Simple ODE Solvers - Derivation

These notes provide derivations of some simple algorithms for generating, numerically, approximate solutions to the initial value problem

$$y'(t) = f\big(t, y(t)\big)$$
$$y(t_0) = y_0$$

Here $f(t, y)$ is a given function, t_0 is a given initial time and y_0 is a given initial value for y. The unknown in the problem is the function $y(t)$. We start with

Euler's Method

Our goal is to determine (approximately) the unknown function $y(t)$ for $t \ge t_0$. We are told explicitly the value of $y(t_0)$, namely y_0. Using the given differential equation, we can also determine exactly the instantaneous rate of change of y at time t_0.

$$y'(t_0) = f\big(t_0, y(t_0)\big) = f\big(t_0, y_0\big)$$

If the rate of change of $y(t)$ were to remain $f\big(t_0, y_0\big)$ for all time, then $y(t)$ would be exactly $y_0 + f\big(t_0, y_0\big)(t - t_0)$. The rate of change of $y(t)$ does not remain $f\big(t_0, y_0\big)$ for all time, but it is reasonable to expect that it remains close to $f\big(t_0, y_0\big)$ for t close to t_0. If this is the case, then the value of $y(t)$ will remain close to $y_0 + f\big(t_0, y_0\big)(t - t_0)$ for t close to t_0. So pick a small number h and define

$$\begin{aligned} t_1 &= t_0 + h \\ y_1 &= y_0 + f\big(t_0, y_0\big)(t_1 - t_0) = y_0 + f\big(t_0, y_0\big)h \end{aligned}$$

By the above argument

$$y(t_1) \approx y_1$$

Now we start over. We now know the approximate value of y at time t_1. If $y(t_1)$ were exactly y_1, then the instantaneous rate of change of y at time t_1 would be exactly $f(t_1, y_1)$. If this rate of change were to persist for all future time, $y(t)$ would be exactly $y_1 + f\big(t_1, y_1\big)(t - t_1)$. As $y(t_1)$ is only approximately y_1 and as the rate of change of $y(t)$ varies with t, the rate of change of $y(t)$ is only approximately $f(t_1, y_1)$ and only for t near t_1. So we approximate $y(t)$ by $y_1 + f\big(t_1, y_1\big)(t - t_1)$ for t bigger than, but close to, t_1. Defining

$$\begin{aligned} t_2 &= t_1 + h = t_0 + 2h \\ y_2 &= y_1 + f\big(t_1, y_1\big)(t_2 - t_1) = y_1 + f\big(t_1, y_1\big)h \end{aligned}$$

we have

$$y(t_2) \approx y_2$$

We just repeat this argument ad infinitum. Define, for $n = 0, 1, 2, 3, \cdots$

$$t_n = t_0 + nh$$

Suppose that, for some value of n, we have already computed an approximate value y_n for $y(t_n)$. Then the rate of change of $y(t)$ for t close to t_n is $f\big(t, y(t)\big) \approx f\big(t_n, y(t_n)\big) \approx f\big(t_n, y_n\big)$ and, again for t close to t_n, $y(t) \approx y_n + f\big(t_n, y_n\big)(t - t_n)$. Hence

$$\boxed{y(t_{n+1}) \approx y_{n+1} = y_n + f\big(t_n, y_n\big)h} \tag{Eul}$$

This algorithm is called **Euler's Method**. The parameter h is called the **step size**.

Here is a table applying a few steps of Euler's method to the initial value problem

$$\begin{aligned} y' &= -2t + y \\ y(0) &= 3 \end{aligned}$$

with step size $h = 0.1$. For this initial value problem

$$\begin{aligned} f(t, y) &= -2t + y \\ t_0 &= 0 \\ y_0 &= 3 \end{aligned}$$

Of course this initial value problem has been chosen for illustrative purposes only. The exact solution is, easily, $y(t) = 2 + 2t + e^t$.

n	t_n	y_n	$f(t_n, y_n) = -2t_n + y_n$	$y_{n+1} = y_n + f(t_n, y_n) * h$
0	0.0	3.000	$-2 * 0.0 + 3.000 = 3.000$	$3.000 + 3.000 * 0.1 = 3.300$
1	0.1	3.300	$-2 * 0.1 + 3.300 = 3.100$	$3.300 + 3.100 * 0.1 = 3.610$
2	0.2	3.610	$-2 * 0.2 + 3.610 = 3.210$	$3.610 + 3.210 * 0.1 = 3.931$
3	0.3	3.931	$-2 * 0.3 + 3.931 = 3.331$	$3.931 + 3.331 * 0.1 = 4.264$
4	0.4	4.264	$-2 * 0.4 + 4.264 = 3.464$	$4.264 + 3.464 * 0.1 = 4.611$
5	0.5	4.611		

The Improved Euler's Method

Euler's method is one algorithm which generates approximate solutions to the initial value problem

$$\begin{aligned} y'(t) &= f\big(t, y(t)\big) \\ y(t_0) &= y_0 \end{aligned}$$

In applications, $f(t, y)$ is a given function and t_0 and y_0 are given numbers. The function $y(t)$ is unknown. Denote by $\varphi(t)$ the exact solution for this initial value problem. In other words $\varphi(t)$ is the function that obeys

$$\begin{aligned} \varphi'(t) &= f\big(t, \varphi(t)\big) \\ \varphi(t_0) &= y_0 \end{aligned}$$

exactly.

Fix a step size h and define $t_n = t_0 + nh$. We now derive another algorithm that generates approximate values for φ at the sequence of equally spaced time values $t_0,\ t_1,\ t_2,\ \cdots$. We shall denote the approximate values y_n with

$$y_n \approx \varphi(t_n)$$

By the fundamental theorem of calculus and the differential equation, the exact solution obeys

$$\begin{aligned} \varphi(t_{n+1}) &= \varphi(t_n) + \int_{t_n}^{t_{n+1}} \varphi'(t)\ dt \\ &= \varphi(t_n) + \int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt \end{aligned}$$

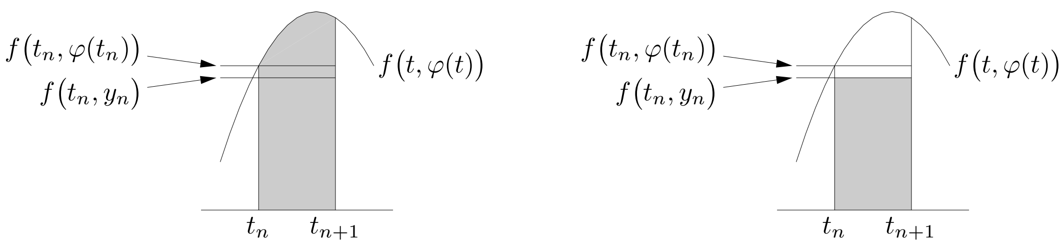
Fix any n and suppose that we have already found $y_0,\ y_1,\ \cdots,\ y_n$. Our algorithm for computing y_{n+1} will be of the form

$$y_{n+1} = y_n + \text{ approximate value for } \int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt$$

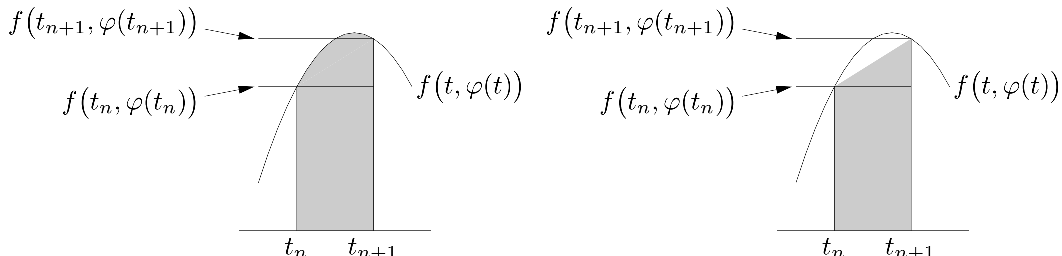
In fact Euler's method is of precisely this form. In Euler's method, we approximate $f\big(t, \varphi(t)\big)$ for $t_n \le t \le t_{n+1}$ by the constant $f\big(t_n, y_n\big)$. Thus

$$\text{Euler's approximate value for } \int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt = \int_{t_n}^{t_{n+1}} f\big(t_n, y_n\big)\ dt = f\big(t_n, y_n\big)h$$

The area of the complicated region $0 \le y \le f\big(t, \varphi(t)\big),\ t_n \le t \le t_{n+1}$ (represented by the shaded region under the parabola in the left half of the figure below) is approximated by the area of the rectangle $0 \le y \le f\big(t_n, y_n\big),\ t_n \le t \le t_{n+1}$ (the shaded rectangle in the right half of the figure below).



Our second algorithm, the improved Euler's method, gets a better approximation by attempting to approximate by the trapezoid on the right below rather than the rectangle on the right above. The exact area



of this trapezoid is the length h of the base multiplied by the average, $\frac{1}{2}[f\big(t_n, \varphi(t_n)\big) + f\big(t_{n+1}, \varphi(t_{n+1})\big)]$, of the heights of the two sides. Of course we do not know $\varphi(t_n)$ or $\varphi(t_{n+1})$ exactly. Recall that we have already found $y_0, \cdots, y_n$ and are in the process of finding y_{n+1}. So we already have an approximation for $\varphi(t_n)$, namely y_n, but not for $\varphi(t_{n+1})$. Improved Euler uses

$$\varphi(t_{n+1}) \approx \varphi(t_n) + \varphi'(t_n)h \approx y_n + f(t_n, y_n)h$$

in approximating $\frac{1}{2}[f\big(t_n, \varphi(t_n)\big) + f\big(t_{n+1}, \varphi(t_{n+1})\big)]$. Altogether

$$\begin{aligned}\text{Improved Euler's approximate value for } &\int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt \\ &= \tfrac{1}{2}\Big[f\big(t_n, y_n\big) + f\Big(t_{n+1}, y_n + f(t_n, y_n)h\Big)\Big]h\end{aligned}$$

so that the improved Euler's method algorithm is

$$\boxed{y(t_{n+1}) \approx y_{n+1} = y_n + \tfrac{1}{2}\Big[f\big(t_n, y_n\big) + f\Big(t_{n+1}, y_n + f(t_n, y_n)h\Big)\Big]h} \qquad \text{(ImpEul)}$$

Here are the first two steps of the improved Euler's method applied to

$$\begin{aligned} y' &= -2t + y \\ y(0) &= 3 \end{aligned}$$

with $h = 0.1$. In each step we compute $f(t_n, y_n)$, followed by $y_n + f(t_n, y_n)h$, which we denote $\tilde{y}_{n+1}$, followed by $f(t_{n+1}, \tilde{y}_{n+1})$, followed by $y_{n+1} = y_n + \frac{1}{2}\big[f\big(t_n, y_n\big) + f\big(t_{n+1}, \tilde{y}_{n+1}\big)\big]h$.

$$\begin{array}{lllrl}
t_0 = 0 & y_0 = 3 & \Longrightarrow & f(t_0, y_0) &= -2 * 0 + 3 = 3 \\
& & \Longrightarrow & \tilde{y}_1 &= 3 + 3 * 0.1 = 3.3 \\
& & \Longrightarrow & f(t_1, \tilde{y}_1) &= -2 * 0.1 + 3.3 = 3.1 \\
& & \Longrightarrow & y_1 &= 3 + \frac{1}{2}[3 + 3.1] * 0.1 = 3.305 \\
t_1 = 0.1 & y_1 = 3.305 & \Longrightarrow & f(t_1, y_1) &= -2 * 0.1 + 3.305 = 3.105 \\
& & \Longrightarrow & \tilde{y}_2 &= 3.305 + 3.105 * 0.1 = 3.6155 \\
& & \Longrightarrow & f(t_2, \tilde{y}_2) &= -2 * 0.2 + 3.6155 = 3.2155 \\
& & \Longrightarrow & y_2 &= 3.305 + \frac{1}{2}[3.105 + 3.2155] * 0.1 = 3.621025
\end{array}$$

Here is a table which gives the first five steps.

n	t_n	y_n	$f(t_n, y_n)$	$\tilde{y}_{n+1}$	$f(t_{n+1}, \tilde{y}_{n+1})$	y_{n+1}
0	0.0	3.000	3.000	3.300	3.100	3.305
1	0.1	3.305	3.105	3.616	3.216	3.621
2	0.2	3.621	3.221	3.943	3.343	3.949
3	0.3	3.949	3.349	4.284	3.484	4.291
4	0.4	4.291	3.491	4.640	3.640	4.647
5	0.5	4.647				

The Runge-Kutta Method

The Runge-Kutta algorithm is similar to the Euler and improved Euler methods in that it also uses, in the notation of the last section,

$$y_{n+1} = y_n + \text{ approximate value for } \int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt$$

But rather than approximating $\int_{t_n}^{t_{n+1}} f\big(t, \varphi(t)\big)\ dt$ by the area of a rectangle, as does Euler, or by the area of a trapezoid, as does improved Euler, it approximates by the area under a parabola. That is, it uses Simpson's rule. According to Simpson's rule (if you don't know Simpson's rule, just take my word for it)

$$\int_{t_n}^{t_n+h} f\big(t, \varphi(t)\big)\ dt \approx \tfrac{h}{6}\Big[f\big(t_n, \varphi(t_n)\big) + 4f\big(t_n + \tfrac{h}{2}, \varphi(t_n + \tfrac{h}{2})\big) + f\big(t_n + h, \varphi(t_n + h)\big)\Big]$$

As we don't know $\varphi(t_n)$, $\varphi(t_n + \frac{h}{2})$ or $\varphi(t_n + h)$, we have to approximate them as well. The Runge-Kutta algorithm, incorporating all these approximations, is

$$\boxed{\begin{aligned}
k_{n,1} &= f(t_n, y_n) \\
k_{n,2} &= f(t_n + \tfrac{1}{2}h, y_n + \tfrac{h}{2}k_{n,1}) \\
k_{n,3} &= f(t_n + \tfrac{1}{2}h, y_n + \tfrac{h}{2}k_{n,2}) \\
k_{n,4} &= f(t_n + h, y_n + hk_{n,3}) \\
y_{n+1} &= y_n + \tfrac{h}{6}\left[k_{n,1} + 2k_{n,2} + 2k_{n,3} + k_{n,4}\right]
\end{aligned}} \tag{RK}$$

Here are the first two steps of the Runge-Kutta algorithm applied to

$$\begin{aligned} y' &= -2t + y \\ y(0) &= 3 \end{aligned}$$

with $h = 0.1$.

$$\begin{aligned}
&t_0 = 0 \qquad y_0 = 3 \\
&\quad\Longrightarrow \quad k_{0,1} = f(0,3) = -2*0+3 = 3 \\
&\quad\Longrightarrow \quad\quad y_0 + \tfrac{h}{2}k_{0,1} = 3 + 0.05*3 = 3.15 \\
&\quad\Longrightarrow \quad k_{0,2} = f(0.05, 3.15) = -2*0.05+3.15 = 3.05 \\
&\quad\Longrightarrow \quad\quad y_0 + \tfrac{h}{2}k_{0,2} = 3 + 0.05*3.05 = 3.1525 \\
&\quad\Longrightarrow \quad k_{0,3} = f(0.05, 3.1525) = -2*0.05+3.1525 = 3.0525 \\
&\quad\Longrightarrow \quad\quad y_0 + hk_{0,3} = 3 + 0.1*3.0525 = 3.30525 \\
&\quad\Longrightarrow \quad k_{0,4} = f(0.1, 3.30525) = -2*0.1+3.30525 = 3.10525 \\
&\quad\Longrightarrow \quad\quad y_1 = 3 + \tfrac{0.1}{6}[3+2*3.05+2*3.0525+3.10525] = 3.3051708 \\
&t_1 = 0.1 \qquad y_1 = 3.3051708 \\
&\quad\Longrightarrow \quad k_{1,1} = f(0.1, 3.3051708) = -2*0.1+3.3051708 = 3.1051708 \\
&\quad\Longrightarrow \quad\quad y_1 + \tfrac{h}{2}k_{1,1} = 3.3051708 + 0.05*3.1051708 = 3.4604293 \\
&\quad\Longrightarrow \quad k_{1,2} = f(0.15, 3.4604293) = -2*0.15+3.4604293 = 3.1604293 \\
&\quad\Longrightarrow \quad\quad y_1 + \tfrac{h}{2}k_{1,2} = 3.3051708 + 0.05*3.1604293 = 3.4631923 \\
&\quad\Longrightarrow \quad k_{1,3} = f(0.15, 3.4631923) = -2*0.15+3.4631923 = 3.1631923 \\
&\quad\Longrightarrow \quad\quad y_1 + hk_{1,3} = 3.3051708 + 0.1*3.4631923 = 3.62149 \\
&\quad\Longrightarrow \quad k_{1,4} = f(0.2, 3.62149) = -2*0.2+3.62149 = 3.22149 \\
&\quad\Longrightarrow \quad\quad y_2 = 3.3051708 + \tfrac{0.1}{6}[3.1051708+2*3.1604293+ \\
&\qquad\qquad\qquad\qquad + 2*3.1631923+3.22149] = 3.6214025 \\
&t_2 = 0.2 \qquad y_2 = 3.6214025
\end{aligned}$$

and here is a table giving the first five steps. The intermediate data is only given to three decimal places even though the computation has been done to many more.

n	t_n	y_n	k_{n1}	y_{n1}	k_{n2}	y_{n2}	k_{n3}	y_{n3}	k_{n4}	y_{n+1}
0	0.0	3.000	3.000	3.150	3.050	3.153	3.053	3.305	3.105	3.305170833
1	0.1	3.305	3.105	3.460	3.160	3.463	3.163	3.621	3.221	3.621402571
2	0.2	3.621	3.221	3.782	3.282	3.786	3.286	3.950	3.350	3.949858497
3	0.3	3.950	3.350	4.117	3.417	4.121	3.421	4.292	3.492	4.291824240
4	0.4	4.292	3.492	4.466	3.566	4.470	3.570	4.649	3.649	4.648720639
5	0.5	4.648								

These notes have, hopefully, motivated the Euler, improved Euler and Runge-Kutta algorithms. So far we not attempted to see how efficient and how accurate the algorithms are. A first look at those questions is provided in the notes “Simple ODE Solvers – Error Behaviour”.