Numerical Methods for Solving DE's

MATH 2250 Lecture 09
Book sections 2.4-2.6

September 10-11, 2019

Differential Equations

The DE

$$y'(x) = f(x, y), \qquad\qquad y(x_0) = y_0,$$

cannot be solved analytically (with pencil and paper yielding an explicit formula for $y(x)$) for general f.

However, we can *approximate* the solution using an algorithm.

Differential Equations

The DE

$$y'(x) = f(x, y), \qquad\qquad y(x_0) = y_0,$$

cannot be solved analytically (with pencil and paper yielding an explicit formula for $y(x)$) for general f.

However, we can *approximate* the solution using an algorithm.

We will present three algorithmic ways to approximate the solution $y(x)$ at a discrete set of x values:

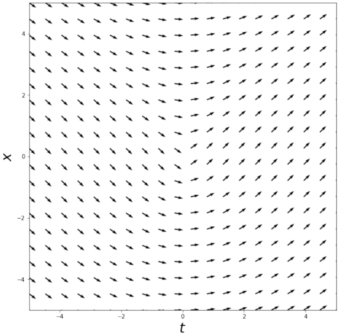
- the Euler method
- the Improved Euler method
- a Runge-Kutta method

The Euler method

Recall that for the DE

$$y'(x) = f(x, y),$$

we can plot a slope fields that visually approximate DE solutions:

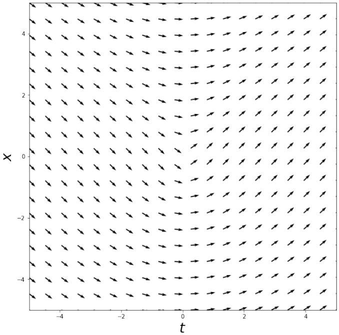


The Euler method

Recall that for the DE

$$y'(x) = f(x, y),$$

we can plot a slope fields that visually approximate DE solutions:



The idea behind the Euler method is straightforward from this picture: we will use lines whose slopes are defined by the slope field to trace out a(n approximate) solution.

The Euler method algorithm

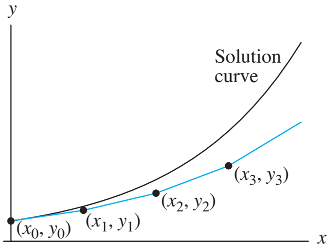
The Euler method algorithm is as follows: let $h > 0$ be a fixed *stepsize*. We will travel h units in the x direction along a line in the slopefield:

$$y(x_0) = y_0 \quad \Longrightarrow \quad y(x_0 + h) \approx y_0 + hf(x_0, y_0).$$

Let us give our approximation to $y(x_0 + h)$ some notation:

$$\begin{aligned} x_1 &:= x_0 + h \\ y_1 &:= y_0 + hf(x_0, y_0) \approx y(x_1) \end{aligned}$$

Note that $y_1 \neq y(x_1)$! One is an approximation, the other is the *exact* value.



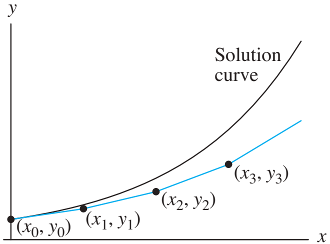
The Euler method algorithm

The Euler method iterates this procedure: with

$$x_n := x_0 + nh,$$

then y_{n+1} is computed by assuming that $y(x_n) = y_n$ and using the slope field there:

$$y(x_{n+1}) \approx y_{n+1} := y_n + hf(x_n, y_n).$$



Note that this is an *algorithm* and can (should) be programmed.

Euler's method

Example (Example 1, section 2.4)

Apply Euler's method to approximate the solution of the initial value problem

$$y'(x) = x + \frac{1}{5}y, \qquad\qquad y(0) = -3.$$

First use $h = 1$ for $x \in [0, 5]$, and then use $h = 0.2$ on $x \in [0, 1]$.

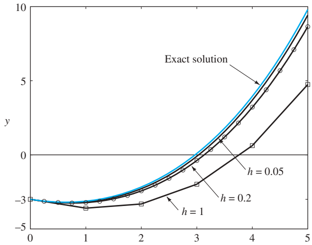
Euler's method

Example (Example 1, section 2.4)

Apply Euler's method to approximate the solution of the initial value problem

$$y'(x) = x + \frac{1}{5}y, \qquad\qquad y(0) = -3.$$

First use $h = 1$ for $x \in [0, 5]$, and then use $h = 0.2$ on $x \in [0, 1]$.



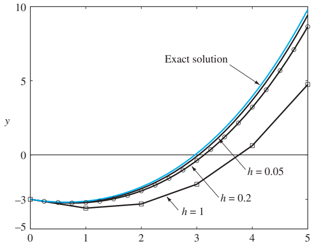
Euler's method

Example (Example 1, section 2.4)

Apply Euler's method to approximate the solution of the initial value problem

$$y'(x) = x + \frac{1}{5}y, \qquad\qquad y(0) = -3.$$

First use $h = 1$ for $x \in [0, 5]$, and then use $h = 0.2$ on $x \in [0, 1]$.

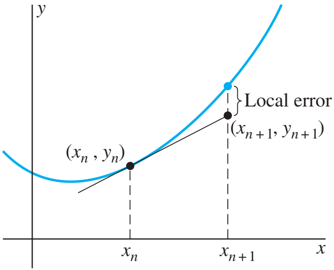


Demo: `euler_demo.ipynb`

Errors committed using Euler's method

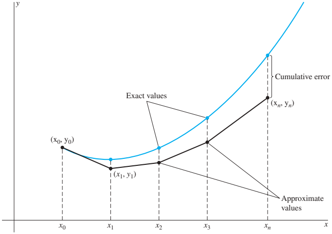
We have seen that using the stepsize h with Euler's method results in some errors.

A relatively benign error is that committed at each step, the *local* error:



Errors committed using Euler's method

Far more troublesome is the fact that we approximate future values using current values, which are already approximate. This is a *global* or *cumulative* error.



Convergence of the Euler algorithm

One of the main reasons why Euler's algorithm is useful is that the computed solutions *converge* to the real solution as $h \downarrow 0$.

Theorem

Consider the differential equation

$$y'(x) = f(x, y), \qquad\qquad y(a) = y_0,$$

Suppose this IVP has a unique solution $y(x)$ for x on the interval $[a, b]$. Further assume that $y(x)$ has continuous second derivative on this interval. Then for all $h > 0$, there is a constant C such that

$$|y_n - y(x_n)| \leqslant Ch,$$

for all $n \geqslant 0$ such that $x_n := a + nh \leqslant b$, and y_n is the Euler algorithm approximation to $y(x_n)$ computed using the stepsize h.

"Better" algorithms

The Euler algorithm, despite its apparent use, is rarely used in practice.

It is less "stable", less "efficient", and less "accurate" than alternative methods.

Many algorithms improve on Euler by observing that the slope of the line, $f(x_n, y_n)$, can be changed for improvement.

"Better" algorithms

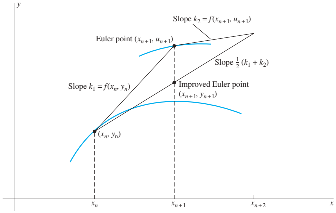
The Euler algorithm, despite its apparent use, is rarely used in practice.

It is less "stable", less "efficient", and less "accurate" than alternative methods.

Many algorithms improve on Euler by observing that the slope of the line, $f(x_n, y_n)$, can be changed for improvement.

A simple improvement is the so-called **improved Euler** algorithm.

$$u_{n+1} = y_n + hf(x_n, y_n),$$
$$y_{n+1} = y_n + h\left[\frac{1}{2}f(x_n, y_n) + \frac{1}{2}f(x_{n+1}, u_{n+1})\right)$$



Convergence of improved Euler

Theorem

Consider the differential equation

$$y'(x) = f(x, y), \qquad\qquad y(a) = y_0,$$

Suppose this IVP has a unique solution $y(x)$ *for* x *on the interval* $[a, b]$*. Further assume that* $y(x)$ *has continuous third derivative on this interval. Then for all* $h > 0$*, there is a constant* C *such that*

$$|y_n - y(x_n)| \leqslant Ch^2,$$

for all $n \geqslant 0$ *such that* $x_n := a + nh \leqslant b$*, and* y_n *is the improved Euler algorithm approximation to* $y(x_n)$ *computed using the stepsize* h*.*

Convergence of improved Euler

Theorem

Consider the differential equation

$$y'(x) = f(x, y), \qquad\qquad y(a) = y_0,$$

Suppose this IVP has a unique solution $y(x)$ for x on the interval $[a, b]$. Further assume that $y(x)$ has continuous third derivative on this interval. Then for all $h > 0$, there is a constant C such that

$$|y_n - y(x_n)| \leqslant Ch^2,$$

for all $n \geqslant 0$ such that $x_n := a + nh \leqslant b$, and y_n is the improved Euler algorithm approximation to $y(x_n)$ computed using the stepsize h.

Note that if h is small, say $h = 0.01$, then

$$\text{(improved Euler error)} \qquad h^2 \ll h \qquad \text{(Euler error)}.$$

This is one main motivation for using improved Euler.

Demo: `improved_euler_demo.ipynb`

A standard, ubiquitous algorithm

Euler and improved Euler are actually special cases of a *family* of algorithms called **Runge-Kutta** methods.

A particular, popular Runge-Kutta method is "Runge-Kutta 4", and is based on *Simpson's Rule* for numerical approximation of integrals. Recall that

$$\int_0^h f(x)\mathrm{d}x \approx \frac{h}{6}\left[f(0) + 2f\left(\frac{h}{2}\right) + 2f\left(\frac{h}{2}\right) + f(h)\right].$$

This is Simpson's rule for approximating integrals.

This is relevant for DE's since

$$y' = f(x,y) \;\; + \;\; y(x_0) = y_0 \Longrightarrow y(x_0+h) = y(x_0) + \int_{x_0}^{x_0+h} f(x, y(x))\mathrm{d}x.$$

(This is the Fundamental Theorem of Calculus.)

Runge-Kutta 4

The Runge-Kutta 4 algorithm applies Simpson's rule for integration at every time step, using approximations to estimate intermediate slopes.

$$
\begin{aligned}
k_1 &= f(x_n, y_n), \\
k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right) \\
k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right) \\
k_4 &= f\left(x_n + h, y_n + h k_3\right), \\
y_{n+1} &= y_n + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right].
\end{aligned}
$$

Runge-Kutta 4

The Runge-Kutta 4 algorithm applies Simpson's rule for integration at every time step, using approximations to estimate intermediate slopes.

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \\ k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \\ k_4 &= f\left(x_n + h, y_n + hk_3\right), \\ y_{n+1} &= y_n + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]. \end{aligned}$$

Under appropriate assumptions (y is several times differentiable), then the error committed by the Runge-Kutta 4 algorithm is

$$|y_n - y(x_n)| \leqslant Ch^4,$$

Again, $h^4 \ll h^2 \ll h$ for small h, motivating the use of this algorithm.

Runge-Kutta 4

The Runge-Kutta 4 algorithm applies Simpson's rule for integration at every time step, using approximations to estimate intermediate slopes.

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1\right) \\ k_3 &= f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2\right) \\ k_4 &= f\left(x_n + h, y_n + h k_3\right), \\ y_{n+1} &= y_n + \frac{h}{6}\left[k_1 + 2k_2 + 2k_3 + k_4\right]. \end{aligned}$$

Under appropriate assumptions (y is several times differentiable), then the error committed by the Runge-Kutta 4 algorithm is

$$|y_n - y(x_n)| \leqslant C h^4,$$

Again, $h^4 \ll h^2 \ll h$ for small h, motivating the use of this algorithm.

Demo: `runge_kutta_demo.ipynb`, `numerical_converence.ipynb`