Numerical methods for differential equations
Lecture notes

Łukasz Płociniczak[*]

April 2, 2020

Contents

[*]Faculty of Pure and Applied Mathematics, Wrocław University of Science and Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland

1 Introduction

The aim of this course is to develop a methodology of devising, analysing, choosing, and implementing numerical schemes used in approximation of a general class of differential equations. These include either ordinary (ODEs),

$$y^{(n)}(t) = f(t, y, y', ..., y^{(n-1)}), \tag{1.1}$$

or partial differential equations (PDEs)

$$F(x_i, u, u_{x_i}, u_{x_i,x_j}, ...) = 0, \tag{1.2}$$

linear or nonlinear, scalar or vector. Differential equations are the one of the most robust tools of modelling various natural, medical, and industrial phenomena. Many of them cannot be solved in an closed analytical form and thus one has to use some approximation technique such as asymptotic theory (analytical) or numerical methods. Both of these are complementary and help each other in a tremendous way helping humanity to understand reality around us.

In this lecture we will develop mostly finite difference methods since they can be applied to virtually any type of equation. However, we will discuss their applicability and note the fact that in some areas they fail do provide a useful discretization. We will use this opportunity to introduce some powerful methods such as Finite Volumes or Finite Elements.

Using numerical methods has a lot of art in it. Different problems require different methods, discretizations, and implementations. Usually it is not wise to perform blindly without any understanding of the underlying nature of the analysis. Some methods are better than other but only in some specific situations. Comprehending these subtle relationships between the nature of differential equation and a numerical method is a crucial skill for numerical analysts and applied mathematicians.

1.1 Literature

There is a vast amount of good numerical analysis books. We can mention the following very good items on which this lecture borrowed some material.

1. Hairer, Nørsett, and Wanner, Solving Ordinary Differential Equations
2. LeVeque, Finite Difference Methods for Ordinary and Partial Differential Equations
3. LeVeque, Numerical Methods for Conservation Laws
4. Larson, Thomee, Partial Differential Equations with Numerical Methods

2 Finite differences and Euler methods

2.1 Finite differences

Differential equations are constructed with derivatives and in numerical analysis we would like to find a way of approximating them numerically. Since derivatives are limits of difference quotients we have to find a way of representing these limits with a finite precision.

Let $y = y(t)$ be a function of variable. In what follows we will always assume sufficient smoothness of y unless otherwise stated. Then, of course

$$y'(x) = \lim_{h \to 0^+} \frac{u(x+h) - u(x)}{h}, \tag{2.1}$$

is the definition of the derivative. Since the above requires a limit passage we can truncate it to obtain a finite precision approximation in the form of *forward difference*

$$\delta_+ y(x) := \frac{y(x+h) - y(x)}{h}, \tag{2.2}$$

or *backward difference*

$$\delta_- y(x) := \frac{y(x) - y(x-h)}{h}, \tag{2.3}$$

where $h > 0$. We can also take the average of the two preceding operators and form the *centred difference*

$$\delta_0 y(x) := \frac{1}{2}\left(\delta_- y(t) + \delta_+ y(t)\right) = \frac{y(x+h) - y(x-h)}{2h}, \quad h > 0. \tag{2.4}$$

It should be clear that the centred approximation is usually better that either backward or forward difference.

Example. Let $y(x) = \sin x$. We would like to approximate $y'(x) = \cos x$. We have

$$\delta_+ y(x) = \frac{\sin(x+h) - \sin x}{h} = \cos\left(x + \frac{h}{2}\right)\frac{\sin\frac{h}{2}}{\frac{h}{2}} = \cos x + \frac{1}{2}\sin x\, h + O(h^2), \tag{2.5}$$

where we used the Taylor expansion when $h \to 0^+$. We can thus see that $D_+ y(x) - \cos x = (\sin x)\frac{h}{2} + O(h^2)$. Similarly, for the centred difference we have

$$\delta_0 y(x) = \frac{\sin(x+h) - \sin(x-h)}{2h} = \cos x - \frac{1}{6}\cos x\, h^2 + O(h^4) \quad \text{as} \quad h \to 0^+. \tag{2.6}$$

We can now see that if $h < 1$ then $h^2 \ll h$ and the error that we make when approximating the derivative with δ_0 should be much smaller than with the δ_+ (apart maybe from points of the form $x = n\pi$).

We see that various approximations of the derivatives are not equivalent and some of them are more accurate than the other. In many places below the error of some method will be proportional to some power of the step h, i.e.

$$E(h) \approx Ch^p, \quad p > 0, \tag{2.7}$$

where $C > 0$ is a constant independent on h. A useful way of visualizing this error during simulations will be to present it in a log-log scale because of the fact that

$$\ln E(h) \approx p \ln h + \ln C. \tag{2.8}$$

Therefore, $\ln E(h)$ is a linear function of $\ln h$ which slope is precisely equal to the *order* of the method p. We will use this tool many times.

The above example has indicated a useful method of obtaining estimates for the error of approximation - the Taylor expansion.

Defition 1. *The **truncation error** of a finite difference operator δ is*

$$\delta y(x) - y'(x). \tag{2.9}$$

When

$$|\delta y(x) - y'(x)| \le C h^p, \quad h > 0, \quad p > 0, \tag{2.10}$$

where $C > 0$ is independent of h, the number p is the order *of approximation.*

Of course, in a similar manner we can construct approximations of higher derivatives. Calculating explicitly

$$\delta_+ y(x) - y'(x) = \frac{y(x+h) - y(x)}{h} - y'(x) = \frac{1}{2} h y''(x) + \frac{1}{6} h^2 y'''(x) + O(h^3) \quad \text{as} \quad h \to 0^+, \tag{2.11}$$

and

$$\delta_0 y(x) - y(x) = \frac{1}{6} h^2 y'''(x) + O(h^4) \quad \text{as} \quad h \to 0^+. \tag{2.12}$$

Therefore, we see that δ_+ is of the first, while δ_0 of second order. We will see that centred approximations have larger order than the one-sided due to cancellations, that is the terms in the expansions of $y(x+h) - y(x-h)$ cancel out.

One may ask whether we have been lucky in finding formulas for finite difference approximations. There are some systematic ways of deriving some useful operators. The simplest of them is the method of undetermined coefficients which utilises Taylor expansion. Suppose we want to find an approximation of $y^{(k)}(x^*)$ based on knowledge of y at points $\{x_i\}_1^n$ where $n \ge k+1$. This set of points is called the *stencil*. For every $i = 1, ..., n$ we have

$$y(x_i) = y(x^*) + y'(x^*)(x_i - x^*) + \frac{1}{2} y''(x^*)(x_i - x^*)^2 + ... + \frac{1}{k!} y^{(k)}(x^*)(x_i - x^*)^k + ... \tag{2.13}$$

We thus want a linear combination of the above to approximate a given derivative

$$\sum_{i=1}^{n} c_i y(x_i) = y^{(k)}(x^*) + O(h^p), \tag{2.14}$$

where $p > 0$ has to be as large as possible. Now, plugging (2.13) we arrive at

$$y^{(k)}(x^*) = \sum_{i=1}^{n} c_i \sum_{j=0}^{n-1} \frac{y^{(j)}(x^*)}{j!} (x_i - x^*)^j = \sum_{j=0}^{n-1} \frac{y^{(j)}(x^*)}{j!} \sum_{i=1}^{n} c_i (x_i - x^*)^j \tag{2.15}$$

Of course, the $j-$sum is terminated at $j = n$ since we would like to obtain a well-posed linear system. Now, the both sides of the above equation are equal if and only if

$$\frac{1}{j!}\sum_{i=1}^{n} c_i(x_i - x^*)^j = \begin{cases} 1, & j = k \\ 0, & j \neq k \end{cases} \qquad j = 0, ..., n-1. \tag{2.16}$$

If all x_i are distinct, the above is just the Vandermonde's system. Unfortunately, for large n the resulting linear system is badly conditioned and hence, difficult to accurately solve numerically. There are however some useful bypasses over this problem.

Example. We will find a finite difference approximation to $y'(x)$ build on a stencil x, $x-h$, and $x-2h$. We write

$$\delta_2 y(x) = ay(x) + by(x-h) + cy(x-2h). \tag{2.17}$$

If we expand two right terms in the above and collect similar expressions we arrive at

$$\delta_2 y(x) = (a+b+c)y(x) - (b+2c)hy'(x) + \frac{1}{2}(b+4c)h^2 y''(x) - \frac{1}{6}(b+8c)h^3 y'''(x) + O(h^4) \quad \text{as} \quad h \to 0^+. \tag{2.18}$$

Now, since our approximation concerns y' we take

$$a+b+c = 0, \quad b+2c = -\frac{1}{h}, \quad b+4c = 0, \tag{2.19}$$

which has a solution

$$a = \frac{3}{2h}, \quad b = -\frac{2}{h}, \quad c = \frac{1}{2h}. \tag{2.20}$$

Therefore, our finite difference has the form

$$\delta_2 y(x) = \frac{1}{2h}\left(3y(x) - 4y(x-h) + y(x-2h)\right). \tag{2.21}$$

The error of the approximation can be readily calculated from the expansion yielding

$$\delta_2 y(x) - y'(x) = \frac{1}{12}h^2 y'''(\xi), \tag{2.22}$$

where ξ is some point. The approximation is thus second order accurate.

2.2 Euler methods

Now, we are able to proceed to design of some numerical methods for ODEs. We start from the simplest ones - Euler schemes. Suppose we would like to solve

$$y'(x) = f(x, y), \quad y(0) = y_0. \tag{2.23}$$

Let us introduce the grid $x_n = nh$ where $h > 0$ is the grid spacing. We can use any finite difference to approximate the derivative on the left-hand side. When we use $\delta_\pm$ we obtain

$$y'(x_n) = \delta_\pm y(x_n) + R_\pm = f(x_n, y(x_n)). \tag{2.24}$$

All numerical methods are based on truncating the remainder $R_\pm$. If y_n denotes the numerical approximation to $y(x_n)$ (notice that these are usually different quantities!), we obtain the *Euler forward* method

$$y_n = y_{n-1} + hf(x_{n-1}, y_{n-1}), \tag{2.25}$$

and *Euler backward* method

$$y_n = y_{n-1} + hf(x_n, y_n). \tag{2.26}$$

The difference is basically in the point at which the function f is evaluated. Note also that the forward method is *explicit*, i.e. the next step is calculated directly from the previous ones, while backward method is *implicit*, i.e. the next step additionally requires solving a nonlinear equation $z = y_{n-1} + f(x_n, z)$. This increased computational cost has its merits and advantages as we will see in the sequel.

Immediately, we ask a question how accurate are the methods (2.25) and (2.25)? Do they approximate the exact solution of (2.23) arbitrarily good when $h \to 0^+$? Naively thinking, we can expect that since $\delta_\pm$ is a first order operator, the Euler methods should also be first order accurate. This appears to be true however, is not that simple. When integrating a differential equation the error in each step is accumulated. We have to ascertain whether it not accumulates too much. This is the problem of *convergence*.

On the other hand, a numerical method can be convergent but to a different solution than the original ODE's. This is the problem of *consistency*. We can rigorously define the relevant terms.

Defition 2. *A* local truncation error *(LTE) is the remainder of the numerical scheme when y_n is replaced with the exact solution of the corresponding ODE, that is $y(x_n)$. If LTE vanishes as $h \to 0^+$ the method is said to be* ***consistent****.*

A consistent numerical method approximates the relevant differential equation. For forward Euler method we have

$$\begin{aligned} \mathrm{LTE} &= \frac{y(x_n) - y(x_{n-1})}{h} - f(x_{n-1}, y(x_{n-1})) = y'(x_n) + \frac{1}{2}hy''(x_n) + O(h^3) - f(x_{n-1}, y(x_{n-1})) \\ &= \frac{1}{2}hy''(x_n) + O(h^3) \quad \text{as} \quad h \to 0^+, \end{aligned} \tag{2.27}$$

since $y'(x_n) = f(x_n, y(x_n))$. Therefore, Euler method is a consistent method. A consistent method however, may not be convergent to the exact solution since the error can accumulate to fast.

Example. Consider a numerical method

$$y_{n+1} = y_{n-1} - 2hy_n, \quad y_0 = 1, \quad y_1 = 1. \tag{2.28}$$

Computing the local truncation error we have

$$\mathrm{LTE} = \frac{y(x_{n+1}) - y(x_{n-1})}{2h} + y(x_n) = \frac{1}{6}h^2 y'''(\xi_n), \tag{2.29}$$

therefore the method is consistent with the following ODE

$$y' = -y, \quad y(0) = 1, \tag{2.30}$$

which has a solution $y(x) = e^{-x}$. Now, we recall that a linear recurrence can be solved by looking for power function solutions $y_n = Cr^n$ for some C and r. Plugging this ansatz we obtain from (2.28)

$$Cr^{n+1} + 2Chr^n - r^{n-1} = 0. \tag{2.31}$$

Cancelling yields

$$r^2 + 2hr - 1 = 0, \tag{2.32}$$

and hence

$$r_\pm = -h \pm \sqrt{1 + h^2}. \tag{2.33}$$

Hence $|r_-| > 1$ and the recurrence is divergence for a general initial condition.

The above method is called the *leap-frog* since it jumps two steps ahead. Notice that although the truncation error is of second order, the method is useless due to its lack of convergence. Now, we state what we will mean by a convergent numerical method.

Defition 3. *Fix $x \in \mathbb{R}$. A numerical method is* convergent *with order $p > 0$ if*

$$|y(x) - y_n| \leq Ch^p \quad \textit{as} \quad nh \to x \quad \textit{and} \quad h \to 0^+, \tag{2.34}$$

for some constant $C > 0$ independent of n and h.

Therefore, a convergent method yields an arbitrarily accurate approximation of $y(x)$ when the grid is refined with nh converging to x (for example take $h = x/n$ and $n \to \infty$). The convergence proofs are usually difficult to obtain especially for PDEs. In due course we will indicate the various issues appearing in them.

Theorem 1. *Let $y = y(x)$, $x \in [0, X]$ be a solution of (2.23) with $f(x, y)$ being continuosly twice differentiable with respect to y variable. Then, both of Euler methods (2.25)-(2.26) are first order convergent.*

Proof. We will proceed by induction and without any loss of generality we consider only the forward case. Assume that $y_0 = y(0)$ (this is a slight simplification since computing y_0 always contains a round-off error). Define the convergence error

$$e_n := y(x_n) - y_n. \tag{2.35}$$

Then, from (2.25) and Taylor expansion the first term can be bounded as follows

$$|e_1| = |y(x_1) - y_1| = |y(x_0) + y'(x_0)h + \frac{1}{2}y''(\xi_0)h^2 - y_0 - hf(x_0, y_0)| \leq \frac{1}{2}\max_{x \in [0,X]} |y''(x)|h^2 =: \tau, \tag{2.36}$$

since the initial conditions are the same and $y'(x_0) = f(x_0, y(x_0))$. Then, the inductive step can be carried over similarly

$$\begin{aligned} |e_n| &= |y(x_{n-1}) + hy'(x_{n-1}) + \frac{1}{2}h^2 y''(\xi_{n-1}) - y_{n-1} - hf(x_{n-1}, y_{n-1})| \\ &\leq |e_{n-1}| + |f(x_{n-1}, y(x_{n-1})) - f(x_{n-1}, y_{n-1})|h + \tau, \end{aligned} \tag{2.37}$$

where we used the fact that $y'(x_{n-1}) = f(x_{n-1}, y_{n-1})$. Since f is Lipschitz with respect to the second variable we have $|f(x_{n-1}, y(x_{n-1})) - f(x_{n-1}, y_{n-1})| \leq L|e_{n-1}|$ for some $L > 0$. Therefore,

$$|e_n| \leq (1 + Lh)|e_{n-1}| + \tau. \tag{2.38}$$

Proceeding inductively, we obtain

$$|e_n| \leq (1+Lh)^2|e_{n-2}|+(1+Lh)\tau \leq \ldots \leq (1+Lh)^{n-1}|e_1|+\tau\sum_{i=0}^{n-2}(1+Lh)^i \leq \tau\sum_{i=0}^{n-1}(1+Lh)^i. \tag{2.39}$$

The last sum is geometric and hence

$$|e_n| \leq \tau\sum_{i=0}^{n-1}(1 + Lh)^i = \frac{\tau}{Lh}((1 - Lh)^n - 1) = \frac{1}{2L}\max_{x\in[0,X]}|y''(x)|\left(\left(1 - \frac{Lnh}{n}\right)^n - 1\right)h. \tag{2.40}$$

Now, since $(1 + 1/n)^n \leq e$ and $nh \leq X$ we have

$$|e_n| \leq \frac{1}{2L}\max_{x\in[0,X]}|y''(x)|\left(e^{LX} - 1\right)h = Ch. \tag{2.41}$$

This ends the proof. □

Euler methods are the simplest ones to solve ODEs and many times are not very useful due to their low accuracy. However, there are many important applications of using them in numerically solving PDEs since they are very easy to implement in time-advancement of a scheme. Moreover, sometimes it is not desirable to use high order methods due to unstable behaviour about which we will have much more to say.

3 Runge-Kutta methods

There are several ways of generalizing Euler methods to arbitrary high orders. However, as we have seen some of them may not be convergent and it is crucial to develop a scheme of deriving reliable high order methods. One of them are so-called multistage methods aka *Runge-Kutta methods*. They are widely known and used in many commercial ODE integrators. We will take a closer look on possibilities to find high order versions of Euler methods.

3.1 The overall idea

A forward Euler method can also be found from simple Taylor series. Let $y = y(x)$ satisfy the following ODE

$$y' = f(x, y), \quad y(0) = y_0. \tag{3.1}$$

Then, with the usual grid $x_n = nh$ for $h > 0$ we have

$$y(x_{n+1}) = y(x_n) + y'(x_n)h + \frac{1}{2}y''(x_n)h^2 + ... \tag{3.2}$$

Using the ODE and dropping $O(h^2)$ terms we reobtain the Euler forward method $y_{n+1} = y_n + hf(x_n, y_n)$. A natural generalization is to keep further terms in the expansion. However, in subsequent terms we need an expression to calculate higher derivatives of y. This is relatively straightforward since we know (3.1). For example,

$$y''(x) = f_x(x, y(x)) + y'(x)f_y(x, y(x)) = f_x(x, y(x)) + f(x, y(x))f_y(x, y(x)). \tag{3.3}$$

This approach is usually very cumbersome and unpractical. It requires calculating derivatives of arbitrary functions which is computationally expensive. This is especially troublesome for systems of equations. The Taylor series methods are not frequently used in practice however, some of its modifications are useful in deriving schemes for hyperbolic PDEs.

Another possibility is to increase the order by calculating the derivatives of f in some approximate *multistage* way. The relevant ideas were developed at the beginning of the XX century by German mathematicians Carl Runge and Wilhelm Kutta. To introduce the notion we start from autonomous equations of the form

$$y' = f(y). \tag{3.4}$$

Consider the following two-stage RK method

$$\begin{aligned} y^* &= y_n + \frac{1}{2}hf(y_n), \\ y_{n+1} &= y_n + hf(y^*), \end{aligned} \tag{3.5}$$

which can be combined to yield a scheme

$$y_{n+1} = y_n + hf\left(y_n + \frac{1}{2}hf(y_n)\right). \tag{3.6}$$

The first step is just an Euler forward approximation to the value of $u_{n+1/2}$ while the second yields the full step. It can also be understood in a different, more historical, way. Le us integrate our ODE from x_n to x_{n+1}

$$y(x_{n+1}) = y(x_n) + \int_{x_n}^{x_{n+1}} f(y(t))dt. \tag{3.7}$$

We can use any quadrature to compute the resulting integral. For example, if we approximate the are under $f(y(t))$ by a rectangle with width $x_{n+1} - x_n = h$ and height calculated at the midpoint $(x_{n+1} + x_n)/2 = x_n + \frac{1}{2}h$ we obtain (the midpoint quadrature is of second order)

$$y_{n+1} = y_n + hf\left(y\left(x_n + \frac{1}{2}h\right)\right). \tag{3.8}$$

Notice that we have truncated the remainder of the quadrature and used the subscripts to indicate the numerical approximation. Further, if we expand

$$y\left(x_n + \frac{1}{2}h\right) = y(x_n) + \frac{1}{2}hy'(x_n) + O(h^2) = y(x_n) + \frac{1}{2}hf(y(x_n)) + O(h^2) \quad \text{as} \quad h \to 0^+ \tag{3.9}$$

and truncate the remainder we obtain (3.6).

Runge asked whether it is possible to generalize the above approach to the general ODE (3.1). Before we do that, we will prove that (3.6) is second order accurate. To this end we will compute the local truncation error

$$\text{LTE} = \frac{y(x_{n+1}) - y(x_n)}{h} - f\left(y(x_n) + \frac{1}{2}hf(y(x_n))\right). \tag{3.10}$$

Since, $y'(x) = f(y(x))$ for any x we first have

$$\begin{aligned} f\left(y(x_n) + \frac{1}{2}hf(y_n)\right) &= f\left(y(x_n) + \frac{1}{2}hy'(x_n)\right) \\ &= f(y(x_n)) + \frac{1}{2}hf'(y(x_n))y'(x_n) + O(h^2) \quad \text{ash} \to 0^+. \end{aligned} \tag{3.11}$$

Now, because of the fact that $y''(x) = y'(x)f'(y(x))$ we can write

$$f\left(y(x_n) + \frac{1}{2}hf(y_n)\right) = y'(x_n) + \frac{1}{2}hy''(x_n) + O(h^2). \tag{3.12}$$

Therefore, the above approximates the Taylor expansion of y' up to the second order. Moreover,

$$\begin{aligned} \text{LTE} &= y'(x_n) + \frac{1}{2}hy''(x_n) + \frac{1}{6}h^2y'''(x_n) + O(h^3) - \left(y'(x_n) + \frac{1}{2}hy''(x_n) + O(h^2)\right) \\ &= \frac{1}{6}h^2y'''(x_n) + O(h^3) \quad \text{as} \quad h \to 0^+. \end{aligned} \tag{3.13}$$

Therefore, the method is of second order. The multistage nature of various RK methods is based on successively approximating derivatives of f by stepping forward in some fractional distances. A generalization of the above method to the fully general case (3.1) is the following *mid-point* method

$$\begin{aligned} y^* &= y_n + \frac{1}{2}hf(x_n, y_n) \\ y_{n+1} &= y_n + hf\left(x_n + \frac{h}{2}, y^*\right). \end{aligned} \tag{3.14}$$

This is a second order method what can be checked by computing the LTE. However, the computations are cumbersome.

3.2 General RK methods and Butcher's tableaux

The calculations leading to some higher order RK methods soon become very cluttered. There is however, a very useful method of organizing them. A general s-stage RK method has the form

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i, \tag{3.15}$$

where

$$k_i = f\left(x_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij} k_j\right). \tag{3.16}$$

The consistency requires that we have

$$\sum_{j=1}^{s} a_{ij} = c_i, \quad \sum_{j=1}^{s} b_j = 1. \tag{3.17}$$

The constants k_i are approximations to the derivatives of f at time steps weighted by c_i.

If it happens that $a_{ij} = 0$ for all $j \geq i$ the method is *explicit*. Otherwise, it is *implicit*. We will focus only on explicit methods since the other type can be prohibitively expensive to implement. There is one widely used exception, that is the so-called *diagonally implicit RK* methods (DIRK) in which $a_{ij} = 0$ for $j > i$.

If a method satisfies (3.17) it is guaranteed that it will be at least first order accurate. However, some additional conditions are required for higher accuracy. For example, second order requires

$$\sum_{j=0}^{s} b_j c_j = \frac{1}{2}. \tag{3.18}$$

The number of conditions to even higher order grows exponentially and some great amount of work has been done in the last century to find the best methods. More specifically, one can prove that if an explicit method has order p then the number of stages $s \geq p$ or $s \geq p+1$ when $p \geq 5$. We do not know, however, whether these bounds are sharp. There is an open problem that given order p determine the minimal number of stages required to construct a RK method. Known values of minimal stage method for a given order are given in Tab. 1.

p	1	2	3	4	5	6	7	8
s	1	2	3	4	6	7	9	11

Table 1: Minimal sharp number of stages s required to devise a RK method with order p.

Method	Euler	Heun	Kutta	RK4
Order	1	2	3	4
Tableau	$\begin{array}{c\|c} 0 & \\ 1 & 0 \\ \hline & 1 \end{array}$	$\begin{array}{c\|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$	$\begin{array}{c\|ccc} 0 & 0 & & \\ \frac{1}{2} & \frac{1}{2} & 0 & \\ 1 & -1 & 2 & 0 \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}$	$\begin{array}{c\|cccc} 0 & 0 & & & \\ \frac{1}{2} & \frac{1}{2} & 0 & & \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$

Table 2: Some of the most popular RK methods.

There is a useful way of organizing coefficients of RK. It goes by the name of Butcher's tableau. It has the form

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array} \tag{3.19}$$

For example, the second order mid-point method (3.14) is represented by (blank place is understood as a 0)

$$\begin{array}{c|cc} 0 & 0 & \\ \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & 0 & 1 \end{array} \tag{3.20}$$

Other common tableaux are presented in Tab. 2. The table can of course be expanded indefinitely. Note that RK4 is very popular since it is easily implemented and gives very accurate fourth order approximation.

3.3 Embedded methods

In many scientific packages ODE integrators are not based upon a fixed step methods. They rather adjust the temporal grid spacing to achieve a numerical solution valid to a given error. Practical way of estimating the error is to use two methods of different order and then to compare the results. This strategy, however, when used naively can double the computational costs and cloak the advantage of a adaptive step functionality. Usually one of the most expensive parts of all ODE methods is evaluation of the function f. It is thus reasonable to try to find two methods that utilize the same set of k_i parameters and producing approximations of different order. These are *embedded methods*.

Fix the order p and suppose we are given two methods with the above property, say y_n is of order p while $\widehat{y}_{n+1}$ of order $p+1$

$$\begin{aligned} y_{n+1} &= y_n + h\sum_{j=1}^{s} b_i k_i, \\ \widehat{y}_{n+1} &= y_n + h\sum_{j=1}^{s} \widehat{b}_i k_i, \end{aligned} \tag{3.21}$$

as $h \to 0^+$. Then, the error

$$e_{n+1} := \widehat{y}_{n+1} - y_{n+1} = h\sum_{j=1}^{s} (\widehat{b}_j - b_j) k_i \approx O(h^p). \tag{3.22}$$

Therefore, we have a straightforward way of estimating the error of the method. It can be then be used to compose an adaptive step algorithm. To this end suppose that $e_{1,2}$ are errors defined above calculated with a time step $h_{1,2}$. Moreover, for sufficiently small steps we should have $e_{1,2} = ch_{1,2}^p$ (we assume that c is essentially the same in both methods). Then,

$$\frac{e_1}{e_2} = \left(\frac{h_1}{h_2}\right)^p. \tag{3.23}$$

Therefore, if TOL represents the desired tolerance, e is the error computed with (3.22), then the new time step should be taken as

$$h_{new} = \eta h_{old} \left|\frac{TOL}{e}\right|^{\frac{1}{p}}, \tag{3.24}$$

where a *safety factor* $\eta \approx 0.8 - 0.9$ has been introduce to offset the inaccurate bounds above. Note that using an embedded method is essentially free since all work is done in computing the k_i coefficients.

The embedded RK methods are depicted in extended Butcher's tableaux of the form

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \cdots & a_{ss} \\ \hline & b_1 & \cdots & b_s \\ & \widehat{b}_1 & \cdots & \widehat{b}_s \end{array} \tag{3.25}$$

For example, the well-known `ode45` package uses RK45 what is a 5th order method used to error estimation for a 4th order method. Its is due to Dormand and Prince. Other useful methods are by Fahlberg and Cash-Karp.

4 Multistep methods and stability

Runge-Kutta methods are multistage and in order to compute y_{n+1} they require y_n. In this section we will turn the situation upside down and consider methods that utilize y_n, y_{n-1}, ... to advance the solution.

4.1 Linear multistep methods

We again start with the usual ODE to solve

$$y' = f(x, y). \tag{4.1}$$

A general *linear multistep* method is usually written as

$$\sum_{j=0}^{s} \alpha_j y_{n+j} = h \sum_{j=0}^{s} \beta_j f(x_{n+j}, y_{n+j}), \tag{4.2}$$

where s is the number of steps used. We can see that the value of y_{n+s} is computed from the values y_{n+s-1}, y_{n+s-2}, ..., y_n. If $\beta_s = 0$ the method is explicit, otherwise it is implicit. We also usually adopt a normalization $\alpha_s = 1$. Now, we will see some examples of multistep methods.

Example. (*Adams methods*) These are the first historical multistep methods developed by John Couch Adams (the same who predicted the existence of Neptune just with mathematics!) appeared as a solution of problems in capillary rise stated by Francis Bashforth in middle of XIX century. Therefore, multistep methods are older that Runge-Kutta's.

Adams method utilize the choice

$$\alpha_s = 1, \quad \alpha_{s-1} = -1, \quad \alpha_j = 0 \text{ for } j < s-1. \tag{4.3}$$

The coefficients β_j are chosen accordingly to maximize the order of convergence. For example, if we choose an explicit method, i.e. $\beta_s = 0$, then it is possible to choose β_j with $j = 0, ..., s-1$ to have a method of order s (by Taylor expansion). For implicit methods we have $\beta_s \neq 0$ and hence, one additional degree of freedom. It is then possible to construct $s+1$ order $s-$step methods. Some examples are summarized in Tabs. 3-4. Explicit methods come with a name Adams-Bashforth while implicit Adams-Moulton.

Usually, Adams methods can be constructed with a use of the polynomial interpolation. Let $p = p(x)$ be a polynomial of degree $s-1$ that agrees with $f(x, y(x))$ at x_n, x_{n+1}, ..., x_{n+s-1} (Lagrange interpolation). Then, by integrating (4.1) and approximating f by p we can construct a method

$$y_{n+s} = y_{n+s-1} + \int_{x_{n+s-1}}^{x_{n+s}} p(t)dt. \tag{4.4}$$

The simplest example is the constant polynomial (Euler's method), however much more useful is the trapezoidal rule devised by approximating f by a linear interpolation at x_n and x_{n+1}.

# of steps	Scheme
1	$y_{n+1} = y_n + hf(x_n, y_n)$ (forward Euler)
2	$y_{n+2} = y_{n+1} + \frac{h}{2}\left(-f(x_n, y_n) + 3f(x_{n+1}, y_{n+1})\right)$
3	$y_{n+3} = y_{n+2} + \frac{h}{12}\left(5f(x_n, y_n) - 16f(x_{n+1}, y_{n+1}) + 23f(x_{n+2}, y_{n+2})\right)$

Table 3: Some Adams-Bashforth methods.

# of steps	Scheme
1	$y_{n+1} = y_n + \frac{h}{2}\left(f(x_n, y_n) + f(x_{n+1}, y_{n+1})\right)$ (trapezoidal rule)
2	$y_{n+2} = y_{n+1} + \frac{h}{12}\left(-f(x_n, y_n) + 8f(x_{n+1}, y_{n+1}) + 5f(x_{n+2}, y_{n+2})\right)$
3	$y_{n+3} = y_{n+2} + \frac{h}{24}\left(f(x_n, y_n) - 5f(x_{n+1}, y_{n+1}) + 19f(x_{n+2}, y_{n+2}) + 9f(x_{n+3}, y_{n+3})\right)$

Table 4: Some Adams-Moulton methods.

Example. (*Backward differentiation formulas (BDFs)*) As the Adams methods are based on approximating the integral of f, BDFs are based on discretizing the derivative of y. The general scheme has the form

$$\sum_{j=0}^{s} \alpha_j y_{n+j} = h\beta_s f(x_{n+s}, y_{n+s}), \tag{4.5}$$

that is we take $\beta_j = 0$ for $j = 0, 1, ..., s-1$. The name comes from the fact that since $y'(x) = f(x, y(x))$ the above says that the derivative can be approximated by past s values of y. Some useful BDFs are presented in Tab. 5. They all have the order equal to number of steps.

These methods were introduced in 1950s by Curtiss and Hirschfelder and later popularized by Gear. They are very useful in integrating the so-called stiff equations which we will describe further.

Similarly as with the RK methods, one can derive some order conditions for a given linear multistep scheme had a particular order. We do not want to address this technical issue here. Moreover, in contrast to the RK methods the question of choosing appropriate starting points for multistep schemes is highly nontrivial. Take a $s-$step method of order p and assume that we know y_0 from the initial condition. We have to calculate y_j for $j = 1, 2, ..., s-1$ in such a way to not to decrease the accuracy. In practice this is done by using some simpler $p-1$ order method (for example RK ones). How a lower order scheme can give starting values for p order method?

It is useful to see this on an example (the general case can be found in the literature). Suppose that we would like to use second order Adams method from Tab. 3 and know

# of steps	Scheme
1	$y_{n+1} = y_n + hf(x_{n+1}, y_{n+1})$ (backward Euler)
2	$3y_{n+2} - 4y_{n+1} + y_n = 2hf(x_{n+2}, y_{n+2})$
3	$11y_{n+3} - 18y_{n+2} + 9y_{n+1} - 2y_n = 6hf(x_{n+3}, y_{n+3})$

Table 5: Some Backward Differentiation methods.

y_0. We can use a simple forward Euler to determine y_1 and then to start Adams. More specifically, the so-called one-step error for Euler is

$$y(x_1) - y_1 = y_0 + hy'(x_0) + h^2 y''(\xi) - y_0 - hf(x_0, y_0) = h^2 y''(\xi), \tag{4.6}$$

since $y_0 = y(x_0)$ and $y'(x) = f(x, y(x))$. Therefore, the error in calculating y_1 is $O(h^2)$ *locally*. When we integrated the ODE with Euler method for all x_n we would conduct $O(n)$ steps which would accumulate the local error into a *global* error of $O(nh^2) = O(h)$ as $h \to 0^+$ and nh-bounded. Of course, we have proved that before however, we now see that one step local error is always one order higher that the LTE. Since the number of steps s in a multistep method is independent of n calculating starting values with a $p-1$ order method will always yield p order accurate results.

4.2 Stability

We now have learned about many different numerical methods for solving ODEs. It is relatively easy to compute their LTE and hence to ascertain that a given method approximates the correct ODE. However, we have seen that there are consistent methods which are not convergent. Moreover, there are methods which are convergent but for only certain types of ODEs. This is certainly too weak requirement because a reliable method should converge for a large class of problems. Sometimes, the convergence is hard to prove (especially for PDEs) and one has to proceed in some other way. A consistent method in each step introduces some one-step error. If this error will not be amplified too much during the iteration, the method should be convergent. This feature of not amplification of truncation errors is called the *stability*. Note that when solving anything on the computer we always have to deal with round-off error. If a method were unstable, these numerical errors would exponentially grow yielding a worthless method. This discussion is very vague at the moment and we will proceed to stating matters rigorously. We will start from an example that will motivate our definition of the first notion of stability.

Example. Consider a trivial ODE $y' = 0$ with $y(0) = 0$ and a first order consistent multistep method (verify it!)

$$y_{n+2} - 3y_{n+1} + 2y_n = 0. \tag{4.7}$$

As a starting values we may take any numbers $y_{1,2}(h) \to 0$ as $h \to 0^+$. The general solution of the recurrence is

$$y_n = 2y_0 - y_1 + 2^n(y_1 - y_0), \tag{4.8}$$

which do not converge to the exact solution $y \equiv 0$ unless $y_0 = y_1 = 0$. In fact it diverges exponentially. The method is thus highly unstable!

Recall that the linear recurrence equations can be solved by looking for the power type solutions. If we have an equation

$$\sum_{j=0}^{s} \alpha_j y_{n+j} = 0, \tag{4.9}$$

then plugging $y_n = r^n$ yields

$$\sum_{j=0}^{s} \alpha_j r^j = 0. \tag{4.10}$$

The solutions are thus roots of the above. This motivates us to make the following definition.

Defition 4. *A* characteristic polynomial *for (4.2) is defined as*

$$\rho(r) = \sum_{j=0}^{s} \alpha_j r^j. \tag{4.11}$$

For instance, the characteristic polynomial for the above example is $\rho(r) = (r-1)(r-2)$. We see that if ρ has a root that is greater than one in modulus, it cannot be convergent. It is interesting to ask the opposite question: whether the method is convergent provided that all roots have modulus less than equal to one? If there are no repeated roots then one can easily see than y_n will converge. However, if there is a repeated root, then a special solution of the recurrence will contain terms $n^k r^n$ with modulus n^k and thus divergent.

Defition 5. *A linear s–step method (4.2) is* zero-stable *if the roots r_j of characteristic polynomial ρ satisfy*

$$|r_j| \le 1 \quad j = 1, 2, ..., s, \tag{4.12}$$

and if r_{j_0} is a repeated root then

$$|r_{j_0}| < 1. \tag{4.13}$$

If the above conditions are satisfied the polynomial ρ is said to satisfy the root condition*.*

Example. For Adams methods of the form

$$y_{n+s} = y_{n+s-1} + h \sum_{j=0}^{s} \beta_j f(x_{n+j}, y_{n+j}), \tag{4.14}$$

we have

$$\rho(r) = r^s - r^{s-1} = r^{s-1}(r-1). \tag{4.15}$$

The repeated root is $r = 0 < 1$ while the other $r = 1$ yielding a zero-stable method.

Not that the zero-stability is a result of application of the multistep method for a particular trivial ODE $y' = 0$. It is remarkable that this suffices for a general ODE!

Theorem 2 (Dahlquist)**.** *If a linear multistep method (4.2) is applied to an ODE (4.1) then*

$$\textit{consistency} \; + \; \textit{zero-stability} \iff \textit{convergence}. \tag{4.16}$$

This result is typical however highly nontrivial in numerical analysis. It basically means that a certain notion of stability is equivalent to convergence (since consistency is almost always satisfied by construction). We will see this result in different guises in further chapters. Note also that for one-step method such as Euler or trapezoidal, the consistency guarantees that the only root of ρ is equal to 1. Therefore, these methods are always zero-stable and hence, convergent.

The notion of zero-stability comes from the fact that we actually have taken $h \to 0^+$ and investigated what happens with the recurrence. However, in practice we are always conducting our calculations for finite $h > 0$. There are certain important stability issues that can arise in this situation. Most importantly, in many cases in order to achieve sufficient accuracy some methods require prohibitively small steps that cannot be resolved by the computer.

We will develop a different notion than zero-stability that can yield some useful information in practice. The study of stability usually starts from the so-called test equation and applying a particular method for solving it. For ODE's the usual choice is

$$y' = \lambda y, \tag{4.17}$$

which has a solution $y(x) = y_0 \exp(\lambda x)$ and hence decays exponentially for $\lambda < 0$. Forward Euler method applied to solving the above has the form

$$y_{n+1} = (1 + \lambda h) y_n. \tag{4.18}$$

This can be continued inductively

$$y_{n+1} = (1 + \lambda h)^n y_0. \tag{4.19}$$

Therefore, if $\lambda < 0$ the method converges to zero only when

$$|1 + \lambda h| \leq 1, \tag{4.20}$$

that is for $\lambda h \in [-2, 0]$. Suppose, that $\lambda = -100$ then we would require $h \leq 0.02$ to have a stable method. This notion of stability can also we understood when there in an error in the initial data (very common situation). For if $y_0^\delta = y_0 + \delta$ is the noisy data we have

$$|y_{n+1}^\delta - y_{n+1}| \leq |1 + \lambda h|^n \delta, \tag{4.21}$$

and hence if $\lambda h < -2$ the initial measurement error will be amplified exponentially. The method would thus be useless in applications.

Since usually one deals with systems of ODEs and eigenvalues of the corresponding Jacobians, which can be imaginary, it is customary to let $z := \lambda h \in \mathbb{C}$ and talk about the *region of absolute stability*. In the case of Euler method $|1 + z| \leq 1$ is a circle on a complex plane centred at $z = -1$ and radius 1. We can now give a definition applicable for a general one-step methods.

Defition 6. *A one step method*

$$y_{n+1} = G(\lambda h) y_n, \tag{4.22}$$

is absolutely stable *if*

$$|G(z)| \leq 1. \tag{4.23}$$

The function G *is called the* gain factor, amplification factor *or* stability function.

If we apply a general multistep method (4.2) to (4.17) then we obtain

$$\sum_{j=0}^{s} (\alpha_j - z\beta_j)\, y_{n+j} = 0. \tag{4.24}$$

Considering power function solutions we are immediately lead to the following definition.

Defition 7. *A stability polynomial for (4.2) is*

$$\pi(r, z) := \sum_{j=0}^{s} (\alpha_j - z\beta_j)\, r^j = \rho(r) - z\sigma(r). \tag{4.25}$$

In order for the above recurrence to have bounded solutions we thus require that π satisfies the root condition stated in Def. 5.

Defition 8. *The* region of absolute stability *for a multistep method (4.2) is the set*

$$\{z \in \mathbb{C} : \pi(\cdot, z) \textit{ satisfies the root condition}\}. \tag{4.26}$$

A method is zero-stable if $z = 0$ lies in the stability region.

Example. (*Backward Euler*) For the backward Euler we have $y_{n+1} = y_n + zy_{n+1}$. Therefore,

$$\pi(r, z) = (1 - z)r - 1, \tag{4.27}$$

with root $r_1 = (1 - z)^{-1}$. Whence,

$$|(1 - z)^{-1}| \leq 1 \iff |1 - z| \geq 1. \tag{4.28}$$

The stability region is thus the exterior of a unit circle centred at $z = 1$. Note that the same can be inferred from actually solving the backward Euler recurrence, i.e. $y_{n+1} = (1 - z)^{-1}y_n$ thus obtaining the gain function.

Example. (*Trapezoidal method*) For trapezoidal scheme we have

$$\pi(r, z) = \left(1 - \frac{1}{2}z\right) r - \left(1 + \frac{1}{2}z\right), \tag{4.29}$$

with a root

$$r_1 = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}. \tag{4.30}$$

The condition $|r_1| \leq 1$ translates into $\mathrm{Re}(z) \leq 0$ which is the whole left half-plane. A rather large region! The trapezoidal method is very simple, robust, and versatile numerical scheme.

5 Boundary value problems

So far we have considered only initial value problems which are natural to evolution equations. When considering quantities that change in space one is usually faced with solving boundary value problems (BVPs) which are of utmost importance in applied mathematics. In this chapter we will consider several methods of solving them.

The general boundary value problem for ODEs can be formulated as follows

$$y'' = f(x, y, y'), \quad y(0) = \alpha, \quad y(1) = \beta. \tag{5.1}$$

Here we have imposed Dirichlet boundary conditions however, any other could be applied here. From the general theory it follows that the above has a unique solution if f, f_y, and $f_{y'}$ are continuous in a region $[0, 1] \times \mathbb{R}^2$ with $f_x > 0$ and f_y bounded. We will develop all the methods for a simple BVP of the form

$$-y'' = f(x), \quad y(0) = 0, \quad y(1) = 0, \quad x \in (0, 1). \tag{5.2}$$

This model example is very simple to solve however, notwithstanding being so simple, (5.2) can present almost all difficulties that arise when constructing numerical methods for solving BVPs.

5.1 Shooting method

The shooting method is based on an idea to transform the BVP into an initial value problem for ODE where the initial value to the derivative is treated as a parameter (shooting angle).

$$-y'' = f(x), \quad y(0) = 0, \quad y'(0) = \varphi. \tag{5.3}$$

From the above cited theory it follows that there exists a unique φ^* such that $y(1) = 0$. Therefore, the problem is well-posed. The interpretation is straightforward: we try to shoot a projectile at the angle φ to hit the target at $(1, 0)$.

In practice we use some ODE integrator to solve (5.3) up to the point $x = 1$ for a general initial condition φ. We can then define the function

$$F(\varphi) = y(1; \varphi), \tag{5.4}$$

and apply some zero searching algorithm (bisection, Newton, secant, ...) to it which usually requires some initial guess of φ^*.

Single shooting method is in practice used mainly for solving some very simple ODEs for which initial value integrators can proceed quickly. The method has some drawbacks:

- computational cost especially in systems (ODE + nonlinear equation to solve),
- stability (BVP can be stable while IVP not).

A partial aid of the above is realized as a multiple shooting methods where one divides the interval $[0, 1]$ into some smaller portions and on each of then shoots and patches the solution. This procedure has much better stability properties.

5.2 Finite difference methods

We can also discretize the BVP directly by the use of the finite differences. Let $h = 1/(n+1)$ he the mesh width for $x_j = jh$ with $j = 0, 1, ..., m+1$. The second derivative can be conveniently approximated by the second order centred difference (check it!)

$$y''(x) \approx \delta_-\delta_+ y(x) = \frac{y(x-h) - 2y(x) + y(x+h)}{h^2}. \tag{5.5}$$

Applying the above to (5.2) yields

$$\frac{1}{h^2}(y_{j-1} - 2y_j + y_{j+1}) = -f(x_j), \quad j = 1, 2, ..., n. \tag{5.6}$$

We know that $y_0 = y_{n+1} = 0$ and hence we are left with a system of n linear equations to solve with n unknowns. If $y = (y_j)_{j=1}^n$ is a column vector we can write the system as

$$Ay = -F, \tag{5.7}$$

where

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & & \ddots & & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}, \quad F = \begin{bmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_n) \end{bmatrix}. \tag{5.8}$$

This is a so-called tridiagonal system and it is nonsingular and can be solved in a very efficient numerical algorithms than yield the solution in $O(n)$ steps (instead of $O(n^3)$ required by the Gaussian elimination).

Now, we will discuss the convergence properties of the above scheme. Notice that when $h \to 0^+$ the number of equations grow to infinity. We have to somehow estimate whether the resulting solution approximates the exact value of $y(x)$. By $\widehat{y} = (y(x_j))_{j=1}^n$ denote the vector of the exact solution of (5.2) evaluated at the grid points. The local truncation error τ_j is clearly

$$\tau_j = -\left(\frac{y(x_{j-1}) - 2y(x_j) + y(x_{j+1})}{h^2} + f(x)\right) = -\frac{1}{12}h^2 y''''(\xi_j), \quad h \to 0^+. \tag{5.9}$$

Therefore, the method is second order accurate. Now, if define the truncation error vector as $\tau = (\tau_j)_{j=1}^n$ we have

$$\tau = -\left(A\widehat{y} + F\right). \tag{5.10}$$

Whence, the error $e = \widehat{y} - y$ of the method satisfies a similar linear system as y

$$Ae = -\tau, \tag{5.11}$$

with $e_0 = e_{n+1} = 0$. The nonhomogeneous term is now represented by the truncation error. Since A is tridiagonal it can be inverted and hence after taking a norm

$$\|e\| \le \|A^{-1}\| \, \|\tau\|. \tag{5.12}$$

Since $\|\tau\| = O(h^2)$ as $h \to 0^+$ we would like to have $\|A^{-1}\| \leq C$ in order to $\|e\|$ be of second order. Remember that the size of A increases with $h \to 0^+$ and hence, we are faced with the notion of stability. The truncation error should not be amplified when we refine the grid.

Defition 9. *A numerical method for solving BVP in the form* $Ay = F$ *is* stable *if* $\|A^{-1}\| \leq C$ *as* $h \to 0^+$ *for some constant* $C > 0$ *independent on* h.

We now again can see that consistency + stability is equivalent to convergence. In order to prove that our scheme (5.6) is stable we have to choose an appropriate norm. There are many choices but probably the second norm yields the clearest results

$$\|A\|_2 := \sup_{y \neq 0} \frac{\|Ay\|_2}{\|y\|_2}, \quad \|y\|_2 := \left(h \sum_{j=1}^{n} y_j^2 \right)^{\frac{1}{2}}. \tag{5.13}$$

Since, A is symmetric the computation of the norm is simple because

$$\|A\|_2 = \max_{1 \leq k \leq m} |\lambda_k|, \tag{5.14}$$

where λ_j is the eigenvalue of A. The norm of the inverse is just

$$\|A^{-1}\|_2 = \left(\min_{1 \leq k \leq m} |\lambda_k| \right)^{-1}. \tag{5.15}$$

It follows from linear algebra that for A as in (5.8) we have

$$Ay^k = \lambda_k y^k, \quad y^k = (\sin(k\pi jh))_{j=1}^n, \quad \lambda_k = \frac{2}{h^2} \left(\cos(k\pi h) - 1 \right), \tag{5.16}$$

where y^k is the eigenvector of A with corresponding eigenvalue λ_k. We conclude that the smallest eigenvalue is

$$\lambda_1 = \frac{2}{h^2} \left(\cos(\pi h) - 1 \right) = -\pi^2 + O(h^2) < 0 \quad \text{as} \quad h \to 0^+. \tag{5.17}$$

Further, we have

$$\|e\|_2 \leq \|A^{-1}\|_2 \|\tau\|_2 \leq \frac{h^2}{2\left(1 - \cos(\pi h)\right)} \|\tau\|_2, \tag{5.18}$$

and to show the convergence we have to estimate the truncation error. First, from (5.2) we have $\tau_j = -\frac{1}{2}h^2 y''''(\xi_j) = -\frac{1}{12}h^2 f''(\xi_j)$ what yields

$$\|\tau\|_2 = \frac{h^2}{12} \left(h \sum_{j=1}^{n} f(\xi_j)^2 \right)^{\frac{1}{2}} = \frac{h^2}{12} \|f''\|_2 + O(h^3) \quad \text{as} \quad h \to 0^+, \tag{5.19}$$

since the sum in the above is just the Riemann sum for the integral. Here, $\|f''\|_2$ is the continuous second norm of f''. We have thus proved the following result.

Theorem 3. *The discrete scheme (5.6) is second order convergent with*

$$\|e\|_2 \leq \frac{\pi^2}{12} \|f''\|_2 h^2 + O(h^3) \quad \textit{as} \quad h \to 0^+. \tag{5.20}$$

Notice how different is this approach to our previous considerations concerning initial value problems. We will see that similar reasoning is needed for solving PDEs.

5.3 Finite element methods

Now we change gears and introduce yet another, very robust, method of a completely different nature. Finite element method is indispensable when one solves problems on unstructured grids or realistic geometries. It is mainly used in many multiphysics applications such as construction, heat conduction, acoustics, material science, and electricity. We will develop it from the very beginning based on our simple problem (5.2).

The finite element method begins with the so-called weak formulation. Let us go back to the original problem (5.2), multiply it by a sufficiently smooth function φ such that $\varphi(0) = \varphi(1) = 0$, and integrate by parts to obtain

$$(f, \varphi) := \int_0^1 f(x)\varphi(x)dx = -\int_0^1 u''(x)\varphi(x)dx = \int_0^1 u'(x)\varphi'(x)dx =: a(u, \varphi), \qquad (5.21)$$

where we have defined two (symmetric) bilinear forms $(\cdot, \cdot)$ and $a(\cdot, \cdot)$. Notice that thanks to the vanishing of φ at the boundary of $[0, 1]$ guarantees that there are no remainder terms from the integration by parts. This Dirichlet boundary condition in the jargon of finite elements is called *essential* while Neumann would be called *natural*. We define the function space in which we will look for solutions of the above

$$V = \left\{u \in L^2[0, 1] : \quad a(u, v) < \infty \text{ and } u(0) = u(1) = 0\right\}, \qquad (5.22)$$

where $L^2[0, 1]$ is the Lebesgue space of square-integrable functions on $[0, 1]$. Therefore, we are interested in solving the following *variational* or *weakly formulated* problem

$$\text{Find } u \in V \text{ such that } a(u, \varphi) = (f, \varphi) \text{ for all } \varphi \in V. \qquad (5.23)$$

The adjective *variational* indicates that we can *vary* φ in order to satisfy the above equation. Notice that also we have moved some regularity requirements from u to φ. This means that the solution of (5.23) does not have to be twice-differentiable, hence the name - *weak*. In contrast, the solution of (5.2) is called *classical*. It is also easy to show that when $u \in C^2[0, 1]$ is a solution of the weak problem it is also a classical solution. Therefore, sufficiently regular weak solutions are also classical. However, it is sometimes much more convenient to work with variational formulation since we can then use the theory of Hilbert spaces.

Up to now we have only formulated the original problem in a new setting. Now, we will see how to construct a very robust numerical method known as *Ritz-Galerkin approximation*. This is extremely simple. Choose any finite dimensional subspace $S \subset V$ and replace (5.23) with

$$\text{Find } u_S \in S \text{ such that } a(u_S, \varphi) = (f, \varphi) \text{ for all } \varphi \in S. \qquad (5.24)$$

Of course, the above formulation does not guarantee that it is well-defined, i.e. whether it possesses a unique solution. The following result establishes that fact.

Proposition 1. *Let* $f \in L^2[0, 1]$. *Then, there is only one solution of (5.24).*

Proof. Let us choose a basis of S, say $\{\phi_i\}_{i=1}^n$. Due to linearity, the system (5.24) can then be written as

$$KU = F, \tag{5.25}$$

which is a linear finite dimensional system. Here, $U = (U_i)$ is the solution vector, $K = (K_{ij}) = (a(\phi_i, \phi_j))$ is the *stiffness matrix*, and $F = (F_i) = ((f, \phi_i))$ is the *load vector* (terms borrowed from engineering).

Note that for linear finite dimensional problems the existence is equivalent to uniqueness. We will show the latter. Suppose that there exists a vector $V = (V_i)$ such that $KV = 0$ and a corresponding function $v = \sum_1^n V_i\phi_i$. It, in turn, means that $a(V, \phi_i) = 0$. If we multiply this by V_i, sum over j and use linearity, we obtain $a(v, v) = 0$. But from the definition of a we have $0 = \int_0^1 (v')^2 dx$ which can happen only when $v' \equiv 0$. However, the boundary conditions imply that $v \equiv 0$ and since $\{\phi_i\}$ constitute a basis we have $V = 0$. This ends the proof. □

Now, the method (5.24) can yield some useful approximations however, it remains to be proved that it really converges in some sense when we increase the dimensionality of the subspace S. We begin with a very important remark concerning the orthogonality. When we subtract (5.24) from (5.23) we obtain

$$a(u - u_s, w) = 0, \quad w \in S. \tag{5.26}$$

Since $a(\cdot, \cdot)$ is symmetric, bilinear, and positive definite it can be promoted to be a scalar product which implies the so-called *energy norm*

$$\|v\|_E := \sqrt{a(v, v)}. \tag{5.27}$$

Suppose we would like to estimate the difference between u and u_S in this natural norm. To this end we choose $v \in S$ and write

$$\|u-u_S\|_E^2 = a(u-u_S, u-u_S) = a(u-u_s, u-v)+a(u-u_S, v-u_S) = a(u-u_S, u-v) \leq \|u-u_S\|_E\|u-v\|_E, \tag{5.28}$$

where in the third equality we have used (5.26) while in the last - Schwarz inequality. Therefore

$$\|u - u_S\|_E \leq \|u - v\|_E, \quad v \in S. \tag{5.29}$$

Since the above is true for any $v \in S$ it is also valid for the minimum (the subspace S is finite dimensional)

$$\|u - u_S\|_E \leq \min\{\|u - v\|_E : v \in S\}. \tag{5.30}$$

But of course, $\min\{\|u - v\|_E : v \in S\} \leq \|u - u_S\|_E$ since $u_S \in S$. Therefore, we have proved the following result.

Theorem 4. *It holds that* $\|u - u_S\|_E = \min\{\|u - v\|_E : v \in S\}$.

The above states that the Ritz-Galerkin approximation is optimal in the energy norm. That is, this is the best approximation when we measure the discrepancy in that norm.

We can go even further and try to find the bound for the error in the usual L^2 norm. We will use a reasoning that is called the duality argument. Let w be a solution of the following equation

$$-w'' = u - u_S, \quad w(0) = w(1) = 0. \tag{5.31}$$

Taking the usual scalar product of the above with $u - u_S$ we obtain

$$\|u-u_S\|^2 = (u-u_S, u-u_S) = (u-u_S, -w'') = a(u-u_S, w) = a(u-u_S, w-v), \tag{5.32}$$

for $v \in S$. In the third equality we have integrated by parts and used the boundary conditions while in the last the orthogonality (5.26). Now, an application of the Schwarz inequality leads to

$$\|u - u_S\| \leq \frac{\|u - u_S\|_E \|w - v\|_E}{\|u - u_S\|} = \frac{\|u - u_S\|_E \|w - v\|_E}{\|w''\|}, \tag{5.33}$$

which after taking infimum produces

$$\|u - u_S\| \leq \frac{\|u - u_S\|_E}{\|w''\|} \inf\{\|w - v\|_E : v \in S\}. \tag{5.34}$$

We can see that if w can be well-approximated with a function from S the right-hand side of the above can be made very small. To be precise, we make the following *approximation assumption*

$$\inf\{\|w - v\|_E : v \in S\} \leq \epsilon \|w''\|, \tag{5.35}$$

for some $\epsilon > 0$. Whence,

$$\|u - u_S\| \leq \epsilon \|u - u_S\|_E, \tag{5.36}$$

and the L^2 norm is much smaller than the energy norm. When we replace u instead of w in (5.35) and invoke Theorem 4 we further have

$$\|u - u_S\|_E \leq \epsilon \|u''\|. \tag{5.37}$$

Combining the two above estimates leads us finally to the following important result.

Theorem 5. *Assume (5.35). Then, it holds that* $\|u - u_S\| \leq \epsilon^2 \|f\|$.

What is left to show is the fact that the above approximation assumption can in reality be made. That is, ϵ can be made arbitrarily small with a suitable choice of S. This leads to the notion of *finite element method*.

Let us partition the $[0, 1]$ interval into (not necessarily equal) segments $0 = x_0 < x_1 < ... < x_n = 1$ and let S be a space of *continuous* functions such that $v|[x_{i-1}, x_i]$ is a *linear polynomial* and $v(0) = v(1) = 0$. As the basis we choose ϕ_i such that $\phi_i(x_j) = \delta_{ij}$. These functions can be easily visualised as so-called *hat functions*. Given $v \in S$ is called an *interpolant* and can be written as $v_I = \sum_1^n v(x_i)\phi_i$. Having chosen the linear element we are able to prove the fundamental approximation result.

Lemma 1. *Let* $h = \max_i(x_i - x_{i-1})$. *Then for all* $u \in V$

$$\|u - u_I\|_E \leq Ch\|u''\|, \tag{5.38}$$

where C *is independent of* h *and* u.

Proof. It is sufficient to prove the inequality piecewise, i.e.

$$\int_{x_{i-1}}^{x_i} (u-u_I)^2 dx \le c(x_i - x_{i-1})^2 \int_{x_{i-1}}^{x_i} (u'')^2 dx. \tag{5.39}$$

Let us introduce the error $e = u - u_I$. Observe that since u_I is piecewise linear we have $u_I'' = 0$ almost everywhere and hence, $e'' = u''$. Moreover, after substitution $x = x_{i-1} + y(x_i - x_{i-1})$ we can write the above integrals as

$$\int_0^1 e(y)^2 dy \le c \int_0^1 e''(y)^2 dy. \tag{5.40}$$

Note that we have got rid of any scale present in the problem, i.e. the estimate do not depend on the mesh spacing h. This procedure is very useful and known as homogeneity argument.

Now, to prove our claims we observe that since $e(0) = e(1) = 0$ by Rolle's Theorem we must have $w'(\xi) = 0$ for some $\xi \in (0,1)$. From this we can estimate

$$|e'(x)| = \left|\int_\xi^x e''(y)dy\right| \le \left(\int_\xi^x dy\right)^{\frac{1}{2}} \left(\int_\xi^x e''(y)^2 dy\right)^{\frac{1}{2}} \le |x-\xi|^{\frac{1}{2}} \left(\int_0^1 e''(y)^2 dy\right)^{\frac{1}{2}}, \tag{5.41}$$

where the first inequality follows from Schwarz. Now, squaring and integrating the above yields

$$\int_0^1 e'(y)^2 dy \le \left(\int_0^1 |x-\xi| dx\right) \int_0^1 e''(y)^2 dy \le \frac{1}{2} \int_0^1 e''(y)^2 dy, \tag{5.42}$$

and therefore $c = \frac{1}{2}$. The proof is completed. □

Note that the above lemma is valid for any $u \in V$. If we gather all we have found for the solution of the Ritz-Galerkin approximation (5.24) we arrive at the fundamental estimate for the finite element method

Theorem 6. *Let* u *satisfy (5.23) while* u_S *is a solution of (5.24) with* S *being the linear element space. Then, for* $h = \max_i (x_i - x_{i-1})$ *we have*

$$\|u - u_S\| + Ch\|u - u_S\|_E \le 2(Ch)^2 \|f\|, \tag{5.43}$$

where $C > 0$ *is a constant independent on* u *and* h.

Proof. Notice that due to Lemma 1 we can take $\epsilon = Ch$ in (5.35). Then, we can use Theorem 5. □

We have thus proved that the finite element method with linear interpolation is second order accurate for approximating the function while first order for approximating the derivative. The finite element method is very robust since the overall methodology stays the same regardless of the geometry of the domain. It can be carried over to two and three dimensional regions with irregular shapes without the effort required by the finite difference method. In two dimensional geometries, the main domain is usually decomposed into triangles on which hat functions are constructed. This allows to treat almost arbitrary shapes and thus has found its applications in engineering.

6 Methods for conservation laws and hyperbolic equations

Conservation laws are one of the most fundamental relations in science. They mathematically encapsulate the balance between various quantities in the investigated system. For example, they are a basis of fluid mechanics, electromagnetism, diffusion models, weather prediction and many more. The resulting mathematical formulation is usually given as a system of nonlinear first order hyperbolic equations. Since their importance many numerical methods are still being developed to treat different situations that can arise in practice. Mostly, they go into two families: finite differences (which we know a lot) and finite volumes (which we will learn a little). In what follows we will demonstrate the development of numerical methods for scalar equations in one dimensional space. However, many of ideas can be carried over to higher dimensions or systems.

As we recall from the PDE course, conservation laws can usually be written as

$$u_t(x,t) + f(u(x,t))_x = 0, \tag{6.1}$$

where $u = u(x,t)$ is the conserved quantity while $f = f(u)$ is the flux. We will mostly be concerned with linear advection equation for which $f(u) = cu$ and hence

$$u_t + cu_x = 0, \quad x \in \mathbb{R}, \quad t > 0. \tag{6.2}$$

We know that the solution of the above is given by $u(x,t) = \phi(x - ct)$ for $\phi = \phi(x)$ being the initial condition. This equation will be our model problem for developing many useful (or not) methods.

6.1 Finite difference methods and stability

We begin with finite difference methods, i.e. we introduce the grid $(x_j, t_n) = (jk, nh)$ with $n\mathbb{N}$ and $j \in \mathbb{Z}$. Moreover, by u_j^n we denote the numerical approximation of $u(x_j, t_n)$. As usual in the finite difference method, we discretize the equation (6.2) by approximating derivatives. First, let us use δ_+ to discretize temporal and δ_- for the spatial derivative to obtain

$$\frac{u_j^{n+1} - u_j^n}{h} + c\frac{u_j^n - u_{j-1}^n}{k} = 0, \tag{6.3}$$

which can be written as

$$u_j^{n+1} = u_j^n - \nu\left(u_j^n - u_{j-1}^n\right), \quad \nu = \frac{ch}{k}. \tag{6.4}$$

Since for $c > 0$ the exact solution of the PDE (6.2) is a wave travelling to the right, the above numerical scheme is called the *upwind method* because it needs values of u in the direction of the propagating wave. The upwind method for $c < 0$ could be obtained exactly in the same manner with δ_+ used instead of δ_- on the spatial side.

Due to the first order finite differences used in the discretization we expect that the method is itself of the same accuracy. This is formulated in the following result. Note that since we are considering PDEs we have to measure the numerical error in some norm. In practice it is convenient to use either first, second or supremum norm with respect to the spatial variable, couple h and k somehow (we will see how), and take the limit $h \to 0^+$.

Theorem 7. *Assume that $0 < \nu \leq 1$ then the upwind method (6.4) gives a first order approximation to the solution of (6.2). More specifically, for $t \leq T$ we have*

$$\max_j |u(x_j, t_n) - u_j^n| \leq CT\, h, \quad h \to 0^+ \quad \textit{with} \quad \frac{h}{k} \quad \textit{fixed}, \tag{6.5}$$

where C is independent on h.

Proof. Let us denote the error by $e_j^n := u(x_j, t_n) - u_j^n$. Then, expanding in Taylor series and using (6.4) we have

$$e_j^{n+1} = u(x_j, t_{n+1}) - u_j^{n+1} = e_j^n + u_t(x_j, t_n)h + \frac{1}{2}u_{tt}(x_j, \tau_n)h^2 + \nu(u_j^n - u_{j-1}^n). \tag{6.6}$$

Since by (6.2) we have $u_t = -cu_x$ we can further write

$$e_j^{n+1} = e_j^n + \nu\left(ku_x(x_j, t_n) - u_j^n + u_{j-1}^n\right) + \frac{1}{2}u_{tt}(x_j, \tau_n)h^2. \tag{6.7}$$

Further, because for any function y we have $y'(x) = k^{-1}(y(x) - y(x-k)) + ky''(\xi)/2$ we can simplify

$$e_j^{n+1} = e_j^n + \nu\left(e_j^n - e_{j-1}^n\right) + \frac{1}{2}\left(-u_{xx}(\xi_j, t_n)k + u_{tt}(x_j, \tau_n)h\right)h. \tag{6.8}$$

Taking the maximum and using the assumption that $0 < \nu \leq 1$ we obtain

$$\max_j |e_j^{n+1}| = (1-\nu)\max_j |e_j^n| + \nu \max_j |e_j^n| + \frac{h}{2}\max_j |(-u_{xx}(\xi_j, t_n)k + u_{tt}(x_j, \tau_n)h|. \tag{6.9}$$

The truncation error can be estimated by the fact that h/k remain fixed when $h \to 0^+$. Therefore,

$$\max_j |e_j^{n+1}| \leq \max_j |e_j^n| + Ch^2. \tag{6.10}$$

When we repeat this inequality n times we arrive at

$$\max_j |e_j^{n+1}| \leq Cnh^2, \tag{6.11}$$

where we used the fact that the initial conditions are the same, i.e. $e_j^0 = 0$. Because $nh \leq T$ we conclude the proof. □

Notice the importance of the number $\frac{ch}{k}$ in the above proof. It appears in any numerical method for hyperbolic equation and goes by the name of the pioneers of finite different schemes: *Courant, Friedrichs, Lewy number* (CFL). We will denote it by

$$\nu := \frac{ch}{k}. \tag{6.12}$$

Loosely speaking, solutions of hyperbolic equations evaluated at some point in space-time depend on its past values in some region in space. For example, a solution of (6.2) at (x, t) depends only on the point $(x - ct, 0)$. A set of all points that influence the value of $u(x, t)$ is called the *domain of dependence*. A similar definition can be given for a *numerical domain of dependence*. The seminal paper of CFL gave a fundamental theorem stating that a numerical method can be convergent only if the domain of dependence is a subset of the numerical domain of dependence (at least for small k and h). In the upwind method we see that since calculating u_j^{n+1} requires u_j^n and u_{j-1}^n. If we carry this over until $t = 0$, we see that it requires knowing $\phi(x_i)$ such that $x_j - k/ht_n \leq x_i \leq x_j$. Due to CFL condition, this set has to contain $x_n - ct_n$ and this could only happen when $0 < \nu \leq 1$.

Note that CFL condition is only *necessary* and there are examples when it is not sufficient. In practice one usually fixes the number $0 < \nu \leq 1$ and chooses the temporal grid size according to $h = k\nu/c$. This means that the space and time discretization has to be of the same order.

Another way of discretizing (6.2) is to use centred approximation of the derivative. Having replaced temporal and spatial derivatives with δ_0 would lead to the so-called *leapfrog method*. This scheme is sometimes useful however, in order to calculate the solution at time $n + 1$ it requires to use values at time $n - 1$. This may not be very useful in practise when we would like to have more time-stepping algorithm when the next step is calculated from the previous one. We are thus left with centred space discretization which leads to

$$u_j^{n+1} = u_j^n - \frac{1}{2}\nu \left(u_{j+1}^n - u_{j-1}^n \right). \tag{6.13}$$

This may seem as a second-order in space accurate method. However, it has some very serious stability properties making it useless in practice.

Analysing stability for PDEs is usually more elaborate than for ODEs. There are several approaches and one of them is particularly useful, fast, and easy to conduct for linear equations with constant coefficients. This is the *von Neumann* analysis and it is based on Fourier transform. Before we explain it, we define a correct notion of the stability used in PDEs.

Defition 10. *A numerical method calculating* u_j^n *is* ***stable*** *if*

$$\|u^{n+1}\| \leq \|u^n\|, \tag{6.14}$$

where $\|\cdot\|$ *is a chosen norm taken with respect to the spatial variable.*

This definition states that a stable method does not increase the norm of the solution when time advances forward. In other words, a small error introduced in the initial step will not grow with time. Sometimes, the stability condition can be made more stringent $\left\|u^{n+1}\right\| \leq (1 + \alpha h) \|u^n\|$ which would also lead to a useful scheme (compare the proof of the convergence of Euler method). However, we will focus on the more simple one.

Suppose that we have a grid $x_j = jk$ and sequence $\nu_j \in l_{2,k}$ (square summable) where $j \in \mathbb{Z}$. Then, we can define its discrete Fourier transform as

$$\widehat{\nu}(\xi) = \frac{k}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} \nu_j e^{-ijk\xi}, \tag{6.15}$$

where the factor k is needed for consistency with the continuous Fourier transform when $k \to 0^+$. Further, we can recover the original sequence with

$$v_j = \frac{1}{\sqrt{2\pi}} \int_{-\pi/k}^{\pi/k} \widehat{v}(\xi) e^{ijk\xi} d\xi. \tag{6.16}$$

The von Neumann analysis stems from two ingredients. The first one is the Parseval's theorem stating that Fourier transform is an isometry

$$\|v\|_{2,k} = \|\widehat{v}\|_2, \quad \text{where} \quad \|v\|_{2,k} = \left(h \sum_{j=-\infty}^{\infty} |v_j|^2 \right)^{\frac{1}{2}}, \quad \|\widehat{v}\|_2 = \left(\int_{-\pi/k}^{\pi/k} |\widehat{v}(\xi)|^2 d\xi \right)^{\frac{1}{2}}. \tag{6.17}$$

The second ingredient is the fact that $e^{ijk\xi}$ is an *eigenfunction* of any difference operator (similarly, as e^x is a eigenfunction of any derivative operator). This means that we can „diagonalize" any finite difference scheme, i.e. decouple its modes. It has a tremendous meaning since calculating the norm of u_j^n requires knowledge of all points in its numerical domain of dependence. To see this choose any finite difference scheme

$$u_j^{n+1} = \sum_p a_p u_{j-p}^n. \tag{6.18}$$

Its Fourier transform is

$$\widehat{u}^{n+1}(\xi) = \frac{k}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} \left(\sum_p a_p u_{j-p}^n \right) e^{-ijk\xi}. \tag{6.19}$$

Interchanging order of summation we can arrive at

$$\widehat{u}^{n+1}(\xi) = \frac{k}{\sqrt{2\pi}} \sum_p a_p e^{-ipk\xi} \sum_{j=-\infty}^{\infty} u_{j-p}^n e^{-i(j-p)k\xi} = \left(\sum_p a_p e^{-ipk\xi} \right) \widehat{u}^n(\xi) = G(\xi)\widehat{u}^n(\xi). \tag{6.20}$$

Therefore, time-stepping finite difference scheme forward is equivalent to saying that the Fourier transform is multiplied by some *amplification factor* (or symbol). Therefore, the stability in the second norm can be stated by requirement that

$$\|G\widehat{u}^n\|_2 = \left\|\widehat{u}^{n+1}\right\|_2 = \left\|u^{n+1}\right\|_{2,k} \leq \|u^n\|_{2,k} = \|\widehat{u}^n\|_2, \tag{6.21}$$

which happens if and only if

$$|G(\xi)| \leq 1, \tag{6.22}$$

which is the von Neumann stability condition. We have just proved the following fact.

Theorem 8. *A linear numerical method with constant coefficients is stable if and only if its amplification factor satisfies (6.22).*

Example. Let us investigate the stability of the upwind method (6.4). Calculating the discrete Fourier transform we obtain

$$\widehat{u}^{n+1} = \widehat{u}^n - \nu \left(\widehat{u}^n - e^{ik\xi} \widehat{u}^n \right) = \left(1 - \nu + \nu e^{ik\xi} \right) \widehat{u}^n. \tag{6.23}$$

Then, we require that

$$|G(\xi)| = |1 - \nu + \nu e^{ik\xi}| \leq 1. \tag{6.24}$$

We can square the above to obtain

$$|1-\nu+\nu e^{ik\xi}|^2 = (1 - \nu + \nu\cos(k\xi))^2 + (\nu\sin(k\xi))^2 = (1-\nu)^2 + 2\nu(1-\nu)\cos(k\xi) + \nu^2 \leq 1, \tag{6.25}$$

which is equivalent to

$$(1 - \nu)^2 + 2\nu(1 - \nu)\cos(k\xi) \leq (1 - \nu)(1 + \nu). \tag{6.26}$$

When $\nu = 1$ this is obviously satisfied, while when $0 < \nu < 1$ we can divide and obtain

$$\cos(k\xi) \leq 1, \tag{6.27}$$

which is true. Notice that we have obtain exactly the same condition as in the convergence proof and CFL condition.

There is also a quicker way of estimating the size of amplification factor. Look for $k\xi$, for which the expression attains its minimum and maximum values. Computing the derivative and equating it to zero we would obtain $k\xi = 0, \pi$. In these cases $|G(0)| = 1$ and $|G(\pi/k)| = (1 - 2\nu)^2 \leq 1$ only for $0 < \nu \leq 1$.

Example. Now, for the second method we derived (6.13) we obtain

$$\widehat{u}^{n+1} = \widehat{u}^n - \frac{1}{2}\nu\left(e^{-k\xi}\widehat{u}^n - e^{ik\xi}\widehat{u}^n\right) = \left(1 - \frac{1}{2}\nu(e^{-k\xi} - e^{ik\xi})\right)\widehat{u}^n. \tag{6.28}$$

Therefore, the von Neumann stability condition is

$$|G(\xi)| = |1 + i\nu\sin(k\xi)| = \sqrt{1 + \nu^2\sin^2(k\xi)} > 1, \tag{6.29}$$

for any $\nu \neq 0$. Therefore, the method is severely unstable.

As we have seen, the presumably second order in space scheme (6.13) is hopelessly useless. There is, however, a way of making it stable. Since averaging usually improves stability, we can propose the following method

$$u_j^{n+1} = \frac{1}{2}\left(u_{j+1}^n + u_{j-1}^n\right) - \frac{1}{2}\nu\left(u_{j+1}^n - u_{j-1}^n\right) = \frac{1}{2}\left[(1-\nu)u_{j+1}^n + (1+\nu)u_{j-1}^n\right], \tag{6.30}$$

where instead of u_j^n we have taken its averaged value taken over neighbouring grid points. This method is called *Lax-Friedrichs scheme* and it is left as an exercise to show that it is a stable first order method. Therefore, we are still left with a question how to develop higher order schemes? We can for example use the leapfrog method discussed before however, it may be very inconvenient in higher dimensions. We may also devise a trapezoidal scheme but, as we have seen, for hyperbolic equations there is usually no need for introducing another complication like the implicitly defined method (since explicit ones are stable). Another way of improving order is to time-discretize using some RK methods. This requires more storage and careful treatment near the boundaries. One useful way is to consider Taylor series expansions

$$u(x,t+h) = u(x,t) + u_t(x,t)h + \frac{1}{2}u_{tt}(x,t)h^2 + O(h^2), \quad \text{as} \quad h \to 0^+. \tag{6.31}$$

Now, because u is a solution of (6.2) we have $u_t = -cu_x$ and $u_{tt} = c^2 u_{xx}$ and hence

$$u(x,t+h) = u(x,t) - cu_x(x,t)h + \frac{1}{2}c^2 u_{xx}(x,t)h^2 + O(h^2), \quad \text{as} \quad h \to 0^+. \tag{6.32}$$

If we approximate the derivatives with centred differences and drop $O(h^3)$ terms we obtain

$$u_j^{n+1} = u_j^n - \frac{1}{2}\nu\left(u_{j+1}^n - u_{j-1}^n\right) + \frac{1}{2}\nu^2\left(u_{j+1}^n - 2u_j^n + u_{j-1}^n\right), \tag{6.33}$$

which is known as the *Lax-Wendroff scheme*. It is also left as an exercise to show that it is second order stable. Notice that for the first order equation we have introduces a discretization of the *second order derivative*. This adds the so-called artificial diffusion into the scheme and has some strong stabilization properties. Lax-Friedrichs method can also be recast in a way to see the diffusion operator explicitly. This method is robust and can be applied to many other equations.

We end this section with a fundamental convergence theorem for linear constant coefficients numerical methods for PDEs. This is analogous to the Dahlquist theorem for multistep methods (see Theorem 2).

Theorem 9 (Lax). *If a linear constant coefficient finite difference method is applied to a PDE then*

$$\textit{consistency} + \textit{stability} \iff \textit{convergence} \tag{6.34}$$

The order of convergence is the same as of the truncation error.

Therefore, for linear methods we can easily check the consistency and use von Neumann analysis to ascertain stability. The convergence follows from Lax equivalence theorem. The situation is much for difficult for nonlinear equations.

6.2 Finite volume methods

Numerical treatment of a general conservation laws requires a many-volume treatment and is certainly beyond the scope of this lecture. However, we will sketch the overall theory. We will consider possibly nonlinear equations in the form (6.1). Recall from the PDE course that in many situations solutions of nonlinear equations can develop discontinuities even if the initial data is smooth (the so-called shockwaves). This forces to use the integral equation from which the PDE originates and obtain the solution with a help of Rankine-Hugoniot's condition. Of course, the resulting solution has to be interpreted in a weak sense. Certainly these features have to be resolved by a good numerical method.

The essential RH condition that yields the correct shock velocity is a manifestation of the conservation law that can be reobtained by integrating (6.1) from $x = a$ to $x = b$ where a and b are arbitrary

$$\frac{d}{dt}\int_a^b u(x,t)dx = -\int_a^b \left(f(u(x,t))\right)_x dx = f(u(a)) - f(u(b)), \tag{6.35}$$

which states that the total amount of u in $[a, b]$ can change only due to the flux through the boundary. In order to develop a discrete method which has a similar property we introduce a space-time *cell* $[x_{j-1/2}, x_{j+1/2}] \times [t_n, t_{n+1}]$ where are usual $(x_j, t_n) = (jk, nh)$. Here $x_{j\pm1/2} = x_j + k/2$. If we now integrate (6.1) over this cell we have

$$\int_{x_{j-1/2}}^{x_{j+1/2}} \left(u(x,t_{n+1}) - u(x,t_n)\right) dx = -\int_{t_n}^{t_{n+1}} \left(f(u(x_{j+1/2},t)) - f(u(x_{j-1/2},t))\right) dt. \tag{6.36}$$

If we now consider the *cell average*

$$u_j^n := \frac{1}{k}\int_{x_{j-1/2}}^{x_{j+1/2}} u(x,t_n)dx, \tag{6.37}$$

and the so-called *numerical flux*

$$F_{j+1/2}^n := \frac{1}{h}\int_{t_n}^{t_{n+1}} f(u(x_{j+1/2},t))dt, \tag{6.38}$$

we can recast the scheme as

$$u_j^{n+1} = u_j^n - \frac{h}{k}\left(F_{j+1/2}^n - F_{j-1/2}^n\right), \tag{6.39}$$

which is known as the *finite volume method*. The first thing that is worth noticing is that u_j^n now represent the cell average and not an approximation to its point value. Note also that usually, in order to approximate the integral in (6.38) we have to reconstruct the value of u inside some cells. In that case we have

$$F_{j+1/2}^n = F(u_{j-p}^n, \cdots, u_{j+q}^n). \tag{6.40}$$

In order to ascertain consistency we have to assume that $F(u, \cdots, u) = f(u)$, i.e. the numerical flux reduces to the continuous one. As it appears, this requirement is not strong enough and we need the Lipschitz continuity

$$|F(u_{-p}, \cdots, u_q) - f(u)| \le L \max_{-p\le r\le q} |u_r - u| \quad \text{for} \quad |u_r - u| \quad \text{small enough.} \tag{6.41}$$

As a result and by construction (6.39) satisfies the conservation law which is stated in the following result.

Theorem 10. *Finite volume method (6.39) is conservative in the discrete sense.*

Proof. For simplicity we assume that $p = 0$ and $q = 1$ and therefore $F^n_{j+1/2} = F^n_{j+1/2}(u^n_j, u^n_{j+1})$. When we sum the method (6.39) with respect to $J \le j \le K$ with $J < K$ arbitrary we obtain

$$k\left(\sum_{j=J}^{K}\left(u_j^{n+1} - u_j^n\right)\right) = -h\sum_{j=J}^{K}\left(F^n_{j+1/2} - F^n_{j-1/2}\right) = -h\left(F^n_{K+1/2} - F^n_{J-1/2}\right) \tag{6.42}$$

If now, the initial condition ϕ is constant outside some bounded interval then so is u^n_j since explicit methods have finite domain of dependence. Therefore, if we choose J and K sufficiently far apart we obtain

$$k\left(\sum_{j=J}^{K}\left(u_j^{n+1} - u_j^n\right)\right) = -h\left(f(u_\infty) - f(u_{-\infty})\right), \tag{6.43}$$

where by $u_{\pm\infty}$ we have denoted limits of ϕ at $\pm\infty$. Notice also that here we have used the consistency of the numerical flux, i.e. $F(u, \cdots, u) = f(u)$. Now, if we apply the above equality recursively (and translate the indices) we can get

$$k\sum_{j=J}^{K} u^n_j = k\sum_{j=J}^{K} u^0_j - t^n\left(f(u_\infty) - f(u_{-\infty})\right). \tag{6.44}$$

Therefore, if we define the initial step as

$$k\sum_{j=J}^{K} u^0_j = \int_{x_{J-1/2}}^{x_{K+1/2}} \phi(x)dx, \tag{6.45}$$

we recover

$$\int_{x_{J-1/2}}^{x_{K+1/2}} u(x, t_n)dx = \int_{x_{J-1/2}}^{x_{K+1/2}} \phi(x)dx - t^n\left(f(u_\infty) - f(u_{-\infty})\right), \tag{6.46}$$

which precisely is a specific form of the integral conservation law (6.35). □

Knowing that any finite volume method (6.39) is conservative ascertains that the shock speeds will have the correct value and we will not simulate any unphysical mass fluxes. This is of utmost importance in applications. For example, we do not want to compute any artificial winds or ocean currents in weather prediction or electromagnetic fields in engineering.

Therefore, in order to construct a conservative numerical method we have to prescribe a numerical flux. This is many times a very nontrivial task especially if we aim for high resolution methods. More or less, the flux should be given in order to obtain a method which reconstructs u^{n+1}_j based on some cell averages at previous times. It is also a good news that many finite difference methods can be immediately cast into conservative form that can be easily generalized to nonlinear equations or even systems. For example, the upwind method (6.4) can solve a nonlinear equation (6.1) when written as

$$u^{n+1}_j = u^n_j - \frac{h}{k}\left(f(u^n_j) - f(u^n_{j-1})\right), \tag{6.47}$$

however care has to be taken to modify the scheme according to the velocity of the wave (which is $\sim f'$). To see that it gives a conservative method we have to identify the numerical flux. This is simple for that method and here $F^n_{j+1/2} = F(u^n_j) = f(u^n_j)$. A more refined example is the generalization of the Lax-Friedrichs scheme (6.30)

$$u^{n+1}_j = \frac{1}{2}\left(u^n_{j+1} + u^n_{j-1}\right) - \frac{h}{2k}\left(f(u^n_{j+1}) - f(u^n_{j-1}))\right), \tag{6.48}$$

with the flux

$$F^n_{j+1/2} = F(u^n_j, u^n_{j+1}) = \frac{h}{2k}\left(u^n_j - u^n_{j+1}\right) + \frac{1}{2}\left(f(u^n_j) + f(u^n_{j+1})\right). \tag{6.49}$$

Many other finite difference schemes can be shown to be conservative in the above sense. However, this certainly is not the only way of developing numerical methods for conservation laws. It can be shown that Lax-Friedrichs has some serious problems when resolving discontinuous solution, specifically it smears out them due to the diffusive character of the stabilisation component. Other methods introduce oscillations which are even worse since can predict negative values of quantities that should necessarily be positive (for ex. density, energy). Mathematicians have thus worked very hard to overcome these difficulties by finding other ways of devising numerical fluxes. Godunov's, WENO, MUSCL methods are just some examples. Going beyond first order accuracy is even harder since it just makes no sense to use a high order method near discontinuities. Flux- or slope-limiter methods are the ones that weight high and low resolution schemes according to the smoothness of the solution. This is a fascinating subject that an interested reader can find in many voluminous books.

7 Methods for parabolic equations

An archetype for the parabolic PDEs is the heat equation

$$\begin{cases} u_t = u_{xx}, & x \in \mathbb{R}, \quad t > 0, \\ u(x,0) = \phi(x), & x \in \mathbb{R}. \end{cases} \tag{7.1}$$

where for simplicity we consider only one dimensional case in the initial value formulation for the whole real space. Adding boundary conditions is not very difficult however, makes the stability analysis harder. Since finite difference methods are very robust for the one dimensional problem we will focus solely on them. On the other hand, finite element methods are much more convenient for higher dimensional domains and hence we leave their analysis to the next section on elliptic equations.

As always, we begin by discretization of (7.1). The spatial derivative is most conveniently tackled by a symmetric second order finite difference, while for temporal side we have several choices. For example, approximating ∂_t by δ_+ yields *Euler forward method*

$$u_j^{n+1} = u_j^n + \lambda \left(u_{j+1}^n - 2u_j^n + u_{j-1}^n \right), \tag{7.2}$$

where, as usual, u_j^n denotes the numerical approximation to $u(x_j, t_n)$ where $x_j = jk$, and $t_n = nh$. We have also introduced the mesh ratio $\lambda := h/k^2$. This parameter is crucial for stability analysis. Derived method is explicit since u_j^{n+1} depends only on the knowledge of u_j^n at previous time step. Recall that a numerical scheme is stable if it does not increase the norm of the solution when time advances. We can show the following result.

Proposition 2. *Forward Euler method (7.2) for solving heat equation (7.1) is conditionally stable in the uniform norm provided that* $0 < \lambda \leq 1/2$*, that is*

$$\|u^{n+1}\|_\infty \leq \|u^0\|_\infty, \tag{7.3}$$

where $\|u^n\| := \max_{j \in \mathbb{Z}} |u_j^n|$.

Proof. We begin by writing

$$|u_j^{n+1}| = |\lambda u_{j+1}^n + (1-2\lambda)u_j^n + \lambda u_{j-1}^n| \leq \lambda |u_{j+1}^n| + (1-2\lambda)|u_j^n| + \lambda u_{j-1}^n, \tag{7.4}$$

where we have used the assumption. Now, taking supremum over $j \in \mathbb{Z}$ we obtain

$$\|u^{n+1}\|_\infty \leq \lambda \|u^n\|_\infty + (1-2\lambda)\|u^n\|_\infty + \lambda \|u^n\|_\infty = \|u^n\|_\infty. \tag{7.5}$$

Iterating the above finishes the proof. □

This stability result is sharp, that is to say, when we chose $\lambda > 1/2$ we obtain an unstable method.

Example. Choose $v_j^n = (-1)_n^j \epsilon$ where $\epsilon \ll 1$. Then, having $\lambda > 1/2$ and using (7.2) we have

$$v_j^1 = \left[(-1)^j + \lambda \left((-1)^{j+1} - 2(-1)^j + (-1)^{j-1} \right) \right] \epsilon = (1-4\lambda)(-1)^j \epsilon. \tag{7.6}$$

Therefore,

$$v_j^n = (1-4\lambda)^n(-1)^j\epsilon, \tag{7.7}$$

which leads to

$$\|v^n\|_\infty = (4\lambda-1)^n\epsilon \to \infty \quad \text{when} \quad n\to\infty. \tag{7.8}$$

Hence, the norm of the approximation becomes arbitrarily large regardless of the smallness of the initial data even when the final time step $t \le T$ is fixed.

Since we have shown stability, due to Lax equivalence theorem (Theorem 9) in order to show convergence we are left with showing consistency. The truncation error can be calculated by plugging the exact solution of (7.1) into the numerical method (7.2)

$$\tau(x,t) := \frac{u(x,t+h)-u(x,t)}{h} - \frac{u(x+k,t)-2u(x,t)+u(x-k,t)}{k^2}. \tag{7.9}$$

Expanding in the Taylor series we have

$$\tau(x,t) = u_t(x,t) + \frac{1}{2}u_{tt}(x,\eta)h - u_{xx}(x,t) - \frac{1}{12}u_{xxxx}(\xi,t), \tag{7.10}$$

and since $u_t = u_{xx}$ we have $\tau(x,t) = O(h+k^2)$ when $h,k\to 0^+$ provided that u is sufficiently smooth (and from theory of PDEs we known that it is indeed *infinitely* smooth!). Therefore, combining the above with (2) with (9) we arrive at

Theorem 11. *The forward Euler method (7.2) is convergent to the exact solution of the heat equation (7.1) provided that $\lambda \le 1/2$. Moreover, it is first order accurate in time and second in space.*

In practice we usually fix the value of λ along with k and choose h according to $h = \lambda k^2$. Note that in order to provide stability we can obtain prohibitively small values of time grid when we aim for higher spatial accuracy (compare the fact that for hyperbolic equations we had $h = \nu k$ which yields the same order of mesh sizes). Therefore, forward Euler method has a limited practical value.

Exactly as with ODEs we can propose a *Backward Euler Scheme*

$$u_j^{n+1} = u_j^n + \lambda\left(u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}\right), \tag{7.11}$$

which, as can be easily shown, is stable for any $\lambda > 0$. The stability comes for the price of the method being implicit: in each time step we have to solve a system of linear equations. Since computers cannot cope with infinite matrices we impose some artificial bounds

$$u_j^n = 0 \quad \text{for} \quad |j| \ge J, \quad J\in\mathbb{N}, \quad j\in\mathbb{Z}. \tag{7.12}$$

Then, (7.11) can written in the form

$$-\lambda\left(u_{j+1}^{n+1} + u_{j-1}^{n+1}\right) + (1+2\lambda)u_j^{n+1} = u_j^n, \tag{7.13}$$

and the system can be cast into

$$A\mathbf{u}^{n+1} = \mathbf{u}^n, \tag{7.14}$$

where A is a $(2J-2)\times(2J-2)$ matrix

$$A = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & \cdots & 0 \\ -\lambda & 1+2\lambda & -\lambda & \cdots & 0 \\ \vdots & \cdots & \cdots & \cdots & \vdots \\ 0 & \cdots & -\lambda & 1+2\lambda & -\lambda \\ 0 & \cdots & \cdots & -\lambda & 1+2\lambda \end{bmatrix}. \tag{7.15}$$

Note that A is a tridiagonal matrix and therefore can be very easily and cheaply inverted numerically (notice that Gaussian elimination requires $O(N^3)$ operations where $N\times N$ is the size of the matrix, whereas tridiagonal algorithm requires only $O(N)$).

Having a stable and relatively cheap method of solving one dimensional heat equation one would like to aim for some more accuracy. Unfortunately, backward Euler method is only first order in time. As always, there are many ways to achieve it and probably the most versatile and useful is the following *Crank-Nicolson scheme*

$$u_j^{n+1} = u_j^n + \lambda\left(\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{2} + \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{2}\right), \tag{7.16}$$

which can be conveniently written as

$$-\frac{\lambda}{2}\left(u_{j+1}^{n+1} + u_{j-1}^{n+1}\right) + (1+\lambda)\,u_j^{n+1} = \frac{\lambda}{2}\left(u_{j+1}^n + u_{j-1}^n\right) + (1-\lambda)\,u_j^n. \tag{7.17}$$

The above is manifestly a tridiagonal system with the same matrix as before, i.e (7.15). The second order both in space and time accuracy of C-N method is ascertained by the averaging of the spatial derivatives with respect to time. That is to say, we introduce a symmetry around $t_{n+1/2}$ point in time. Moreover, we have the following result.

Theorem 12. *The Crank-Nicolson method (7.16) is stable with respect to the $\|\cdot\|_{2,k}$ norm and second order accurate in space and time. Therefore, it is convergent to the exact solution of (7.1).*

Proof. We start with finding the truncation error

$$\tau(x,t) = \frac{u(x,t+h) - u(t)}{h} - \frac{1}{2}\left(\partial_+\partial_- u(x,t+h) + \partial_+\partial_- u(x,t)\right), \tag{7.18}$$

where for brevity we have denoted the spatial finite differences by $\delta_\pm$. Expanding the first term gives

$$\frac{u(x,t+h)-u(t)}{h} = u_t(x,t) + u_{tt}(x,t)h + \frac{1}{6}u_{ttt}(x,\eta_1) = u_t(x,t) + u_{xxxx}(x,t)h + \frac{1}{6}u_{ttt}(x,\eta_1), \tag{7.19}$$

where we have used the original PDE (7.1) yielding $u_{tt} = u_{xxxx}$. Further,

$$\partial_+\partial_- u(x,t+h) = \left(u_{xx}(x,t) + u_{xxt}(x,t)h + \frac{1}{2}u_{xxtt}(x,\eta_2)h^2\right) + \frac{1}{12}u_{xxxx}(\xi_1,t+h)k^2, \tag{7.20}$$

where the term in the parentheses is the Taylor expansion of $u_x x(x, t+h)$ for small $h > 0$. Finally,

$$\partial_+\partial_- u(x,t) = u_{xx} + \frac{1}{12}u_{xxxx}(\xi_2, t)k^2. \tag{7.21}$$

Now, we plug the above expansions in the truncation error and because $u_{xxt} = u_{xxxx}$ the $O(h)$ terms cancel leaving

$$\tau(x,t) = \left(\frac{1}{6}u_{xxxxxx}(x,\eta_1) - \frac{1}{4}u_{xxxxxx}(x,\eta_2)\right)h^2 - \frac{1}{12}\left(u_{xxxx}(\xi_1, t+h) + u_{xxxx}(\xi_2, t)\right)k^2. \tag{7.22}$$

Therefore, if u is smooth enough (and we know it is) we obtain

$$|\tau(x,t)| \leq C(k^2 + h^2) \quad \text{for} \quad h, k \to 0^+, \tag{7.23}$$

which states that the method is second order accurate.

Now, we turn to the $\|\cdot\|_{2,h}$ stability. To this end we use von Neumann analysis. Fourier expanding (7.16) yields

$$\widehat{u}^{n+1}(\xi) = \widehat{u}^n(\xi) + \frac{\lambda}{2}\left(e^{-ik\xi} - 2 + e^{ik\xi}\right)\widehat{u}^{n+1}(\xi) + \frac{\lambda}{2}\left(e^{-ik\xi} - 2 + e^{ik\xi}\right)\widehat{u}^n(\xi). \tag{7.24}$$

Simplifying,

$$\widehat{u}^{n+1}(\xi) = \frac{1 - \lambda(1 - \cos(k\xi))}{1 + \lambda(1 - \cos(k\xi))}\widehat{u}^n(\xi), \tag{7.25}$$

and therefore the amplification factor is

$$g(\xi) = \frac{1-z}{1+z} = 1 - \frac{2z}{1+z} \quad \text{where} \quad z = \lambda(1 - \cos(k\xi)). \tag{7.26}$$

Since $z \geq 0$ we obviously have $|g(\xi)| \leq 1$ for any $\lambda > 0$. This yields stability and applying Theorem 9 forces convergence. $\square$

It is also possible to investigate the stability of C-N scheme in the $\|\cdot\|_{\infty,h}$ norm however, the analysis is more involved especially in the $\lambda > 1$ case. The strong stability properties of C-N method along its moderately cheap computational cost makes it the method of choice for solving one dimensional heat conduction problems with or without boundary conditions. It also generalizes very easily to nonhomogeneous equations and even nonlinear ones. However, in the latter case the analysis is much more difficult.

8 Methods for elliptic equations

In this section we will investigate numerical methods for solving elliptic PDEs which are represented by Poisson's equation

$$\begin{cases} -\Delta u = f, & x \in D, \\ u = g, & x \in \partial D, \end{cases} \tag{8.1}$$

where the domain $D \subseteq \mathbb{R}^2$ can be bounded or unbounded (higher dimensional cases are also possible). In the homogeneous case, i.e. $f \equiv 0$, we obtain the Laplace's equation. It is good to think about the above PDE as a steady-state heat distribution or potential generated by a charge distributed according to f. We consider two ways of discretizing (8.1): finite differences and finite elements.

8.1 Finite difference methods

This family of schemes is most useful for rectangular domains or the whole two-dimensional space. We assume that $D = [0,1]^2$ with the uniform grid $x_i = ih$ and $y_j = jh$ with $1 \leq i, j \leq n$, that is $h = 1/(n+1)$. The discretization of the Laplace operator can be done in many ways, the simplest one is the so-called 5-point Laplacian

$$\frac{1}{h^2}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}\right) = f_{i,j}, \tag{8.2}$$

where $u_{i,j}$ is the approximation to $u(x_i, y_j)$. This 5-point stencil uses all the values of u around (x_i, y_j). The above is just a sum of two centred finite differences with respect to x and y. We thus arrive at n^2 equations to be solved for $u_{i,j}$. The boundary values, for $i, j = 0$ or $i, j = n+1$, can be moved to the right-hand side since they are represented by g. There are many ways of putting the above scheme into matrix form in each of them we would obtain a very sparse matrix, i.e. having almost all entries equal to zero. In two dimensions there is no such a convenient way to order the unknowns and hence, no cheap algorithm to perform the inversion. For example, if one would blindly like to use Gaussian elimination they would perform $O(n^6)$ operations and which would be disastrous even for fast computers (say take modest $n = 100$). If we use some more sophisticated methods that take sparsity into account we can reduce the number of operations to $O(n^3)$ and it has been proved to be the best we can hope for when using Gaussian elimination.

Another possibility of solving the system (8.2) is to resign from aiming for exact methods. That is to say, we use some iterative linear system solving method which improved accuracy in each iteration. This proves to be very fast and reliable. We give a short account of two classical methods that, although converge very slowly, help to build some more sophisticated modern algorithms.

To begin, write (8.2) in the form

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}\right) - \frac{h^2}{4} f_{i,j}. \tag{8.3}$$

Note that for Laplace's equation the above is a discrete version of the well-known fact that a harmonic function has the mean-value property, i.e. its value at (x, y) is an averaged of values around that point. The above formula suggest the following iteration

$$u_{i,j}^{(m+1)} = \frac{1}{4}\left(u_{i-1,j}^{(m)} + u_{i+1,j}^{(m)} + u_{i,j-1}^{(m)} + u_{i,j+1}^{(m)}\right) - \frac{h^2}{4} f_{i,j}, \tag{8.4}$$

where the superscript denotes the number of iterations. This is known as the *Jacobi method* and it can be shown to converge to the solution of (8.2) (which itself is an approximation to (8.1)). This algorithm can be refined by noticing that $u_{i-1,j}$ and $u_{i,j-1}$ are already found when we try to update $u_{i,j}$. This leads to the *Gauss-Seidel method*

$$u_{i,j}^{(m+1)} = \frac{1}{4}\left(u_{i-1,j}^{(m+1)} + u_{i+1,j}^{(m)} + u_{i,j-1}^{(m+1)} + u_{i,j+1}^{(m)}\right) - \frac{h^2}{4} f_{i,j}, \tag{8.5}$$

which converges twice fast as the Jacobi iteration. These methods can be upgraded to Successive Over-Relaxation (SOR) iteration or, instead, we can use Steepest Descent or Conjugate Gradient methods. They all are very well described in numerical analysis literature.

8.2 Finite element methods

As we have seen, finite difference methods yield an interesting algebraic system to solve. By construction, they are mostly limited to rather simple domains (however, there are ways of extending their applications to irregular geometries). An easier approach is to use finite element methods. By multiplying (8.1) with a sufficiently smooth function we obtain the weak formulation

$$a(u,v) = (f,v), \quad v \in C_0^1(D), \tag{8.6}$$

where

$$a(u,v) = \int_D \nabla u \cdot \nabla v \, dx, \quad (u,v) = \int_D u v \, dx. \tag{8.7}$$

We assume that D is a convex polygon which implies the regularity estimate (see theory of PDEs)

$$\|u\|_2^2 + \|\nabla u\|_2^2 \le C\|f\|^2. \tag{8.8}$$

Next, as with the one-dimensional case, we have to divide the domain D into subsets on which it is easy to introduce a basis. Usual choice is to define a *triangulation* such that

$$\bar{D} = \bigcup_{K \in \mathcal{T}_h} K, \quad h_K = \operatorname{diam}(K), \quad h = \max_{K \in \mathcal{T}_h} h_K, \tag{8.9}$$

where $\mathcal{T}_h = \{K\}$ is a set of closed triangles. The vertices of these triangles are called nodes and we require that an intersection of two of them is either empty, a node, or a common edge. That is, there are no nodes located in the interior of an edge. With this triangulation we associate a function space

$$S_h = \left\{ v \in C(\bar{D}) : v \text{ is linear in } K \text{ for each } K \in \mathcal{T}_h, \ v = 0 \text{ on } \partial D \right\}. \tag{8.10}$$

One of the simplest examples are pyramid functions which are generalizations of the previously met hat functions. If $\{P_i\}_i^N$ is the set of interior nodes we define the basis as a piecewise linear functions

$$\varphi_i(P_j) = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{8.11}$$

The finite element approximation to (8.6) is thus

$$\text{find } u_h \text{ such that } a(u_h, \varphi) = (f, \varphi) \text{ for all } \varphi \in S_h. \tag{8.12}$$

We can write $u_h(x) = \sum_i u_i \varphi_i(x)$ and plug it in (8.12) to obtain

$$\sum_i u_i a(\varphi_i, \varphi_j) = (f, \varphi_j) \quad j = 1, \cdots, N. \tag{8.13}$$

Again, this can be easily written as a matrix equation $A\mathbf{u} = \mathbf{b}$ with the solution vector, $\mathbf{u} = (u_i)_i$, load vector $\mathbf{b} = ((f, \varphi_i))_i$ and the stiffness matrix $A = (a(\varphi_i, \varphi_j))_i$. The matrix A is symmetric and positive definite and hence possesses a unique solution. Moreover, it is very sparse and hence, allows for some efficient solvers. Similarly as we did in the section on one-dimensional boundary value problems, we can show that

$$\|u_h - u\|_2 \leq C\|f\|_2 h^2, \tag{8.14}$$

that is, the method is of second order. To sum up, if we have a irregular domain D we have to generate its triangulation $\mathcal{T}_h$ which might not be a trivial task. Then, we generate entries of the stiffness matrix and solve the linear system (8.13) with a possibly iterative method.

Both the finite difference method and finite element method can be used in solving time dependent problems in higher dimensions (parabolic or hyperbolic). Usually first we discretize space according to them, and then apply some time marching scheme such as Euler, R-K, or similar. In each time step we solve the linear system and then update for the new time value. Possibilities are many and this is certainly not a trivial exercise since we always have to take into account computational and storage costs. However, this is another story...