

10.2: Forward Euler Method

The **Forward Euler Method** is the conceptually simplest method for solving the initial-value problem. For simplicity, let us discretize time, with equal spacings:

$$t_0, t_1, t_2, \dots \text{ where } h \equiv t_{n+1} - t_n. \quad (10.2.1)$$

Let us denote $\vec{y}_n \equiv \vec{y}(t_n)$. The Forward Euler Method consists of the approximation

$$\vec{y}_{n+1} = \vec{y}_n + h\vec{F}(\vec{y}_n, t_n). \quad (10.2.2)$$

Starting from the initial state \vec{y}_0 and initial time t_0 , we apply this formula repeatedly to compute \vec{y}_1, \vec{y}_2 , and so forth. The Forward Euler Method is called an **explicit method**, because, at each step n , all the information that you need to calculate the state at the next time step, \vec{y}_{n+1} , is already explicitly known—i.e., you just need to plug \vec{y}_n and t_n into the right-hand side of the above formula.

The Forward Euler Method formula follows from the usual definition of the derivative, and becomes exactly correct as $h \rightarrow 0$. We can deduce the numerical error, which is called the **local truncation error** in this context, by Taylor expanding the left-hand side around $t = t_n$:

$$\vec{y}_{n+1} = \vec{y}_n + h \left. \frac{d\vec{y}}{dt} \right|_{t_n} + \frac{h^2}{2} \left. \frac{d^2\vec{y}}{dt^2} \right|_{t_n} + \dots \quad (10.2.3)$$

The first two terms are precisely equal to the right-hand side of the Forward Euler Method formula. The local truncation error is the magnitude of the remaining terms, and hence it scales as $O(h^2)$.

10.2.1 Instability

For the Forward Euler Method, the local truncation error leads to a profound problem known as **instability**. Because the method involves repeatedly applying a formula with a local truncation error at each step, it is possible for the errors on successive steps to progressively accumulate, until the solution itself blows up. To see this, consider the differential equation

$$\frac{dy}{dt} = -\kappa y. \quad (10.2.4)$$

Given an initial state y_0 at time $t_0 = 0$, the solution is $y(t) = y_0 e^{-\kappa t}$. For $\kappa > 0$, this decays exponentially to zero with increasing time. However, consider the solutions produced by the Forward Euler Method:

$$y_1 = y_0 + h \cdot (-\kappa y_0) = (1 - h\kappa) y_0 \quad (10.2.5)$$

$$y_2 = y_1 + h \cdot (-\kappa y_1) = (1 - h\kappa)^2 y_0 \quad (10.2.6)$$

$$\vdots = \vdots \quad (10.2.7)$$

$$y_n = (1 - h\kappa)^n y_0. \quad (10.2.8)$$

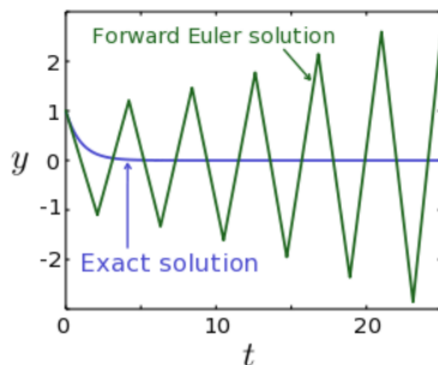


Figure 10.2.1: The exact solution (blue) and Forward Euler solution (green) for $dy/dt = -\kappa y(t)$, for $y(0) = 1$, $\kappa = 1$, and $h = 2.1$. The numerical solution is unstable; it blows up at large times, even though the exact solution is decaying to zero.

If $h > 2/\kappa$, then $1 - h\kappa < -1$, and as a result $y_n \rightarrow \pm\infty$ as $n \rightarrow \infty$. Even though the actual solution decays exponentially to zero, the numerical solution blows up, as shown in Fig. 10.2.1. Roughly speaking, the local truncation error causes the numerical

solution to "overshoot" the true solution; if the step size is too large, the magnitude of the overshoot keeps growing with each step, destabilizing the numerical solution.

Stability is a fundamental problem for the integration of ODEs. The equations which tend to destabilize numerical ODE solvers are those containing spring constants which are "large" compared to the time step; such equations are called **stiff equations**. At first, you might think that it's no big deal: just make the step size h sufficiently small, and the blow-up can be avoided. The trouble is that it's often unclear how small is sufficiently small, particularly for complicated (e.g. nonlinear) ODEs, where $F(\vec{y}, t)$ is something like a "black box". Unlike the above simple example, we typically don't have the exact answer to compare with, so it can be hard to tell whether the numerical solution blows up because that's how the true solution behaves, or because the numerical method is unstable.

This page titled [10.2: Forward Euler Method](#) is shared under a [CC BY-SA 4.0](#) license and was authored, remixed, and/or curated by [Y. D. Chong](#) via [source content](#) that was edited to the style and standards of the LibreTexts platform.