

## Simple ODE Solvers - Error Behaviour

These notes provide an introduction to the error behaviour of Euler's Method, the improved Euler's method and the Runge–Kutta algorithm for generating approximate solutions to the initial value problem

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

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  $y(t)$ .

Two obvious considerations in deciding whether or not a given algorithm is of any practical value are (a) the amount of computational effort required to execute the algorithm and (b) the accuracy that this computational effort yields. For algorithms like our simple ODE solvers, the bulk of the computational effort usually goes into evaluating the function  $f(t, y)$ . Euler's method uses one evaluation of  $f(t, y)$  for each step; the improved Euler's method uses two evaluations of  $f$  per step; the Runge–Kutta algorithm uses four evaluations of  $f$  per step. So Runge–Kutta costs four times as much work per step as does Euler. But this fact is extremely deceptive because, as we shall see, you typically get the same accuracy with a few steps of Runge–Kutta as you do with hundreds of steps of Euler.

To get a first impression of the error behaviour of these methods, we apply them to a problem that we know the answer to. The solution to the first order constant coefficient linear initial value problem

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

is

$$y(t) = 2 + 2t + e^t$$

In particular, the exact value of  $y(1)$ , to ten decimal places, is 6.7182818285. The following table lists the error in the approximate value for this number generated by our three methods applied with three different step sizes. It also lists the number of evaluations of  $f$  required to compute the approximation.

steps	Euler		Improved Euler		Runge–Kutta	
	error	#evals	error	#evals	error	#evals
5	$2.3 \times 10^{-1}$	5	$1.6 \times 10^{-2}$	10	$3.1 \times 10^{-5}$	20
50	$2.7 \times 10^{-2}$	50	$1.8 \times 10^{-4}$	100	$3.6 \times 10^{-9}$	200
500	$2.7 \times 10^{-3}$	500	$1.8 \times 10^{-6}$	1000	$3.6 \times 10^{-13}$	2000

Observe

- Using 20 evaluations of  $f$  worth of Runge–Kutta gives an error 90 times smaller than 500 evaluations of  $f$  worth of Euler.
- With Euler's method, decreasing the step size by a factor of ten appears to reduce the error by about a factor of ten.
- With improved Euler, decreasing the step size by a factor of ten appears to reduce the error by about a factor of one hundred.
- With Runge–Kutta, decreasing the step size by a factor of ten appears to reduce the error by about a factor of about  $10^4$ .

So it looks like

$$\begin{aligned}\text{approx value of } y(1) \text{ given by Euler with step size } h &\approx y(1) + K_E h \\ \text{approx value of } y(1) \text{ given by improved Euler with step size } h &\approx y(1) + K_{IE} h^2 \\ \text{approx value of } y(1) \text{ given by Runge–Kutta with step size } h &\approx y(1) + K_{RK} h^4\end{aligned}$$

with some constants  $K_E$ ,  $K_{IE}$  and  $K_{RK}$ . To test these conjectures further, I have applied our three methods

with about ten different step sizes of the form  $\frac{1}{2^m}$  with  $m$  integer. On the next page are three graphs, one for each method. Each contains a plot of  $\log_2 e_n$ , the (base 2) logarithm of the error for step size  $\frac{1}{n}$ , against the logarithm (of base 2) of  $n$ . The logarithm of base 2 is used because  $\log_2 2^m = m$  is nice and simple.

Notice that if, for some algorithm,

$$\text{approx value of } y(1) \text{ with step size } h = y(1) + Kh^k$$

then the error with step size  $\frac{1}{n}$  is  $e_n = K\frac{1}{n^k}$  and obeys  $\log_2 e_n = \log_2 K - k \log_2 n$ . The graph of  $\log_2 e_n$  against  $\log_2 n$  is a straight line of slope  $-k$ .

On each of the three graphs, I have also plotted a straight line, chosen by linear regression to best fit the data. (No, linear regression is not part of this course.) It sure looks like  $k = 1$  for Euler,  $k = 2$  for improved Euler and  $k = 4$  for Runge-Kutta.

This procedure can still be used even if we do not know the exact value of  $y(1)$ . Call  $A_h$  the approximate value with step size  $h$  and call  $A$  the (unknown) exact value. If

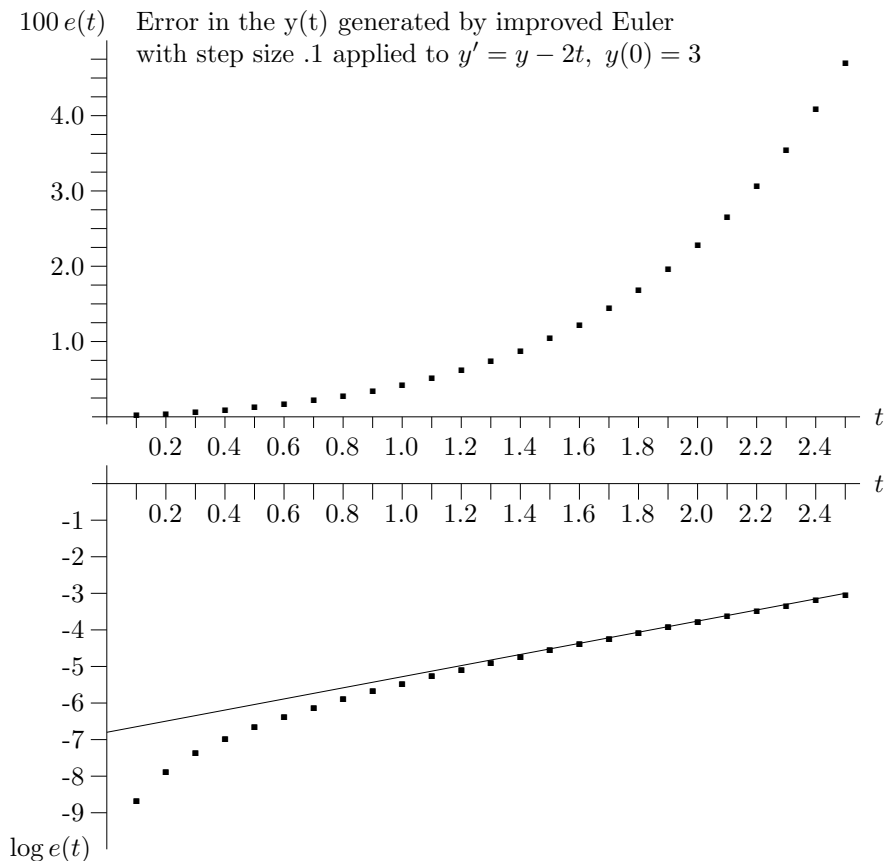
$$A_h = A + Kh^k \tag{1}$$

with  $K$  and  $k$  constant (but also unknown), then plotting

$$\log(A_h - A_{h/2}) = \log(Kh^k - K(\frac{h}{2})^k) = \log K(1 - \frac{1}{2^k}) + k \log h \tag{2}$$

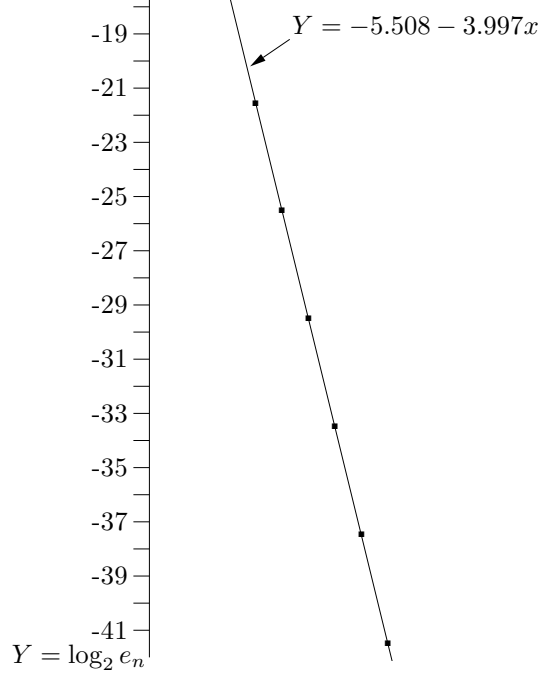
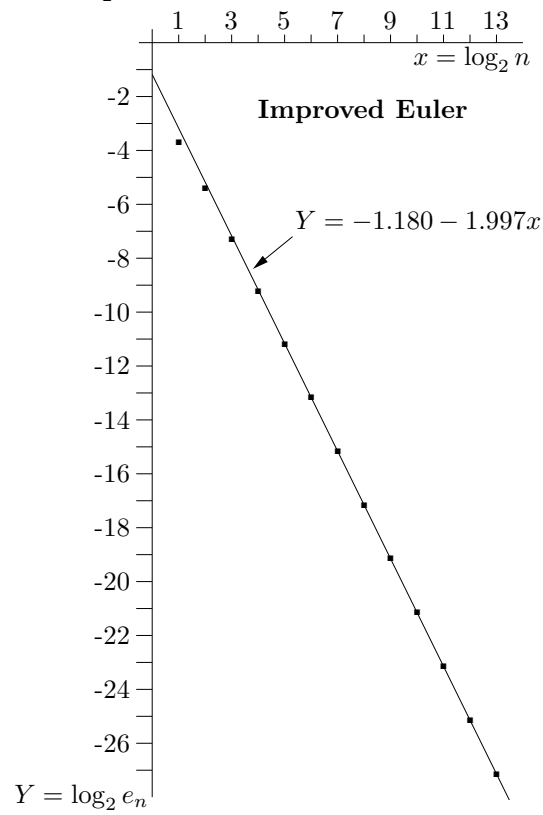
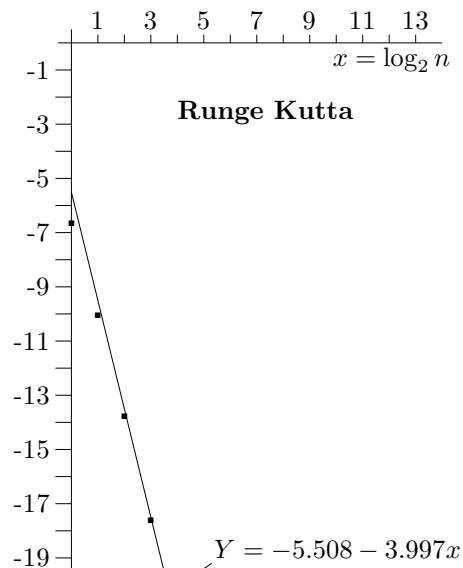
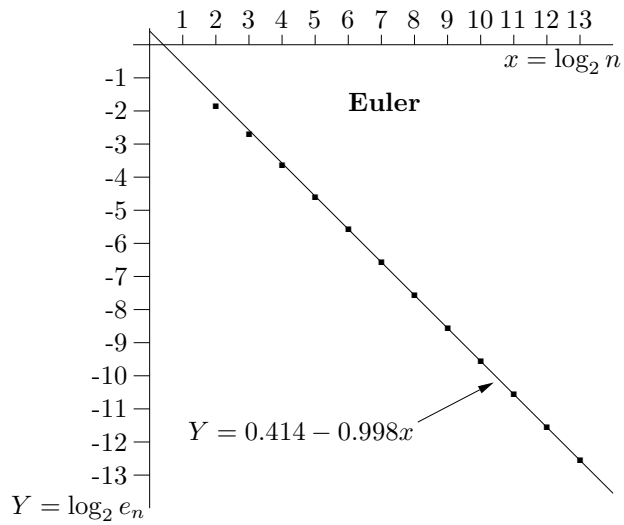
against  $\log h$  also gives a straight line of slope  $k$ .

So far we have only looked at the error in the approximate value of  $y(t_f)$  as a function of the step size  $h$  with  $t_f$  held fixed. The graphs below illustrate how the error behaves as a function of  $t_f$ , with  $h$  held fixed.



The error grows exponentially with  $t$ , at least for this example.

Error in the approximation, with step size  $\frac{1}{n}$ , to  $y(1)$  where  $y'(t) = y - 2t$ ,  $y(0) = 3$



## Local Truncation Error for Euler's Method

We now try develop some understanding as to why we got the above experimental results. We start with the error generated by a single step of Euler's method, under the assumptions that we start the step with the exact solution and that there is no roundoff error. This is called the **local truncation error** for Euler's method. So suppose that we are solving an initial value problem with differential equation

$$y'(t) = f(t, y)$$

Denote by  $\phi(t)$  the exact solution to the initial value problem. So, in particular,  $\phi(t)$  obeys

$$\phi'(t) = f(t, \phi(t))$$

for all  $t$ . Suppose further that we have already found  $y_n$  and that, by some miracle,  $y_n = \phi(t_n)$ . Now execute one more step of Euler's method with step size  $h$ :

$$y_{n+1} = y_n + hf(t_n, y_n)$$

Because we are assuming that  $y_n = \phi(t_n)$

$$y_{n+1} = \phi(t_n) + hf(t_n, \phi(t_n))$$

Because  $\phi(t)$  is the exact solution  $\phi'(t_n) = f(t_n, \phi(t_n))$  and

$$y_{n+1} = \phi(t_n) + h\phi'(t_n)$$

The error in  $y_{n+1}$  is, by definition  $\phi(t_{n+1}) - y_{n+1}$ . Taylor expanding  $\phi(t_{n+1}) = \phi(t_n + h)$  about  $t_n$

$$\phi(t_n + h) = \phi(t_n) + \phi'(t_n)h + \frac{1}{2}\phi''(t_n)h^2 + \frac{1}{3!}\phi'''(t_n)h^3 + \dots$$

so that

$$\begin{aligned} \text{error} &= \phi(t_{n+1}) - y_{n+1} \\ &= [\phi(t_n) + \phi'(t_n)h + \frac{1}{2}\phi''(t_n)h^2 + \frac{1}{3!}\phi'''(t_n)h^3 + \dots] - [\phi(t_n) + h\phi'(t_n)] \\ &= \frac{1}{2}\phi''(t_n)h^2 + \frac{1}{3!}\phi'''(t_n)h^3 + \dots \end{aligned} \tag{3}$$

So we conclude that the local truncation error for Euler's method is some unknown constant (we usually don't know the value of  $\frac{1}{2}\phi''(t_n)$  because we don't usually know the solution  $\phi(t)$  of the differential equation) times  $h^2$  plus smaller terms that are proportional to  $h^r$  with  $r \geq 3$ . This conclusion is typically written

$$\boxed{\text{Local truncation error for Euler's method} = Kh^2 + O(h^3)} \tag{4}$$

The symbol  $O(h^3)$  is used to designate any function that, for small  $h$ , is bounded by a constant times  $h^3$ .

To get from an initial time  $t = t_0$  to a final time  $t = t_f$  using steps of size  $h$  requires  $(t_f - t_0)/h$  steps. If each step were to introduce an error  $Kh^2 + O(h^3)$ , then the final error in the approximate value of  $y(t_f)$  would be  $K(t_f - t_0)h + O(h^2)$ . This is consistent with the experimental data for the dependence of error on step size with  $t_f$  held fixed, shown on the first graph of page three. It is not consistent with the experimental data on page four, which shows exponential, rather than linear, growth in  $t_f - t_0$ . We can get some rough understanding of this exponential growth as follows. The general solution to  $y' = y - 2t$  is  $y = 2 + 2t + c_0e^t$ . The arbitrary constant,  $c_0$ , is to be determined by initial conditions. When  $y(0) = 3$ ,  $c_0 = 1$ . At the end of step 1, we have computed an approximation  $y_1$  to  $y(h)$ . This  $y_1$  is not exactly  $2 + 2h + e^h$ . Instead, it is a number that differs from  $2 + 2h + e^h$  by  $O(h^2)$ . We chose to write this number as  $2 + 2h + (1 + \epsilon)e^h$  with  $\epsilon$  of order of magnitude  $h^2$ . If we were to make no further errors we would end up with the solution to

$$y' = y - 2t, \quad y(h) = 2 + 2h + (1 + \epsilon)e^h$$

which is

$$\begin{aligned} y(t) &= 2 + 2t + (1 + \epsilon)e^t = 2 + 2t + e^t + \epsilon e^t \\ &= \phi(t) + \epsilon e^t \end{aligned}$$

So, once an error has been introduced, the natural time evolution of the solutions to this differential equation cause the error to grow exponentially. Other differential equations with other time evolution characteristics will exhibit different  $t_f$  dependence of errors. In the next section, we show that, for many differential equations, errors grow at worst exponentially with  $t_f$ .

## Global Truncation Error for Euler's Method

Suppose once again that we are applying Euler's method with step size  $h$  to the initial value problem

$$\begin{aligned} y'(t) &= f(t, y) \\ y(0) &= y_0 \end{aligned}$$

Denote by  $\phi(t)$  the exact solution to the initial value problem and by  $y_n$  the approximation to  $\phi(t_n)$ ,  $t_n = t_0 + nh$ , given by  $n$  steps of Euler's method (applied without roundoff error). The error in  $y_n$  is  $\phi(t_n) - y_n$  and is called the **global truncation error** at time  $t_n$ . The word "truncation" is supposed to signify that this error is due solely to Euler's method and does not include any effects roundoff error. We now derive a bound on the global truncation error.

Define

$$\varepsilon_n = \phi(t_n) - y_n$$

The first half of the derivation is to find a bound on  $\varepsilon_{n+1}$  in terms of  $\varepsilon_n$ .

$$\begin{aligned} \varepsilon_{n+1} &= \phi(t_{n+1}) - y_{n+1} \\ &= \phi(t_{n+1}) - y_n - hf(t_n, y_n) \\ &= [\phi(t_n) - y_n] + h[f(t_n, \phi(t_n)) - f(t_n, y_n)] + [\phi(t_{n+1}) - \phi(t_n) - hf(t_n, \phi(t_n))] \end{aligned} \quad (5)$$

The first  $[\dots]$  is exactly  $\varepsilon_n$ . The third  $[\dots]$  is exactly the local truncation error. Assuming that  $|\phi''(t)| \leq A$  for all  $t$  of interest we have that

$$|\phi(t_{n+1}) - \phi(t_n) - hf(t_n, \phi(t_n))| \leq Ah^2$$

To see this, just use the Taylor expansion with remainder,  $\phi(t_{n+1}) = \phi(t_n) + \phi'(t_n)h + \frac{1}{2}\phi''(\tilde{t})h^2$  for some  $t_n < \tilde{t} < t_{n+1}$ , in place of  $\phi(t_{n+1}) = \phi(t_n) + \phi'(t_n)h + \frac{1}{2}\phi''(t_n)h^2 + \frac{1}{3!}\phi'''(t_n)h^3 + \dots$  in (3). Finally, by the mean value theorem,

$$\begin{aligned} |f(t_n, \phi(t_n)) - f(t_n, y_n)| &= \left| \frac{\partial f}{\partial y}(t_n, \tilde{y}) \right| |\phi(t_n) - y_n| \quad \text{for some } \tilde{y} \text{ between } y_n \text{ and } \phi(t_n) \\ &= \left| \frac{\partial f}{\partial y}(t_n, \tilde{y}) \right| |\varepsilon_n| \\ &\leq B|\varepsilon_n| \end{aligned}$$

assuming that  $\left| \frac{\partial f}{\partial y}(t, y) \right| \leq B$  for all  $t$  and  $y$  of interest. Substituting into (5) gives

$$|\varepsilon_{n+1}| \leq (1 + Bh)|\varepsilon_n| + Ah^2 \quad (6_n)$$

Hence the total error  $\varepsilon_{n+1}$  consists of two parts. One part is the local truncation error, which is no more than  $Ah^2$  and which is present even if we start the step with no error at all, i.e. with  $\varepsilon_n = 0$ . The other part is due to the combined error from all previous steps. At the beginning of step number  $n + 1$ , the combined error is  $|\varepsilon_n|$ . During the step, this error gets magnified by no more than a factor of  $1 + Bh$ .

The second half of the derivation is to repeatedly apply (6<sub>n</sub>) with  $n = 0, 1, 2, \dots$ . By definition  $\phi(t_0) = y_0$  so that  $\varepsilon_0 = 0$ , so

$$\begin{aligned} (6_0) &\implies |\varepsilon_1| \leq (1 + Bh)|\varepsilon_0| + Ah^2 = Ah^2 \\ (6_1) &\implies |\varepsilon_2| \leq (1 + Bh)|\varepsilon_1| + Ah^2 = (1 + Bh)Ah^2 + Ah^2 \\ (6_2) &\implies |\varepsilon_3| \leq (1 + Bh)|\varepsilon_2| + Ah^2 = (1 + Bh)^2 Ah^2 + (1 + Bh)Ah^2 + Ah^2 \end{aligned}$$

Continuing in this way

$$|\varepsilon_n| \leq (1 + Bh)^{n-1} Ah^2 + \cdots + (1 + Bh) Ah^2 + Ah^2 = \sum_{m=0}^{n-1} (1 + Bh)^m Ah^2$$

Applying  $\sum_{m=0}^{n-1} ar^m = \frac{r^n - 1}{r - 1} a$  with  $a = Ah^2$  and  $r = 1 + Bh$  gives

$$|\varepsilon_n| \leq \frac{(1 + Bh)^n - 1}{(1 + Bh) - 1} Ah^2 = \frac{A}{B} [(1 + Bh)^n - 1] h$$

Setting  $x = Bh$  in

$$x \geq 0 \implies 1 + x \leq 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots = e^x$$

gives  $(1 + Bh)^n \leq e^{Bhn} = e^{B(t_n - t_0)}$  since  $t_n = t_0 + nh$  and we arrive at the conclusion

$$\boxed{|\varepsilon_n| \leq \frac{A}{B} [e^{B(t_n - t_0)} - 1] h}$$

which is of the form  $K(t_f)h^k$  with  $k = 1$  and the coefficient  $K(t_f)$  growing exponentially with  $t_f - t_0$  as our experimental data suggested.