4. Numerical solutions of PDE

There are very few PDE that can be solved analytically. Therefore, it is important to understand the techniques for solving PDE on a computer, but also the difficulties and pitfalls that can arise when we try to do so. Let us start with a simpler setting.

4.1. **Ordinary differential equations.** Consider the ordinary differential equation

$$\partial_t y(t) = F(y(t), t), \qquad y(t_0) = y_0. \tag{4.1}$$

The simplest way to solve this equation on a computer is the *forward Euler scheme:*
Step 1: Discretize the t-axis with step size h, giving grid points $t_0, t_0+h, t_0+2h, \ldots$.
Step 2: Use Taylor-expansion to find that a solution of (4.1) fulfils

$$y(t+h) = y(t) + h\partial_t y(t) + \mathcal{O}(h^2) = y(t) + hF(y(t), t) + \mathcal{O}(h^2),$$

where the notation $\mathcal{O}(h^2)$ means the following: it is possible to find a function that, when written in place of the symbol $\mathcal{O}(h^2)$, ensures that the equality sign is a true statement. This function may depend on $t, y(t), h$ or whatever other parameters there are, but it must vanish at least as quickly as a constant times h^2 (or whatever expression we write into brackets after the $\mathcal{O}$), when $h \to 0$.
Step 3: Recall that the initial value of the ODE is y_0. Define

$$y_1 = y_0 + hF(y_0, t_0),\ y_2 = y_1 + F(y_1, t_2),\ \ldots,\ y_{n+1} = y_n + F(y_n, t_n).$$

The values y_j should approximate the values $y(t_j)$ of the true solution; after all, they solve the difference equation from Step 2 that is an approximation to the ODE.

The second step above was a bit arbitrary. We could as well have used

$$y(t-h) = y(t) - h\partial_t y(t) + \mathcal{O}(h^2) = y(t) - hF(y(t), t) + \mathcal{O}(h^2),$$

which would have led to the scheme

$$y_0 = y_1 - hF(y_1, t_1), y_1 = y_2 - hF(y_2, t_2), \ldots$$

This is the *backwards Euler scheme*, or implicit Euler scheme. In each step we now still have to solve a (possibly difficult) equation to obtain the value of y_{j+1} as a function of the value y_j and t_j. It is not clear at this moment why anyone would like to do such a thing, but we will see later when we treat PDE that there are large benefits in doing so.
However we do it, we hope that the values $y_1, y_2, \ldots$ approximate $y(t_1), y(t_2), \ldots$, where $y(t)$ is the true solution. We now want to quantify how good that approximation is. For this, consider a finite time interval $[t_0, T]$, and put $t_j = t_0 + hj$, with N such that $t_N = T$. Let y solve (4.1), and let y_n be the approximating solution under some numerical scheme. We define

$$e_j = |y_j = y(t_j)| = \text{ the error we make at time } t_j,$$

and

$$E = \max_{0 \leqslant j \leqslant N} e_j = \text{ the worst error we make on the interval.}$$

Definition: A numerical scheme is called *convergent* if $\lim_{h\to 0} E(h) = 0$. It is called *convergent of order p* if $E(h) \leqslant Ch^p$ for some $C > 0$ and all h small enough.

How can we check convergence? A good indicator would be how similar our numerical scheme is to the true ODE, for small h. Assume our numerical scheme is given by

$$y_{n+1} = y_n + h\phi(y_n, t_n, h) \tag{4.2}$$

for some function ϕ that may depend on y_n, t_n but also on h. In the forward Euler scheme we had $\phi(y_n, t_n, h) = F(y_n, t_n)$, so in that case ϕ was independent of h. Now let again $y(t)$ be the true solution of the ODE.

Definition: The *truncation error* $\tau_n(h)$ at step n is defined by the equation

$$y(t_{t+1}) = y(t_n) + h\Big(\phi(y(t_n), t_n, h) + \tau_n(h)\Big).$$

Interpretation: the true solution does not fulfil the recursive scheme (4.2), but instead fulfils another recursive scheme. $\tau_n(h)$ measures how different the two schemes are at time t_n and for discretisation paramter h.

Definition: A numerical scheme is called *consistent* if

$$\lim_{h\to 0} \Big(\max_{0 \leqslant n \leqslant N} \tau_n(h) \Big) = 0.$$

It is called *consistent of order p* if

$$\max_{0 \leqslant n \leqslant N} \tau_n(h) \leqslant Ch^p$$

for some $C > 0$ and all sufficiently small $h > 0$.

So, consistency means that the two *equations* become similar as $h \to 0$, while convergence means that the two *solutions* become similar. You should check that explicit Euler is consistent of order 1.
What about the connections between consistency and convergence? For ODE, the connection is rather simple.
Theorem: For a numerical scheme given by (4.2) let us assume that

$$\Big|\phi(y, t, h) - \phi(\tilde{y}, t, h)\Big| \leqslant L|y - \tilde{y}|, \tag{4.3}$$

for some $L > 0$, all $t \leqslant T$ and all $h < h_0$ with some $h_0 > 0$. (This is called the *Lipschitz condition*.) Assume that the scheme is consistent of order p. Then it is also convergent of order p.

Proof. On the exercise sheet you are asked to prove the following fact (discrete Gronwall Lemma): For nonnegative numbers $z_1, z_2, \dots$ assume that there are $C, D > 0$ such that $z_{n+1} \leqslant Cz_n + D$ for all n. Then

$$z_n \leqslant D\frac{C^n - 1}{C - 1} + z_0 C^n \tag{4.4}$$

for all n. We use this fact in the following argument: Since

$$\begin{aligned} y_{n+1} &= y_n + \phi(y_n, t_n, h) \\ y(t_{n+1}) &= y(t_n) + h\phi(y(t_n), t_n, h) + h\tau_n(h), \end{aligned}$$

we find

$$\begin{aligned} e_{n+1} = |y_{n+1} - y(t_{n+1})| &= e_n + h\Big(\phi(y_n, t_n, h) - \phi(y(t_n), t_n, h)\Big) + h\tau_n(h) \\ &\leqslant e_n + hL\Big|y_n - y(t_n)\Big| + h\tau_n(h) = (*). \end{aligned}$$

The inequality above is due to our assumption (4.3). Now we have assumed consistency of the scheme of order p, thus there is an $M > 0$ such that $\tau_n(h) \leqslant Mh^p$ for all n and all small enough h. Also, $|y_n - y(t_n)| = e_n$. So $(*) \leqslant e_n(1+hL) + Mh^{p+1}$, and by (4.4) with $C = 1 + hL$ and $D = Mh^{p+1}$, we get (notice that $e_0 = 0$)

$$e_n \leqslant Mh^{p+1}\frac{(1+hL)^n - 1}{1 + hL - 1} = Mh^p\frac{1}{L}\Big((1+hL)^n - 1\Big).$$

Finally,

$$\begin{aligned} 0 \leqslant (1+hL)^n - 1 &\leqslant \left(1 + hL + \frac{(hL)^2}{2} + \frac{(hL)^3}{3!} + \dots\right) - 1 \\ &= \left(\mathrm{e}^{hL}\right)^n - 1 = \mathrm{e}^{hLn} - 1 \leqslant \mathrm{e}^{(T-t_0)L} - 1, \end{aligned}$$

where in the last equality we used that h discretizes the interval $[t_0, T]$, so for $n < N = N(h)$, hn can never be larger that $[t_0, T]$. □

Before leave the ODE case and look at numerics for PDE, let us give an example that shows how implicit schemes can be useful even for ODE. Consider the simple equation

$$\partial_t y(t) = -\lambda y(t), \qquad \lambda > 0, y(0) = y_0.$$

The solution is of course $y(t) = y_0\,\mathrm{e}^{-\lambda t}$, and it decays to 0 rapidly and monotonously as $t \to \infty$. Can we say the same for our numerical schemes? Let's look at the Euler forward scheme:

$$y_{n+1} = y_n + h\lambda y_n = (1 - \lambda h)y_n, \qquad \Longrightarrow \qquad y_n = (1 - h\lambda)^n y_0.$$

If λ is large, we will need h to be small for this scheme to give something sensible: indeed, $y_n \to 0$ as $n \to \infty$ *only* if $h \leqslant 2/\lambda$; and, $n \mapsto y_n$ is monotone decreasing *only* if $h \leqslant 1/\lambda$. So, while the true ODE becomes more and more easy to understand when λ gets large, the numerical scheme becomes more difficult to implement in the sense that the time step needed for a sensible solution becomes smaller. After

all, h will always be finite in real world situations, and the smaller we have to take it, the longer we need to run a computer in order to find $y(10)$, say.
The situation is different for the Euler backward scheme. In that scheme, we find

$$y_n = y_{n+1} + h\lambda y_{n+1} \qquad \Longrightarrow \qquad y_n = \frac{1}{(1+h\lambda)^n} y_0,$$

which is monotonously decreasing to zero regardless of the size of h. We say that the forward Euler scheme is *conditionally stable* (i.e. for small enough h it is stable), while the backward scheme is *unconditionally stable*. A proper definition of what a stable scheme means will be given below when we treat PDE.

4.2. **Forward schemes for PDE.** We consider the general PDE

$$\begin{aligned} &\partial_t u(x,t) = (Fu)(x,t) \qquad (a < x < b, t > 0), \\ &u(x,t_0) = u_0(x) \qquad \text{(initial condition)}, \\ &u(a,t) = g_a(t), u(b,t) = g_b(t) \qquad \text{(boundary conditions)}. \end{aligned} \tag{4.5}$$

Above, F can be any linear or nonlinear operator, such as

$$\begin{aligned} Fu &= \frac{\sigma^2}{2}\partial_x^2 u \qquad \text{(heat equation)}, \\ Fu &= -\frac{1}{2}\sigma^2 x^2 \partial_x^2 u + b(x\partial_x u - u) \qquad \text{(Black-Scholes PDE)} \\ Fu &= -\max_{\alpha \in A}\left(f(x,\alpha)\partial_x u(x,t) + h(x,\alpha) + \frac{1}{2}\sigma^2(x,\alpha)\partial_x^2 u \right) \qquad \text{(HJB equation)}. \end{aligned}$$

We now want to solve these numerically, so we discretize time as $t_n = t_0 + hn$ and space as $x_j = h_x j$ with $n \in \mathbb{N}$ and $j \in \mathbb{Z}$. h_x can and usually will be different from h, so we do not discretize the space-time domain into squares but rather rectangles. For discretizing the spatial derivatives we use again Taylor expansion. One possibility is

$$\partial_x u(x_j,t_n) \approx \frac{1}{h_x}\Big(u(x_{j+1},t_n) - u(x_j,t_n)\Big).$$

This can be useful if there is a preferred direction of space, but for the heat equation and related equations, there is none, and so the symmetric finite difference

$$\partial_x u(x_j,t_n) \approx \frac{1}{2h_x}\Big(u(x_{j+1},t_n) - u(x_{j-1},t_n)\Big)$$

is usually better. Whether we best use central of ordinary finite differences, or yet another approximation, depends on the equation and is more an art than a science. We will not go into this here and only use central spatial differences. The second derivative is thus approximated by

$$\partial_x^2 u(x_j,t_n) \approx \frac{1}{h_x^2}\Big(u(x_{j+1},t_n) + u(x_{j-1},t_n) - 2u(x_j,t_n)\Big).$$

Higher derivatives can be disrcetized similarly, but we will not need them here. Plugging the discretized derivatives into Fu gives

$$Fu(x_j,t_n) \approx \tilde{F}\Big(u(x_{j+1},t_n),u(x_j,t_n),u(x_{j-1},t_n),t_n,h_x\Big),$$

for some function $\tilde{F}$, in the case where we have only second derivatives. For higher derivatives, the function on the right hand side above will depend on more values of $u(x,t_n)$, but we will not need this here. As an example, for the heat equation we find

$$\tilde{F}\Big(u(x_{j+1},t_n),u(x_j,t_n),u(x_{j-1},t_n),t_n,h_x\Big) = \frac{\sigma^2}{2h_x^2}\Big(u(x_{j+1},t_n)+u(x_{j-1},t_n)-2u(x_j,t_n)\Big).$$

The approximation to the PDE then becomes

$$u(x_j,t_{n+1}) \approx u(x_j,t_n) + h\tilde{F}\Big(u(x_{j+1},t_n),u(x_j,t_n),u(x_{j-1},t_n),t_n,h_x\Big), \tag{4.6}$$

and the numerical scheme is

$$u_{n+1}^j = u_j^n + h\tilde{F}(u_{j+1}^n,u_j^n,u_{j-1}^n,t_n,h_x),$$

with $u_j^0 = u_0(x_j)$ as initial condition and $u_j^n = g_{a,b}(t_n)$ if $x_j = a,b$, respectively, as boundary conditions. u_j^n are again what we would like to be approximations to $u(x_j,t_n)$. In our case, it is easy to compute the point $u(x_j,t_n)$; it is a known function (namely, $\tilde{F}$) of the points $u(x_{j-1},t_n)$, $u(x_{j+1},t_n)$, $u(x_j,t_n)$ from the previous time discretisation point, and so we can recursively get all values.

Of course, the question is again how good this approximation actually is. As in the ODE case, we define the *truncation error* that measures how different the numerical scheme is from the actual PDE:

Definition: The *truncation error* at point (x_j,t_n) of the numerical scheme (4.6) is defined to be

$$T(x_j,t_n) = \frac{1}{h}\Big(u(x_j,t_{n+1})-u(x_j,t_n)\Big)-\tilde{F}\Big(u(x_{j+1},t_n),u(x_j,t_n),u(x_{j-1},t_n),t_n,h_x\Big),$$

i.e. the extent to which the true solution does not solve the approximate PDE (difference equtaion).

Definition: The scheme is *consistent* if $\lim T(x,t) = 0$ for all x and t as h_x and h go to zero. Here, if we want to be very precise, we need to define $T(x,t)$ as the limit of $t(x_j,t_n)$ for sequences (t_j) and (x_n) of grid points with $x_j \to x$ and $t_n \to t$ as both h and h_x go to zero.

As for ODE, consistency is the minimal requirement that we have for any numerical scheme. Let us check consistency for the forward scheme of the heat equation: we have seen that in this case,

$$u_j^{n+1} = u_j^n + \frac{\sigma^2}{2}\frac{h}{h_x^2}(u_{j+1}^n + u_{j-1}^n - 2u_j^n),$$

so the truncation error is

$$T(x_j,t_n) = \frac{1}{h}\Big(u(x_j,t_{n+1}) - u(x_j,t_n)\Big) - \frac{\sigma^2}{2}\frac{1}{h_x^2}\Big(u(x_{j-1},t_n) + u(x_{j+1},t_n) - 2u(x_j,t_n)\Big)$$
$$\to \partial_t u(x,t) - \frac{\sigma^2}{2}\partial_x^2 u(x,t) = 0$$

as $h, h_x \to 0$, and $x_j \to x$, $t_n \to t$. So, the scheme is consistent. Is it convergent? There seems to be little hope unless $h \leqslant Ch_x^2$ for some $C > 0$ since otherwise the prefactor h/h_x^2 for getting from one time step to the next would diverge as $h \to 0$. But as we will see in the next section, even this condition does not ensure convergence.