

0.1 Numerical Stability

The simplest differential equation to solve is an ordinary differential equation

$$\frac{du}{dt} = f(u, t) \quad (1)$$

with initial value

$$u(0) = u_0 \quad (2)$$

where we are interested in solving on some interval $[0, T]$.

The definition of the derivative as a limit of difference quotients suggests a method of discretization:

$$\frac{du}{dt}(t) \approx \frac{u(t + \Delta t) - u(t)}{\Delta t} \quad (3)$$

where Δt is a small positive parameter. This suggests an algorithm for generating a sequence of values $u_n \approx u(n\Delta t)$ given by (for example)

$$u_n = u_{n-1} + \Delta t f(u_n, t_n) \quad (4)$$

where $t_n = n\Delta t$.

The algorithm (4) is called the **implicit Euler** method, and it can be shown that it generates a sequence with the property that

$$|u(t_n) - u_n| \leq C_{f,T} \Delta t \quad \forall t_n \leq T \quad (5)$$

provided that we solve the implicit equation(4) for u_n exactly and we compute with exact arithmetic. The issue of solving the nonlinear equation at each step is important but not a show-stopper. However, the requirement of using finite-precision arithmetic means that the best error behavior we could expect is

$$|u(t_n) - u_n| \leq C_{f,T} \Delta t + n\epsilon \quad \forall t_n \leq T \quad (6)$$

where ϵ measures the precision error that occurs at each step in (4). It is useful to re-write (6) using the fact that $n = t_n/\Delta t$ as

$$|u(t_n) - u_n| \leq C_{f,T} \Delta t + \frac{t_n \epsilon}{\Delta t} \quad (7)$$

which shows that the error reaches a minimum and cannot be reduced by reducing Δt .

One way to increase the accuracy in (6) is to use a more accurate approximation of the derivative than (3), such as given by the **backwards differentiaion formulæ (BDF)**

$$\frac{du}{dt}(t) \approx \frac{1}{\Delta t} \sum_{i=0}^k a_i u_{n-i} \quad (8)$$

where the coefficients $\{a_i : i = 0, \dots, k\}$ are chosen so that (8) is exact for polynomials of degree k . The BDF for $k = 1$ is the same as implicit Euler. Using the approximation (8), we get an algorithm of the form

$$\sum_{i=0}^k a_i u_{n-i} = \Delta t f(u_n, t_n) \quad (9)$$

which can be solved for u_n provided $a_0 \neq 0$. In this case, the final error estimate would be

$$|u(t_n) - u_n| \leq C_{f,T,k} \Delta t^k + \frac{t_n \epsilon}{\Delta t}. \quad (10)$$

Ultimate accuracy is still limited, but smaller absolute errors (with larger Δt) can be achieved with higher values of k . For example, suppose that

- $\epsilon = 10^{-6}$ (which corresponds to single precision on a 32-bit machine)
- $T = 1$ and
- (for the sake of argument) $C_{f,T,k} = 1$.

Then with implicit Euler ($k = 1$) the smallest error we can get is 10^{-3} with $\Delta t = 10^{-3}$. On the other hand, with $k = 2$ we get an error of size 10^{-4} with $\Delta t = 10^{-2}$. Not only is this a smaller error but less work needs to be done to achieve it. In practice, the constant $C_{f,T,k}$ would depend on k and the exact error behavior would likely be different in detail, but the general conclusion that a higher-order scheme may be better still holds. The BDF methods for $k = 2$ and 3 are extremely popular schemes.

We see that higher-order schemes can lead to more manageable errors and potentially less work for the same level of accuracy. Thus it seems natural to ask whether there are limits to choosing the order to be arbitrarily high. Unfortunately, not all of the BDF schemes are viable. Beyond degree six, they become **unstable**. Let us examine the question of stability via a simple experiment. Suppose that, after some time T_0 , it happens that $f(u, t) = 0$ for $t \geq T_0$. Then the solution u remains constant after T_0 , since $\frac{du}{dt} \equiv 0$. What happens in the algorithm (9) is that we have

$$\sum_{i=0}^k a_i u_{n-i} = 0 \quad (11)$$

for $n \geq T_0/\Delta t$. However, this does not necessarily imply that u_n would tend to a constant. Let us examine what the solutions of (11) could look like.

Consider the sequence $u_n := \xi^{-n}$ for some number ξ . Plugging into (11) we find

$$0 = \sum_{i=0}^k a_i \xi^{-n+i} = \xi^{-n} \sum_{i=0}^k a_i \xi^i \quad (12)$$

If we define the polynomial p_k by

$$p_k(\xi) = \sum_{i=0}^k a_i \xi^i \quad (13)$$

we see that we have a null solution to (11) if and only if ξ is a root of p_k . If there is a root ξ of p_k where $|\xi| < 1$ then we get solutions to (11) which grow like

$$u_n = \xi^{-n} = \left(\frac{1}{\xi}\right)^{t_n/\Delta t}. \quad (14)$$

Not only does this blow up exponentially, the exponential rate goes to infinity as $\Delta t \rightarrow 0$. This clearly spells disaster. On the other hand, if $|\xi| > 1$, then the solution (14) goes rapidly to zero, and more rapidly as $\Delta t \rightarrow 0$. For roots ξ with $|\xi| = 1$ the situation is more complicated, and $\xi = 1$ is always a root because the sum of the coefficients a_i is always zero. Instability occurs if there is a multiple root on the unit circle $|\xi| = 1$. In general, one must consider all complex (as well as real) roots ξ .

Now that we know what to look for, let us return to the question of the higher-order BDF methods. It is well known that

$$\sum_{i=0}^k a_i u_{n-i} = \sum_{j=1}^k \frac{1}{j} \nabla^j u_n \quad (15)$$

where we define ∇u_n to be the sequence whose n -th entry is $u_n - u_{n-1}$. The higher powers are defined by induction: $\nabla^{j+1} u_n = \nabla(\nabla^j u_n)$. (There is multiple notation abuse here: ∇u_n really means the n -th element of the sequence $\nabla\{u.\}$ where $\{u.\}$ denotes the full sequence.) For example, $\nabla^2 u_n = u_n - 2u_{n-1} + u_{n-2}$, and in general ∇^j has coefficients given from Pascal's triangle. We thus see that $a_0 \neq 0$ for all $k \geq 1$.

Given this simple definition of the general case of BDF, it is hard to imagine what could go wrong regarding stability. Unfortunately, the condition that $|\xi| \geq 1$ for roots of $p_k(\xi) = 0$ restricts k to be six or less for the BDF formulæ.

Exercise 0.1 *Consider the example following (10). How small can the error in (10) be made using a 5-th order ($k = 5$) BDF formula? What value of Δt yields the smallest error?*

Exercise 0.2 *Compute the solution of an ordinary differential equation using the 7-th order ($k = 7$) BDF formula. What happens after a few times steps?*