10.5 Time discretization

The simplest discretization for the heat equation uses a spatial discretization method for ordinary differential equation in Section 11.1, for that part of the problem and a finite difference method for the temporal part. This technique of decomposing the problem into two parts is an effective technique to generate a numerical scheme, and it allows us to **reuse existing software** already developed for the o.d.e. problem. Many time dependent problems can be treated in the same manner. This technique goes by many names:

- (time) **splitting** since the time and space parts are separated and treated by independent methods
- the **method of lines** since the problem is solved on a sequence of lines (copies of the spatial domain), one for each time step.

10.5.1 Explicit Euler time discretization

The simplest time discretization method for the heat equation uses the forward (or explicit) Euler difference method. It takes the form

$$\begin{aligned} u^{n+1}(x) = u^n(x) + \Delta t \frac{\partial^2 u^n}{\partial x^2}(x) \quad \forall x \in [0,1], \\ u^0(x) = u_0(x) \quad \forall x \in [0,1] \\ u^n(0) = g_0(n\Delta t) \quad \text{and} \quad u^n(1) = g_1(n\Delta t) \quad \forall n > 0, \end{aligned} \tag{10.31}$$

where $u^n(x)$ denotes an approximation to $u(x, n\Delta t)$. Applying the finite element approximation (8.5) (or the finite difference approximation (11.2)) to (10.31) yields a simple algorithm. It just involves multiplying a matrix times a vector in the finite difference case at each time step, but in the finite element case a system involving the mass matrix (see Section 8.3) must be solved. The difficulty with this simple algorithm is that it is *unstable* unless Δt is sufficiently small. This will be explained in more detail in Section 11.5.

10.5.2 Implicit Euler time discretization

The simplest implicit time discretization method for the heat equation uses the backward (or implicit) Euler difference method. It takes the form

$$\begin{aligned} u^{n+1}(x) = u^n(x) + \Delta t \frac{\partial^2 u^{n+1}}{\partial x^2}(x) \quad \forall x \in [0,1], \\ u^0(x) = u_0(x) \quad \forall x \in [0,1] \\ u^n(0) = g_0(n\Delta t) \quad \text{and} \quad u^n(1) = g_1(n\Delta t) \quad \forall n > 0, \end{aligned} \tag{10.32}$$

where $u^n(x)$ again denotes an approximation to $u(x, n\Delta t)$. Applying the finite element approximation (8.5) (or the finite difference approximation (11.2)) to (10.32) yields now a system of equations involving the $a(\cdot, \cdot)$ form to be solved at each time step. This algorithm is *stable* for all Δt, but now we have to solve a system of equations (instead of just multiplying by a matrix in the finite difference case). Note however that the system to be solved is just the same as in the ODE boundary-value problems studied earlier, so the same family of techniques can be used.