Time-dependent problems

Many partial differential equations involve time. We can approximate these equations using finite-difference schemes. Let's ask ourselves, as time progresses, will our solution diverge? That's mainly the subject of this chapter.

1 Time integration methods

When dealing with a partial differential equation, we first need an integration scheme. As always, there are multiple kinds of schemes. Let's have a look at them first.

1.1 Explicit methods

Let's consider the so-called **linear advection equation** $u_t = -au_x$, with a a constant. To apply integration methods, we need an iteration scheme. One such scheme is the **Forward Euler** (also called **explicit Euler**), being

$$\frac{\mathbf{u_{n+1}} - \mathbf{u_n}}{\Delta t} = F(\mathbf{u_n}) \qquad \text{or, equivalently,} \qquad \mathbf{u_{n+1}} = \mathbf{u_n} + \Delta t\, F(\mathbf{u_n}). \tag{1.1}$$

(Note that we have used vectors. The vector $\mathbf{u_n}$ contains all data points at time level n.) Here the function $F(\mathbf{u})$ is just some finite-difference scheme of $-au_x$; the non-time dependent part of the equation. (We don't really care what kind of function it is at the moment.) The important thing is the following: The solution of time level $n+1$ is expressed as a function of earlier time levels. (It is thus an **explicit method**.) Finding the solution of time level $n+1$ is just a matter of calculating our equations.

1.2 Implicit methods

However, we could also have considered the **Backward Euler** (also called **implicit Euler**) scheme, being

$$\frac{\mathbf{u_{n+1}} - \mathbf{u_n}}{\Delta t} = F(\mathbf{u_{n+1}}) \qquad \text{or, equivalently,} \qquad \mathbf{u_{n+1}} = \mathbf{u_n} + \Delta t\, F(\mathbf{u_{n+1}}). \tag{1.2}$$

Now the solution at time level $n+1$ also depends on the solution at time level $n+1$. Any method with that condition is called an **implicit method**. To find a solution, we need to apply matrix algebra, which involves a lot of difficult computations. However, implicit methods often have an advantage on explicit methods, as we will see in the part on stability.

1.3 Combing explicit and implicit methods

It is also possible to combine the two schemes above. We then get the **θ-method**, being

$$\frac{\mathbf{u_{n+1}} - \mathbf{u_n}}{\Delta t} = \theta F(\mathbf{u_{n+1}}) + (1-\theta)F(\mathbf{u_n}), \tag{1.3}$$

where $0 \leq \theta \leq 1$. This method is most accurate when $\theta = 1/2$. If we in fact have set $\theta = 1/2$, then this method is called the **Crank-Nicolson scheme**.

1.4 Multi-step methods

The methods from the previous paragraphs weren't very accurate. If we decrease our time step Δt by a factor 2, then our error is also halved. We can do better than that. One way to find a more accurate

solution is by using a **multi-step method**. In a multi-step method, not only time levels $n+1$ and n are used in our equation. Also earlier time levels are used. An example is the previously discussed Leapfrog method

$$\frac{\mathbf{u_{n+1}} - \mathbf{u_{n-1}}}{2\Delta t} = F(\mathbf{u_n}) \qquad \text{or} \qquad \frac{\mathbf{u_{n+1}} - \mathbf{u_{n-1}}}{2\Delta t} = F(\mathbf{u_{n+1}}). \tag{1.4}$$

We have written both the explicit and the implicit form here. As you can see, the above example uses three time levels. It is of course also possible to use more time levels. Although this increases the accuracy, it also complicates your calculations.

1.5 Multi-stage methods

Another way to increase accuracy, is by using **multi-stage methods**. We now don't use solutions from previous time levels now. However, between two time steps, we just create some intermediate stages. An example of a multi-stage method is the **Runge-Kutta 3** (RK3) method. This method has three intermediate stages $\mathbf{u_{(1)}}$, $\mathbf{u_{(2)}}$ and $\mathbf{u_{(3)}}$. It proceeds as

$$\begin{aligned}
\mathbf{u_{(1)}} &= \mathbf{u_n}, & (1.5)\\
\mathbf{u_{(2)}} &= \mathbf{u_n} + \Delta t F\left(\mathbf{u_{(1)}}\right), & (1.6)\\
\mathbf{u_{(3)}} &= \mathbf{u_n} + \Delta t \left(\frac{1}{4}F\left(\mathbf{u_{(1)}}\right) + \frac{1}{4}F\left(\mathbf{u_{(2)}}\right)\right), & (1.7)\\
\mathbf{u_{n+1}} &= \mathbf{u_n} + \Delta t \left(\frac{1}{6}F\left(\mathbf{u_{(1)}}\right) + \frac{1}{6}F\left(\mathbf{u_{(2)}}\right) + \frac{2}{3}F\left(\mathbf{u_{(3)}}\right)\right). & (1.8)
\end{aligned}$$

Once more, the function $F(u)$ is a finite-difference scheme for the non-time dependent part of the differential equation.

2 Stability

Just like in the previous chapter, we want to know whether our methods are stable. Do we actually get the right solution? How can we find that out?

2.1 Von Neumann stability analysis

Let's apply the von Neumann stability analysis. For that, we have to assume we can write some data point u_i^n as

$$u_i^n = E^n e^{Ii\phi}. \tag{2.1}$$

By the way, we now have $I = \sqrt{-1}$ as the complex number. We can insert the above relation into our finite-difference scheme. We then need to substitute for the **amplification factor** G, being

$$G = \frac{E^{n+1}}{E^n}. \tag{2.2}$$

We will then find a polynomial with G (often denoted as $P(G)$). This polynomial is called the **characteristic polynomial**. We can solve this relation for G. It often occurs that we find multiple solutions. (Note that these solutions may be complex as well.) Let's denote these m solutions as $\sigma_1, \ldots, \sigma_m$. Our scheme is only **stable** if $|\sigma| \leq 1$ for all solutions σ. If there is a single σ for which $|\sigma| > 1$, then the scheme is **unstable**.

Sometimes there are also certain undetermined parameters in our problem. It may occur that our scheme is only stable if these parameters lie in a certain range. If this is the case, then our problem is

conditionally stable. You then need to find the allowed range for these parameters. If there is only one (possibly complex) parameter λ, then we can even visualize this range. We just take the complex plane, and mark every point λ on it for which the problem is stable. The area we then have marked is called the **stability region**.

2.2 Phase and diffusion error

Let's suppose our method is stable. This doesn't mean there are no errors. Probably there will still be a difference between the amplification factor G of our integration method, and the actual (exact) amplification factor $\tilde{G}$. To examine this difference, we assume that we can write

$$G = e^{-I\omega t} \qquad \text{and} \qquad \tilde{G} = e^{-I\tilde{\omega}t}, \tag{2.3}$$

where $\omega = \xi + \eta I$. We now examine the **amplitude** $|G|$ and the **phase** Φ. They can be found using

$$|G| = e^{\eta t} = \sqrt{Re(G)^2 + Im(G)^2} \qquad \text{and} \qquad \Phi = \xi t = \arctan\left(\frac{Im(G)}{Re(G)}\right), \tag{2.4}$$

where $Re(G)$ is the real part of G and $Im(G)$ is the imaginary part. Similarly, we can find $|\tilde{G}|$ and $\tilde{\Phi}$.

First let's examine the **diffusion error**. It is defined as

$$\epsilon_D = \frac{|G|}{|\tilde{G}|}. \tag{2.5}$$

If $\epsilon_D < 1$, then there is dissipation/diffusion.

We can also examine the **phase error**. It has two definitions. One definition is

$$\epsilon_\phi = \Phi/\tilde{\Phi}. \tag{2.6}$$

However, in some cases $\tilde{\Phi} = 0$. In this case the above definition isn't very suitable. We then often use

$$\epsilon_\phi = \Phi - \tilde{\Phi}. \tag{2.7}$$

If the solution from our scheme is running behind the exact solution (so if $\Phi < \tilde{\Phi}$), then there is a **lagging phase error**. If $\Phi > \tilde{\Phi}$, then there is a **leading phase error**.

2.3 The matrix method

There is a downside to the von Neumann stability analysis. It does not take into account boundary conditions. And sometimes the system only becomes unstable for certain boundary conditions. A method that does take into account boundary conditions, is the **matrix method**. It is, however, only applicable to linear partial differential equations.

Let's suppose we have a partial differential equation $\partial_t u/\partial t = F(u)$, with F some linear x-dependent function. If we apply a finite-difference scheme to this equation, we can write it as

$$\delta_t \mathbf{u} = S\mathbf{u} + \mathbf{r}. \tag{2.8}$$

Here $\mathbf{u}$ is the vector with all the data point u_i. The vector $\mathbf{r}$ contains the boundary conditions. To perform the matrix method, we have to examine the eigenvalues λ_j of S. Our method is only stable if $Re(\lambda_j) < 0$ for all eigenvalues λ_j.