Backward Euler method

From Wikipedia, the free encyclopedia

In numerical analysis and scientific computing, the **backward Euler method** (or **implicit Euler method**) is one of the most basic numerical methods for the solution of ordinary differential equations. It is similar to the (standard) Euler method, but differs in that it is an implicit method. The backward Euler method has order one and is A-stable.

Contents [hide]

Description [edit]

Consider the ordinary differential equation

$$\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)$$

with initial value $y(t_0) = y_0$. Here the function f and the initial data t_0 and y_0 are known; the function y depends on the real variable t and is unknown. A numerical method produces a sequence $y_0, y_1, y_2, \ldots$ such that y_k approximates $y(t_0 + kh)$, where h is called the step size.

The backward Euler method computes the approximations using

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1}).$$ [1]

This differs from the (forward) Euler method in that the latter uses $f(t_k, y_k)$ in place of $f(t_{k+1}, y_{k+1})$.

The backward Euler method is an implicit method: the new approximation y_{k+1} appears on both sides of the equation, and thus the method needs to solve an algebraic equation for the unknown y_{k+1}. Sometimes, this can be done by fixed-point iteration:

$$y_{k+1}^{[0]} = y_k, \quad y_{k+1}^{[i+1]} = y_k + hf(t_{k+1}, y_{k+1}^{[i]}).$$

If this sequence converges (within a given tolerance), then the method takes its limit as the new approximation y_{k+1}. [2]

Alternatively, one can use (some modification of) the Newton–Raphson method to solve the algebraic equation.

Derivation [edit]

Integrating the differential equation $\frac{\mathrm{d}y}{\mathrm{d}t} = f(t, y)$ from t_k to $t_{k+1} = t_k + h$ yields

$$y(t_{k+1}) - y(t_k) = \int_{t_k}^{t_{k+1}} f(t, y(t))\,\mathrm{d}t.$$

Now approximate the integral on the right by the right-hand rectangle method (with one rectangle):

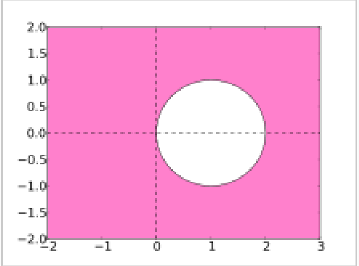
$$y(t_{k+1}) - y(t_k) \approx hf(t_{k+1}, y(t_{k+1})).$$

Finally, use that y_k is supposed to approximate $y(t_k)$ and the formula for the backward Euler method follows.[3]

The same reasoning leads to the (standard) Euler method if the left-hand rectangle rule is used instead of the right-hand one.

Analysis [edit]

The backward Euler method has order one. This means that the local truncation error (defined as the error made in one step) is $O(h^2)$, using the big O notation. The error at a specific time t is $O(h)$.



The pink region outside the disk shows the stability region of the backward Euler method.

The region of absolute stability for the backward Euler method is the complement in the complex plane of the disk with radius 1 centered at 1, depicted in the figure.[4] This includes the whole left half of the complex plane, so the backward Euler method is A-stable, making it suitable for the solution of stiff equations.[5] In fact, the backward Euler method is even L-stable.

Extensions and modifications [edit]

The backward Euler method is a variant of the (forward) Euler method. Other variants are the semi-implicit Euler method and the exponential Euler method.

The backward Euler method can be seen as a Runge–Kutta method with one stage, described by the Butcher tableau:

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array}$$

The backward Euler method can also be seen as a linear multistep method with one step. It is the first method of the family of Adams–Moulton methods, and also of the family of backward differentiation formulas.

Notes [edit]

1. ^ Butcher 2003, p. 57
2. ^ Butcher 2003, p. 57
3. ^ Butcher 2003, p. 57
4. ^ Butcher 2003, p. 70
5. ^ Butcher 2003, p. 71

References [edit]

- Butcher, John C. (2003), *Numerical Methods for Ordinary Differential Equations*, New York: John Wiley & Sons, ISBN 978-0-471-96758-3.

v · t · e	**Numerical integration methods by order**	[hide]
First-order	**Second-order**	**Higher-order**
Euler (**backward** · semi-implicit · exponential)	Verlet (velocity) · Trapezoidal · Beeman · Midpoint · Heun · Newmark-beta · Leapfrog	Exponential integrators · General linear (Runge–Kutta (list) · multistep)

Categories: Numerical differential equations | Runge–Kutta methods