot,\cdot)$ be a continuous bilinear form on V that satisfies the following ellipticity condition*

$$\alpha\|v\|_{1,\Omega}^2 \leq a(v,v).$$

Let $L(\cdot)$ be a continuous linear form on V. We define u the unique solution of the weak problem

$$a(u,v) = L(v), \qquad \forall v \in V,$$

and u_h the unique solution of the approximate weak problem

$$a(u_h,v_h) = L(v_h), \qquad \forall v_h \in V_h.$$

Then there exists a constant $C > 0$ that depends solely on the continuity and ellipticity constants of $a(\cdot, \cdot)$ such that:

$$\|u - u_h\|_V \leq C \inf_{v_h \in V_h} \|u - v_h\|_V \tag{33}$$

where $\|\cdot\|_V$ is the natural norm on V.

Before we study the proof, let us comment a little bit on this lemma. The first point is that there are very few assumptions on V_h, except that it must be a subspace of V. It has not to be something specific like a finite element space (it does not even need to be a finite dimensional vector space). The second point is that you bound the difference between the continuous and discrete errors by an *approximation error* that quantifies how accurately the space V_h approximates V (the richer the space is, the better the approximation should be).

Proof. Let u and u_h be the continuous and discrete weak solutions. First, pick $v_h \in V_h$ and, noting that $v_h - u_h \in V_h \subset V$, we can write

$$a(u, v_h - u_h) - a(u_h, v_h - u_h) = L(v_h - u_h) - L(v_h - u_h) = 0$$

and thus

$$a(u - u_h, v_h - u_h) = 0.$$

Using the above equality we can write

$$a(u - u_h, u - u_h) = a(u - u_h, u - v_h + v_h - u_h) = a(u - u_h, u - v_h).$$

We then use the ellipticity of $a(\cdot, \cdot)$, combined with the above identity:

$$\alpha \|u - u_h\|_V^2 \leq a(u - u_h, u - u_h) = a(u - u_h, u - v_h).$$

There remains to use the continuity of $a(\cdot, \cdot)$:

$$\alpha \|u - u_h\|_V^2 \leq C_a \|u - u_h\|_V \|u - v_h\|_V,$$

with $C_a > 0$ the continuity constant of $a(\cdot, \cdot)$. As a result we get

$$\|u - u_h\|_V \leq \frac{C_a}{\alpha} \|u - v_h\|_V,$$

and we conclude by setting $C := \frac{C_a}{\alpha}$ and taking the infimum over V_h. □

Now to say something more precise about the (concrete) Lagrange finite element space V_h we defined and used in the previous sections, we need some technical results that allow to bound the error for the Lagrange interpolation in appropriate Sobolev norms. Before this, we need now to introduce a geometric assumption on the mesh, and more precisely on the simplices of the mesh.

Let us consider a mesh sequence $(\mathcal{K}_h)_{h>0}$ index by h (the size of the simplices). We say that this sequence is *shape-regular* if there is a constant $C_R > 0$,

independent of h, such that, for any value of h, and for every simplex $K \in \mathcal{K}_h$, there holds

$$\frac{h_K}{\rho_K} \leq C_R,$$

where h_K is the size of K, as defined previously, and ρ_K is the size of the largest ball included in K.

Remark first that we consider a mesh sequence, and not only one fixed mesh, such that the maximum size of the simplices tend to 0. This definition is traditionnaly attributed to Philippe G. Ciarlet (one of the pioneers of this method). It means, roughly speaking, that the geometry of the simplices can not degenerate when the mesh is refined. For $d = 2$, when simplices are triangles, it can be shown that this is equivalent to the *minimal angle condition*: the smallest angle in each triangle is bounded by below (the triangles can not be "too flat"). For structured mesh, such a geometric condition is always satisfied, but, for unstructured mesh, and complicated geometries, this condition may be hard to achieve in practice (this is still the object of current research). Of course, for $d = 1$, this geometric condition is always satisfied ($h_K = \rho_K \ldots$).

We now introduce the *Lagrange interpolation operator* as follows:

$$\mathcal{I}_h(v) := \sum_{i=1}^{N} v(x_i)\varphi_i,$$

for any $v \in H_0^1(\Omega) \cap \mathcal{C}^0(\overline{\Omega})$. We impose this condition on v since continuity is required to define nodal (ponctual) values of v, and functions in $H_0^1(\Omega)$ are not always continuous (at least when $d \geq 2$). Remark that $\mathcal{I}_h(v)$ belongs to the Lagrange finite element space V_h: this function is simply obtained from v by setting its value at interior nodes, and then doing piecewise polynomial interpolation between nodes.

The following estimate can be proven for the linear Lagrange interpolation:

Lemma 3 *Let $d \leq 3$. Let V_h be the linear Lagrange finite element space built from a shape-regular mesh $\mathcal{K}_h$ of the domain Ω. Then, there is a constant $C > 0$, independent of h (but dependent of the shape regularity constant C_R), such that, for every $v \in H^2(\Omega) \cap H_0^1(\Omega)$, there holds:*

$$\|v - \mathcal{I}_h(v)\|_{1,\Omega} \leq Ch\|v\|_{2,\Omega}.$$

The Sobolev space $H^2(\Omega)$ denotes the subspace of $L^2(\Omega)$ with partial weak derivatives up to order two that belong to $L^2(\Omega)$. It can be proven that functions in $H^2(\Omega)$ are always continuous for $d \leq 3$ (this is a consequence of the Sobolev embedding Theorems). The proof of this result in the general case ($d \geq 2$) is a bit harsh and can be admitted at this stage (for an excellent detailed proof, see for instance [14]). In the case $d = 1$, the proof is much more simple [8].

We end this section with the following result, that bounds the error related to the Finite Element solution u_h, as a function of the mesh size h:

Theorem 7 *Let $d \leq 3$, let $(\mathcal{K}_h)_{h>0}$ be a shape regular sequence of meshes, let V_h be the Lagrange finite element space built upon these meshes. We still denote by u the solution to Problem* (25) *and by u_h the solution to Problem* (27)*. We suppose that $u \in H^2(\Omega) \cap H_0^1(\Omega)$. Then there exists a constant $C > 0$, that depends only on the continuity and ellipticity constants of $a(\cdot, \cdot)$, and on the shape-regularity constant C_R, such that*

$$\|u - u_h\|_{1,\Omega} \leq Ch\|u\|_{2,\Omega}$$

for every $h > 0$.

Proof. We set $V = H_0^1(\Omega)$ and $V_h \subset V$ the Lagrange finite element space. Then the result is obtained using Cea's Lemma 2, setting $v_h = \mathcal{I}_h(u)$, and applying the interpolation estimate in Lemma 3. □

The above result is of fundamental importance. First, it allows to assess, that, for shape-regular meshes, the approximate FE solution u_h tends to the solution u when the mesh size h tends to 0 (when the simplices get smaller and smaller). Moreover, the H^1-norm is a quite strong one, since it takes also into account the spatial derivatives of the solution. Also, it quantifies precisely how much the quality of the approximation is improved when the mesh is refined. If h is divided by two, so is the approximation error $\|u-u_h\|_{1,\Omega}$. We talk of *linear convergence* in this situation. This behavior can be illustrated very accurately in practice in many situations.

2.7 Extra remarks

This presentation was a tiny introduction to this wide topic, and many aspects of the method were hidden, mostly to simplify the presentation, and also because we do not have infinite time to discuss about this topic.

First of all, the reaction-diffusion, or even Poisson's problem, is a good, and common choice to start with, but there are many more partial differential equations, with a great variety of boundary conditions. Sometimes, these more complex problems can be treated by rather straightforward adaptations of the present framework, sometimes not, and in this case, it has been object of intensive research activity.

Also, many type of finite elements, with specific attractive features, have been invented since 1950. There is even a *Table of Finite Elements* designed by Douglas Arnold:

`http://www-users.math.umn.edu/~arnold/femtable/`

The Lagrange finite element is just the simplest one (and still among the most widely used). There are also many methods inspired from the finite element method (just to quote a few recent ones: the eXtended Finite Element Method, the Isogeometric Method, the Virtual Element Method, the Hybrid High Order Method, the cut Finite Element method, etc, etc).

Finally, one can go much further into the mathematical analysis, and prove much more things about the properties of the numerical approximation: estimates in various norms can be derived, as well as, for instance, *a posteriori error estimates* that allow to determine on which cells of the mesh the error is the larger. This allows for instance adaptive mesh refinement, which is a powerful technique to improve the accuracy of the solution.

Concerning all these different topics, there are very good monographs that review most of them in details, particularly [1, 3, 5, 14].

3 Time-dependent PDEs

In this final chapter, we will illustrate how the notions we have studied in the two previous chapters can be combined to deal with time-dependent PDEs, that allow to model unsteady phenomena.

3.1 The heat equation

Let us introduce a simple model of PDE with an evolution in time: the heat equation. As in the previous chapter, we need a domain Ω, which is still an open subset of $\mathbb{R}^d$, $d \geq 1$ (in practice, $d = 1, 2$ or 3). We will suppose again that Ω is non-empty and bounded. We will denote by $\Gamma := \partial\Omega$ the boundary of Ω. For $d = 1$ we will take Ω as a finite union of open intervals. For $d = 2$ (respectively $d = 3$) the boundary Γ will be supposed polygonal (respectively polyhedral). We consider the evolution of the temperature u in the domain Ω, so we want to solve the following heat equation:

$$\frac{\partial u}{\partial t} - \kappa\, \Delta u = f \quad \text{in } \mathbb{R}_*^+ \times \Omega, \quad u = 0 \quad \text{on } \mathbb{R}_*^+ \times \Gamma, \quad u(0) = u_0 \quad \text{in } \Omega. \quad (34)$$

In the above equation (34)

$$u : \mathbb{R}^+ \times \overline{\Omega} \to \mathbb{R}$$

is the unknown solution (the temperature). We will denote by $u(t, x)$ its (scalar) value at time $t \geq 0$ and at point $x \in \overline{\Omega}$. The Laplace operator is still denoted by Δ, whereas the notation $\frac{\partial}{\partial t}$ denotes the partial derivative in time. As for the reaction-diffusion equation, $f : \mathbb{R}_*^+ \times \Omega \to \mathbb{R}$ is a given source term, that models the local production, or absorption, of heat. The coefficient $\kappa > 0$ in front of the Laplace operator is a diffusion coefficient, that models the conductive properties of the material under consideration. Still we impose a Dirichlet condition $u = 0$ on the boundary, for the sake of simplicity. It means that we know the value of the temperature u on Γ. This needs to be changed if a different situation is considered. Conversely to the reaction-diffusion equation, and in the same way as for ODEs, we need an initial condition, and we suppose the temperature is known at the initial time $t = 0$, and equal to u_0, a given function.

Remark that if we seek steady (stationnary) solutions to this equation (if we set $u(t, x) = u(x)$) the partial derivative in time vanishes, and we recover the Poisson's equation we have seen before (reaction-diffusion with c=0). This is the reason why we told you earlier that Poisson's equation can also model the diffusion of heat in a continuous medium.

As for most of the other PDEs, and as for the reaction-diffusion equation, closed-form solutions are only known for very particular domains and/or values of the source term, initial conditions and boundary conditions. Notably, a well-known solution, due to Joseph Fourier, can be obtained using the Fourier transform (this motivated the introduction of the Fourier transform). Nevertheless, this solution is valid only for $\Omega = \mathbb{R}^d$. So for most practical situations, we need to discretize Problem (34) and to find an approximate solution with

our (now) favorite computer. Before doing this, and as for the reaction-diffusion equation earlier, we need first to derive a weak form of this equation.

For this purpose, it will be convenient to adopt the following convention, that will allow us to see the space-time field u as a function of time, only:

$$u : t \mapsto u(t) \in \overline{\Omega}$$

so we can say that

$$u(t)(x) = (u(t))(x) = u(t, x).$$

This said and done, we proceed exactly as for the heat equation, *at a fixed time* $t > 0$. We, once again, pick an arbitrary test function $v : \Omega \to \mathbb{R}$, and multiply it to the first equation of (34):

$$\frac{\partial u}{\partial t} v - \kappa \, (\Delta u) v = f v$$

and we integrate over the whole domain Ω:

$$\int_\Omega \frac{\partial u}{\partial t} v - \kappa \int_\Omega (\Delta u) v = \int_\Omega f v.$$

We then apply the Green formula to the second term:

$$\int_\Omega \frac{\partial u}{\partial t} v + \kappa \int_\Omega \nabla u \cdot \nabla v - \kappa \int_\Gamma (\nabla u \cdot n) v = \int_\Omega f v.$$

Still n denotes the outward unit normal to the boundary Γ. Finally, we impose the test function v to vanish on the boundary Γ:

$$\int_\Omega \frac{\partial u}{\partial t} v + \kappa \int_\Omega \nabla u \cdot \nabla v = \int_\Omega f v.$$

We use the identity

$$\int_\Omega \frac{\partial u}{\partial t} v = \frac{d}{dt} \int_\Omega u v$$

and get the weak formulation corresponding to (34):

$$\frac{d}{dt}(u(t), v)_{0,\Omega} + \kappa (\nabla u(t), \nabla v)_{0,\Omega} = (f(t), v)_{0,\Omega}, \ t > 0, \qquad u(0) = u_0, \quad (35)$$

where $u(t), v \in H_0^1(\Omega)$. Remark that it integrates all the three equations present in (34) (the boundary condition has been included in $H_0^1(\Omega)$). We also made explicit the dependency of each term with t.

Now, let us first establish a result of existence and uniqueness for this equation, as we did in the previous section. This result will allow moreover to precise the functional spaces in which the solution is defined. There are plenty of ways to establish it, for instance using a Galerkin approximation, using the semi-group theory or even using the Ladyzenskaya-Babuska-Brezzi (LBB) theory. See for instance [1, 5, 14]. Here, we follow the path of Grégoire Allaire and state the result proven in his book. We skip the proof, that is a bit long (but not really difficult), and we recommand you study it in the book.

Theorem 8 *Let us suppose that that the initial condition u_0 belongs to $L^2(\Omega)$ and the source term f belongs to $L^2((0,+\infty);L^2(\Omega))$. Then Problem* (35) *admits one unique solution $u \in L^2((0,+\infty);H_0^1(\Omega)) \cap \mathcal{C}([0,+\infty);L^2(\Omega))$.*

The above notation $L^2((0,+\infty);L^2(\Omega))$ means functions square-integrable in the time-interval $(0,+\infty)$, with values in $L^2(\Omega)$ ($L^2((0,+\infty);H_0^1(\Omega))$ means the same, except that the functions take their value in $H_0^1(\Omega)$). The notation $\mathcal{C}([0,+\infty);L^2(\Omega))$ means functions of time with values in $L^2(\Omega)$, that are continuous in time. For the detailed proof, see [1, Theorem 8.2.3] (I just adapted here for a time domain that is unbounded).

The heat equation satisfies many interesting mathematical properties, that have also a physical meaning. We will focus especially here on the following stability estimate:

Proposition 2 *Suppose that $u_0 \in L^2(\Omega)$, and f is identically equal to zero, then the following stability estimate holds, for all $t \geq 0$:*

$$\frac{1}{2}\|u(t)\|_{0,\Omega}^2 + \kappa \int_0^t \|\nabla u(s)\|_{0,\Omega}^2 \, \mathrm{d}s = \frac{1}{2}\|u_0\|_{0,\Omega}^2, \tag{36}$$

where u is the solution to (35).

Proof. Let us take $u_0 \in L^2(\Omega)$, $f = 0$, and $t > 0$. We choose

$$v = u(t) \in H_0^1(\Omega)$$

in (35) and get:

$$\left(\frac{d}{dt}u(t), u(t)\right)_{0,\Omega} + \kappa \left(\nabla u(t), \nabla u(t)\right)_{0,\Omega} = 0.$$

This can be reformulated as

$$\frac{1}{2}\frac{d}{dt}\|u(t)\|_{0,\Omega}^2 + \kappa \|\nabla u(t)\|_{0,\Omega}^2. = 0.$$

Then we obtain (36) by integration in time. □

We deduce from (36) that the solution u remains bounded in the $L^2(\Omega)$-norm. This norm decreases when t increases. The term in κ is a *dissipation* term, that depends on the physical properties of the material. There is dissipation meanwhile the temperature is not homogeneous in the material, and dissipation stops when the temperature gets uniform. We will try to mimic this behavior numerically.

3.2 Finite elements

To discretize Problem (35), we use what is called the *method of lines*: we first semi-discretize the space variables using finite elements, ang we will obtain a

system of ODEs. Then we apply a time-marching scheme to compute the ultimate numerical approximation of these equations, as we did in the first chapter. Other strategies can be designed, for instance one could think about discretizing globally the space-time cylinder in which the equation lives. Nevertheless, the method of lines is among the simplest one and has some advantages in terms of implementation, computation cost, etc.

So we just use the Lagrange finite element space V_h, as defined in the previous chapter, and built from a mesh $\mathcal{K}_h$ of the domain Ω. The semi-discrete (FE) weak form associated to (35) consists in finding

$$u_h : t \mapsto u_h(t) \in V_h$$

for any $t > 0$ that is solution to

$$\frac{d}{dt}(u_h(t), v_h)_{0,\Omega} + \kappa(\nabla u_h(t), \nabla v_h)_{0,\Omega} = (f(t), v_h)_{0,\Omega},\ t > 0,\ u_h(0) = u_{h0}, \quad (37)$$

for any $v_h \in V_h$. The initial condition $u_{h0} \in V_h$ is an approximation of the exact initial condition u_0, that can be obtained, for instance, using Lagrange interpolation ($u_{h0} = \mathcal{I}_h(u_0)$).

Still using the same matrix notations as in previous chapter, this problem can be recasted as :

$$\mathbf{M}\frac{d\mathbf{U}}{dt}(t) + \kappa\mathbf{K}\mathbf{U}(t) = \mathbf{F}(t). \quad (38)$$

As a result, we simply obtain a system of (linear) ODEs, and we find ourselves in the situation of the first chapter. Since $\mathbf{K}$ is symmetric definite positive, Cauchy-Lipschitz Theorem (for instance) ensures the existence and uniqueness of a solution $\mathbf{U} \in \mathcal{C}([0; +\infty); \mathbb{R}^N)$. Now there just remain to apply our favorite time-marching scheme for this system of ODEs. Before this, let us state the following result, which is obtained exactly the same way as in the continuous case:

Proposition 3 *Suppose that $u_{h0} \in V_h$, and f is identically equal to zero, then the following stability estimate holds, for all $t \geq 0$:*

$$\frac{1}{2}\|u_h(t)\|^2_{0,\Omega} + \kappa \int_0^t \|\nabla u_h(s)\|^2_{0,\Omega}\ \mathrm{d}s = \frac{1}{2}\|u_{h0}\|^2_{0,\Omega}, \quad (39)$$

where u_h is the solution to (37).

3.3 Backward Euler

Let us use the backward Euler scheme. We keep the same conventions as in the previous chapter, and, for a time-step $\tau > 0$, and introduce a sequence (u_h^n), such that, for each n :

$$u_h^n \simeq u_h(t_n),$$

with $u_h^0 = u_{h0}$, and, for $n \geq 0$:

$$\frac{1}{\tau}(u_h^{n+1} - u_h^n, v_h)_{0,\Omega} + \kappa(\nabla u_h^{n+1}, \nabla v_h)_{0,\Omega} = (f(t_{n+1}), v_h)_{0,\Omega}, \tag{40}$$

still with $v_h \in V_h$. In matrix form, we can write it as

$$\mathbf{M}\frac{\mathbf{U}^{n+1} - \mathbf{U}^n}{\tau} + \kappa\mathbf{K}\mathbf{U}^{n+1} = \mathbf{F}(t_{n+1}). \tag{41}$$

In this form you may (should) recognize Euler scheme as it was introduced at the very beginning of this course. Remark that Problem (40) can be recasted as

$$\frac{1}{\tau}(u_h^{n+1}, v_h)_{0,\Omega} + \kappa(\nabla u_h^{n+1}, \nabla v_h)_{0,\Omega} = (f(t_{n+1}), v_h)_{0,\Omega} + \frac{1}{\tau}(u_h^n, v_h)_{0,\Omega}, \tag{42}$$

so we realize that u_h^{n+1} is solution to the discretized reaction-diffusion equation presented in the previous chapter (and you understand now the interest of the constant c, with here $c = \frac{1}{\tau}$). In the same way, the linear system that needs to be solved at each time-step is

$$(\mathbf{M} + \tau\kappa\mathbf{K})\,\mathbf{U}^{n+1} = \tau\mathbf{F}(t_{n+1}) + \mathbf{M}\mathbf{U}^n. \tag{43}$$

This is a system identical to what we obtained for the reaction-diffusion equation. Of course, since the matrices involved are the same at each time-step, they can be computed before we enter in the loop that computes the sequence $\mathbf{U}^1, \mathbf{U}^2, \ldots$. We can even precompute, for instance, their Cholesky decomposition before we enter the loop.

Theorems of the previous section ensure existence and uniqueness of the fully discrete solution u_h^n for any value of $n \geq 0$. Theory about ODEs also ensures stability and accuracy of this time-marching scheme. Let us, for the end, provide another argument inspired from the stability estimates we presented previously.

Proposition 4 *Suppose that $u_h^0 \in V_h$, and f is identically equal to zero, then the following stability estimate holds, for all $n \geq 1$:*

$$\frac{1}{2}\|u_h^n\|_{0,\Omega}^2 + \frac{1}{2}\sum_{k=0}^{n-1}\|u_h^{k+1} - u_h^k\|_{0,\Omega}^2 + \kappa\tau\sum_{k=0}^{n-1}\|\nabla u_h^{k+1}\|_{0,\Omega}^2 = \frac{1}{2}\|u_h^0\|_{0,\Omega}^2, \tag{44}$$

where u_h^n is the solution to (40).

Proof. Let us take $u_h^0 \in V^h$, $f = 0$, and $n \geq 1$. We choose

$$v_h = u_h^{n+1} \in V_h$$

in (40) and get:

$$\frac{1}{\tau}(u_h^{n+1} - u_h^n, u_h^{n+1})_{0,\Omega} + \kappa(\nabla u_h^{n+1}, \nabla u_h^{n+1})_{0,\Omega} = 0.$$

We rewrite the first term

$$(u_h^{n+1} - u_h^n, u_h^{n+1})_{0,\Omega} = \|u_h^{n+1}\|_{0,\Omega}^2 - (u_h^n, u_h^{n+1})_{0,\Omega}$$

and use the identity

$$\|u_h^{n+1} - u_h^n\|_{0,\Omega}^2 = \|u_h^{n+1}\|_{0,\Omega}^2 + \|u_h^n\|_{0,\Omega}^2 - 2(u_h^n, u_h^{n+1})_{0,\Omega}$$

so as to obtain

$$\begin{aligned}&(u_h^{n+1} - u_h^n, u_h^{n+1})_{0,\Omega}\\ =&\|u_h^{n+1}\|_{0,\Omega}^2 + \frac{1}{2}\left(\|u_h^{n+1} - u_h^n\|_{0,\Omega}^2 - \|u_h^{n+1}\|_{0,\Omega}^2 - \|u_h^n\|_{0,\Omega}^2\right)\end{aligned}$$

that simplifies into

$$(u_h^{n+1} - u_h^n, u_h^{n+1})_{0,\Omega} = \frac{1}{2}\left(\|u_h^{n+1} - u_h^n\|_{0,\Omega}^2 + \|u_h^{n+1}\|_{0,\Omega}^2 - \|u_h^n\|_{0,\Omega}^2\right).$$

We combine this with the first inequality and get:

$$\frac{1}{2\tau}\left(\|u_h^{n+1} - u_h^n\|_{0,\Omega}^2 + \|u_h^{n+1}\|_{0,\Omega}^2 - \|u_h^n\|_{0,\Omega}^2\right) + \kappa\|\nabla u_h^{n+1}\|_{0,\Omega}^2 = 0,$$

or, equivalently:

$$\frac{1}{2}\|u_h^{n+1}\|_{0,\Omega}^2 + \frac{1}{2}\|u_h^{n+1} - u_h^n\|_{0,\Omega}^2 + \tau\kappa\|\nabla u_h^{n+1}\|_{0,\Omega}^2 = \frac{1}{2}\|u_h^n\|_{0,\Omega}^2.$$

Then we obtain (44) by summation. □

There are some comments about this result: then it allows to check that the backward Euler scheme is unconditionnaly stable, since the L^2-norm of u_h^n remains bounded by the initial condition whatever is the value of n and whatever is the value of τ. We also recover a "physical dissipation" that mimics the integral term in the estimate (36). There is also an extra dissipation term, which is purely numerical, and comes from the scheme : we say that the scheme is dissipative. It dissipates artificial energy. In some situations (let us say, for large κ and τ small enough) this dissipation is neglectible, but sometimes it is not really desired since it deteriorates the accuracy of the numerical solution.

3.4 Extra remarks

There are still many aspect to discuss about time-evolution PDEs: the general process to solve other PDEs such as the wave equation or the Schrödinger equation is the same, but care has to be taken to chose the appropriate finite element method and time discretization, so as to recover appropriately some interesting properties, such as energy conservation. Also, some PDEs such as the transport equation or some more complex first-order hyperbolic PDEs need a very specific treatment in order to recover appropriate solutions.

4 Perspectives

This class is just an introduction to some basic facts when one needs to solve numerically some ODEs or PDEs related to physical models. Many topics have not been discussed. For instance, I did not talk about non-linear PDEs, which require some specific treatment, and specific care, to be solved appropriately. Another interesting aspect is to use some modern finite element software, such as FreeFEM++ (`http://www3.freefem.org/`), GetFEM++ (`http://getfem.org/`) or FEniCS (`https://fenicsproject.org/`), that include a high-level language related to weak formulations. For instance, you can try to use FreeFEM++ here:

`https://freefem.org/tryit`

$\star$

References

[1] G. Allaire, *Numerical analysis and optimization*, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 2007. An introduction to mathematical modelling and numerical simulation, Translated from the French by Alan Craig.

[2] G. Allaire and S. M. Kaber, *Numerical linear algebra*, vol. 55 of Texts in Applied Mathematics, Springer, New York, 2008. Translated from the 2002 French original by Karim Trabelsi.

[3] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 15 of Texts in Applied Mathematics, Springer, New York, third ed., 2008.

[4] J. C. Butcher, *Numerical methods for ordinary differential equations*, John Wiley & Sons, Ltd., Chichester, third ed., 2016. With a foreword by J. M. Sanz-Serna.

[5] A. Ern and J.-L. Guermond, *Theory and practice of finite elements*, vol. 159 of Applied Mathematical Sciences, Springer-Verlag, New York, 2004.

[6] M. J. Gander, *50 years of time parallel time integration*, in Multiple shooting and time domain decomposition methods, vol. 9 of Contrib. Math. Comput. Sci., Springer, Cham, 2015, pp. 69–113.

[7] ——, *Time parallel time integration*. Lecture Notes. University of Geneva. Available here: `https://www.unige.ch/~gander/poly.pdf`, 2018.

[8] M. J. Gander and F. Kwok, *Numerical analysis of partial differential equations using Maple and MATLAB*, vol. 12 of Fundamentals of Algorithms, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2018.

[9] W. Gander, M. J. Gander, and F. Kwok, *Scientific computing*, vol. 11 of Texts in Computational Science and Engineering, Springer, Cham, 2014. An introduction using Maple and MATLAB.

[10] W. Gautschi, *Numerical analysis*, Birkhäuser Boston, Inc., Boston, MA, 1997. An introduction.

[11] E. Hairer, C. Lubich, and G. Wanner, *Geometric numerical integration*, vol. 31 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 2002. Structure-preserving algorithms for ordinary differential equations.

[12] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I*, vol. 8 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 1993. Nonstiff problems.

[13] E. Hairer and G. Wanner, *Solving ordinary differential equations. II*, vol. 14 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 1996. Stiff and differential-algebraic problems.

[14] P.-A. Raviart and J.-M. Thomas, *Introduction à l'analyse numérique des équations aux dérivées partielles*, Collection Mathématiques Appliquées pour la Maîtrise. [Collection of Applied Mathematics for the Master's Degree], Masson, Paris, 1983.

[15] F.-J. Sayas, T. S. Brown, and M. E. Hassell, *Variational techniques for elliptic partial differential equations*, CRC Press, Boca Raton, FL, 2019. Theoretical tools and advanced applications.

[16] E. Süli and D. F. Mayers, *An introduction to numerical analysis*, Cambridge University Press, Cambridge, 2003.