Math 2562

Brief Introduction to Numerics: Euler’s Method and Error

T. Forrest Kieffer

October 10, 2018

Most of the information in this lecture was borrowed from [1, Chapter 6] and some notes sent to me by Prof. Weiss. Some of the material in the section on Runge-Kutta methods was borrowed from Wikipedia.

1 Euler’s Method

As you are aware, it is not always possible to explicitly solve an ODE. In fact for generic ODE this is almost always the case. Therefore, it is important to have ways of approximating a solution to an ODE by simply analyzing the ODE itself. This is a vast subject, worthy of an entire course devoted to it (in fact, there are several graduate courses here at GT devoted to it). In these lectures, we take it upon ourselves to study perhaps the simplest numerical procedure for solving ODEs with an initial value: the Euler Method.

Throughout we consider the initial value problem

\[
\begin{align*}
\dot{y} &= f(t, y) \\
y(t_0) &= Y_0
\end{align*}
\]  

(1)

The function \( f(t, y) \) is to be continuous for all \((t, y)\) in some domain \(D\) of the \(ty\)-plane, and \((t_0, Y_0)\) is a point in \(D\). The results obtained for (1) will generalize in a straightforward way to both systems of differential equations and higher order equations, provided appropriate vector and matrix notation is used. We will use the notation \(Y(t)\) to denote the true solution to (1), i.e.

\[
\dot{Y}(t) = f(t, Y(t)), \quad Y(t_0) = Y_0,
\]

whereas \(y(t)\) will be used to denote the approximate solution.

Suppose we want to find the solution to (1) on the interval \([t_0, t]\), \(t > t_0\). Let \(h > 0\). We partition this interval into \(N = N(h)\) subintervals \([t_0, t_1], [t_1, t_2], \ldots, [t_N, t]\) where

\[t_j = t_0 + jh, \quad j = 0, 1, \ldots, N.\]

**Euler’s Method** is a recursive prescription for finding an approximate value \(y_n := y(t_n)\) to \(Y(t_n)\) and is defined by

\[
y_{n+1} = y_n + hf(t_n, y_n), \quad y_0 = Y_0.
\]

(2)

There are 4 ways of arriving at (2): Geometric reasoning, Taylor series, numerical differentiation, and numerical integration. We discuss the first two.

1) **Geometric Reasoning.** Consider the graph of \(Y(t)\) (not shown). Form the tangent line to the graph of \(Y(t)\) at \(t_0\), and use this line as an approximation to the curve \(Y(t)\) in \(t_0 \leq t \leq t_1\). Then

\[
\frac{y_1 - Y_0}{h} = Y'(t_0) = f(t_0, Y_0) \quad \Rightarrow \quad y_1 = Y_0 + hf(t_0, Y_0).
\]

Repeating this argument for the intervals \([t_1, t_2], [t_2, t_3], \ldots\) we arrive at (2).
2) **Taylor Series.** Expand \( Y(t_{n+1}) = Y(t_n + h) \) about \( t_n \):

\[
Y(t_{n+1}) = Y(t_n) + hY'(t_n) + \frac{h^2}{2} Y''(\xi_n), \quad t_n \leq \xi_n \leq t_{n+1}.
\]

Dropping the \( h^2 \)-term, we again arrive at (2). The term

\[
T_n = \frac{h^2}{2} Y''(\xi_n)
\]

is called the local truncation error at \( t_{n+1} \). We will discuss the local truncation error in more detail later.

**Example 1.** Consider the initial value problem

\[
\begin{align*}
\dot{y} + 2y^2 &= \frac{1}{1 + t^2} \\
y(0) &= 0
\end{align*}
\]

This initial value problem is a particular example of a Riccati equation. The exact solution to (3) is given by

\[
Y(t) = \frac{t}{1 + t^2}.
\]

To arrive at this general solution, consider the substitution \( y = \frac{1}{2} \frac{\dot{u}}{u} \). Then (3) becomes

\[
-2 \left( \frac{1}{2} \frac{\dot{u}}{u} \right)^2 + \frac{1}{1 + t^2} = \dot{y} = \frac{1}{2} \frac{\dot{u}}{u} - \frac{1}{2} \left( \frac{\dot{u}}{u} \right)^2
\]

and so

\[
\frac{1}{2} \ddot{u} - \frac{1}{1 + t^2} u = 0, \quad \dot{u}(0) = 0, \quad u(0) = 1.
\]

This equation is now equivalent to a one-dimensional Schrödinger equation with potential \( V(t) = 2/(1 + t^2) \). That aside, by looking at the IVP for \( u \) a moment, we immediately see that \( u(t) = 1 + t^2 \) satisfies the IVP. Computing \( \dot{u}/u \) we arrive at \( Y(t) \) above.

Let us use Euler’s method to solve (3) on the interval \([0, 2]\), using a step size of \( h = 2/N \) where \( N \geq 1 \). Take \( N = 10 \), and then \( h = 2/10 = 0.2 \). Using Euler’s method (2) we find (here \( t_0 = y_0 = 0 \))

\[
\begin{align*}
y_0 &= 0 \\
y_1 &= y_0 + \frac{2}{10} f(t_0, y_0) = \frac{2}{10} f(0, 0) = \frac{2}{10} \\
y_2 &= \frac{2}{10} + \frac{2}{10} f \left( \frac{2}{10}, \frac{2}{10} \right) = \frac{2}{10} + \frac{2}{10} \frac{573}{10} = \frac{1223}{3250} \approx 0.376
\end{align*}
\]

Table 1 shows the values \( \{y_1, \cdots, y_{10}\} \), \( \{Y(t_1), \cdots, Y(t_{10})\} \), and the global truncation error \( E_j = Y(t_j) - y_j \) at each step. From the table it is easy to pick off the maximum error and the corresponding time step for which it occurs:

\[
\max_j \{|E_j|\} = 0.0545, \quad \text{occurring at } \ t_4 = 0.8.
\]

The Appendix 5 contains some Matlab code to apply Euler’s method to this problem, and plot the approximate solution versus the exact solution. For \( N = 10 \), the output of the code is displayed in the Figure 1. We also display a curve of the absolute value of the global truncation error, \( E_j = y_j - Y(t_j) \), in Figure 2. Note how the error in 2 changes with time. It is not the case that the error accumulates as the computation goes on. As this example shows, the error can vary quite dramatically at various stages of the computation.
Table 1: Table displaying the result of Euler’s method applied to (3) with \( N = 10 \). Here \( E_j = Y(t_j) - y_j \) is the global truncation error.

\[
\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
y_j & 0.2000 & 0.3763 & 0.4921 & 0.5423 & 0.5466 & 0.5271 & 0.4979 & 0.4663 & 0.4355 & 0.4068 \\
Y(t_j) & 0.1923 & 0.3448 & 0.4412 & 0.4878 & 0.5000 & 0.4918 & 0.4730 & 0.4494 & 0.4245 & 0.4000 \\
|E_j| & 0.0077 & 0.0315 & 0.0509 & 0.0545 & 0.0466 & 0.0353 & 0.0250 & 0.0169 & 0.0110 & 0.0068 \\
\hline
\end{array}
\]

Figure 1: Black curve: Exact Solution to (3). Red curve: Approximation solution generated via Euler’s method (2). Red circles indicate the points \((t_j, y_j)\) for \( j = 0, \cdots, 10 \).

Figure 2: Black dashed curve shows the error \(|E_j| = |y_j - Y(t_j)|\) in the interval \([0, 2]\).
2 Error in Numerical Approximation

Consider an initial value problem

\[
\begin{aligned}
\dot{y} &= f(t, y) \\
y(0) &= Y_0,
\end{aligned}
\]

(4)

Suppose \( f \) is continuous and has continuous first partial derivatives on some region

\[ R = [0, t_f] \times [Y_0 - r, Y_0 + s], \]

(5)

where \( r, s > 0 \). As before, we let \( Y(t) \) denote the exact solution to (4). Consider the Euler method scheme

\[ y_{n+1} = y_n + hf(t_n, y_n), \quad h = t_n - t_{n-1} = \frac{t_n}{n}. \]

(6)

Define the global truncation error

\[ E_n = Y(t_n) - y_n. \]

(7)

Our goal is to find a worst-case estimate for the global truncation error \( E_n \) for Euler’s method.

The first step is to compare \( E_{n+1} \) to \( E_n \). For this we need to compare \( Y(t_{n+1}) \) with \( Y(t_n) \), and so we need to remind ourselves of Taylor’s Theorem.

**Theorem 1** Let \( Y : \mathbb{R} \to \mathbb{R} \) be twice continuously differentiable on some interval \( I = [0, t_f] \) and let \( f : \mathbb{R}^2 \to \mathbb{R} \) have continuous first partial derivatives on the rectangular region \( R = I \times [Y_0 - r, Y_0 + s] \), with \( r, s > 0 \).

1) Given \( T, t \in I \), there exists some point \( \tau \in I \) such that

\[ Y(t) = Y(T) + Y'(T)(t - T) + \frac{Y''(\tau)}{2}(t - \tau)^2. \]

2) Given points \((T, z), (T, Y) \in R\), there exists some point \((T, \eta) \in R\) such that

\[ f(T, Y) = f(T, z) + f_y(T, \eta)(Y - z) \]

where \( f_y \) is the partial derivative of \( f \) with respect to its second argument.

Using Theorem 1 above with \( t = t_{n+1} \) and \( T = t_n \) we find

\[ Y(t_{n+1}) = Y(t_n) + hf(t_n, Y(t_n)) + \frac{h^2}{2} Y''(\tau). \]

(8)

Using Theorem 1 again, with \( T = t_n \) and \( z = y_n \), we are able to relate \( f(t_n, Y(t_n)) \) to the error \( E_n \) at time step \( t_n \):

\[ f(t_n, Y(t_n)) = f(t_n, y_n) + f_y(t_n, \eta)[Y(t_n) - y_n] = f(t_n, y_n) + f_y(t_n, \eta)E_n. \]

(9)

Substituting equation (9) into (8) we find

\[ Y(t_{n+1}) = Y(t_n) + hf(t_n, y_n) + hf_y(t_n, \eta)E_n + \frac{h^2}{2} Y''(\tau). \]

(10)

Finally, we subtract \( y_{n+1} \) from both sides of (10) and use the Euler scheme 6 to conclude

\[ E_{n+1} = [1 + hf_y(t_n, \eta)]E_n + \frac{h^2}{2} Y''(\tau). \]

(11)

With (11) we can make some important observations. The local truncation error is defined to be the error in step \( n + 1 \) when there is no error in step \( n \); hence, by (11), the local truncation error for Euler’s method going from step \( n \) to step \( n + 1 \) is \( T_n = \frac{h^2}{2} Y''(\tau) \). We don’t have direct access to the local truncation error since we have no information regarding \( \tau \). However, it is possible to obtain an estimate on the local truncation error using the
properties of \( f \) (see the definition of \( M \) in (12) below). Note the local truncation error is \( O(h^2) \), and so the local truncation error decreases quadratically with the step size.

The quantity \([1 + hf_y(t_n, \eta)]E_n\) in (11) represents the error at step \( n+1 \) caused by the error at step \( n \). The propagated error is larger than \( E_n \) if \( f_y > 0 \) and smaller than \( E_n \) if \( f_y < 0 \). For this reason we say that the IVP 4 is well-conditioned if \( \partial f / \partial y < 0 \) and ill-conditioned if \( \partial f / \partial y > 0 \). The terminology comes from the general rule that having \( f_y < 0 \) is a good thing since it causes truncation errors to diminish as they propagate.

Since we don’t know \( \eta \), and, hence, \( f_y(t_n, \eta) \), we would like to take (11) further. Using (4) we note

\[
y'' = f_t + f_y y' = f_t + f_y f
\]

and let

\[
K = \max_{(t,y) \in R} |f_y(t,y)| < \infty, \quad M = \max_{(t,y) \in R} |(f_t + f_y)(t,y)| < \infty.
\]

(12)

We can actually solve this recursive inequality. Observe

\[
E_0 = 0
\]

\[
|E_1| \leq \frac{h^2}{2} M
\]

\[
|E_2| \leq (1 + |1 + hK|) \frac{h^2}{2} M
\]

\[
|E_3| \leq (1 + |1 + hK| + |1 + hK|^2) \frac{h^2}{2} M
\]

\[
|E_n| \leq \frac{h^2}{2} M \sum_{j=0}^{n-1} (1 + hK)^j.
\]

The last estimate for \( |E_n| \) can be worked out using the formula for the geometric series:

\[
|E_n| \leq \frac{hM}{2K} ([1 + hK]^n - 1)
\]

Since \( t_n = nh \), we conclude

\[
|E_n| \leq \frac{hM}{2K} \left([1 + hK]^n - 1\right) \xrightarrow{h \to 0} \frac{hM}{2K} \left(e^{Kt_n} - 1\right).
\]

With this, we conclude with the following Theorem

**Theorem 2** Let \( I \) and \( R \) be defined as in Theorem 1 and let \( K \) and \( M \) be defined as in equation (12). Let \( \{y_1, \cdots, y_N\} \) be the Euler approximations at the points \( t_n = nh \). If \((t_n, y_n), (t_n, Y(t_n)) \in R \) and \( Kh < 1 \), then the local truncation error \( T_n = (h^2/2)Y''(\tau_n) \) from step \( n \) to \( n+1 \) is bounded by

\[
|T_n| \leq \frac{M}{2} h^2
\]

and the global truncation error \( E_n = Y(t_n) - y_n \) in the approximation of \( Y(t_n) \) is bounded by

\[
|E_n| \leq \frac{M}{2K} \left(e^{Kt_n} - 1\right) h.
\]
There are two important conclusions to be made from Theorem 2. The first is that the error $|E|$ vanishes as $h \to 0$. Thus, the truncation error can be made arbitrarily small by decreasing $h$ (although doing so may create large round-off errors). The second is that the error $|E|$ in Euler’s method is approximately proportional to $h$, making Euler’s method a first-order numerical scheme.

**Example 1 (revisited).** Consider again the initial value problem

$$
\begin{align*}
\dot{y} + 2y^2 &= \frac{1}{1 + t^2} \\
y(0) &= 0
\end{align*}
$$

(13)

Recall that the exact solution to (13) is given by

$$
Y(t) = \frac{t}{1 + t^2}.
$$

Observe that $\frac{\partial f}{\partial y}(t, y) = -4y$ and thus the IVP (13) is *well-conditioned for* $y > 0$ and *ill-conditioned for* $y < 0$. In Figure 3 we can see the behaviour described in Theorem 2. In particular, we see that each time we double $N$ or, equivalently, half our step size $h$, the maximum error $\max_j |E_j|$ decreases by roughly $1/2$. Quantitatively, if $E(N)$ denotes the maximum error using a step size of $h = 2/N$, we find

$$
\begin{align*}
\frac{E(5)}{E(10)} &= 2.3684, & \frac{E(10)}{E(20)} &= 2.0995, & \frac{E(20)}{E(40)} &= 2.0320, & \frac{E(40)}{E(80)} &= 2.0203.
\end{align*}
$$


![Figure 3](image-url)

**Figure 3:** For $N = 5, 10, 20, 40, 80$ we plot the corresponding error $|E_j| = |y_j - Y(t_j)|$ in the interval $[0, 2]$. Note how the error at each time step is decreasing as we decrease our step size (or, equivalently, increase $N$).

We now ask the following question: how many subdivisions are needed to assure that the Euler approximation to (13) on the interval $[0, 2]$ has error no greater than 0.001?

We may use Theorem 2, and to apply this Theorem, we need to determine $K$ and $M$ for our problem. Here $f(t, y) = -2y^2 + 1/(1 + t^2)$ is a smooth function in all its variables, so we have no problem with the hypotheses of Theorem 2. We take $R = [0, 2] \times [-1, 1]$, and we have that $f_y(t, y) = -4y$ and

$$
(f_t + f_y)(t, y) = -\frac{2t}{(1 + t^2)^2} - \frac{4y}{1 + t^2} + 8y^3.
$$
Using basic calculus, we can find $K = 4$ and $M = 176/25$. Therefore, since $t_N = 2$,

$$|E_j| \leq \frac{22}{25} (e^{4t_j} - 1) h \implies |E_N| \leq \frac{22}{25} (e^8 - 1) h \approx 2622h$$

Thus, setting $2622h \leq 1/1000$, we find that the sufficient step size required to achieve our desired error tolerance is

$$h = \frac{1}{2622000} \implies N = \frac{2}{h} = 5244000.$$

This conclusion demonstrates an important rule. The number $N$ above is huge compared to what we will have to use in practice; the important part of Theorem 2 is the order of the method, not the estimate itself. For example, we see from Figure 3 that we already almost achieved the desired error tolerance with $N = 80$.

This brings up another important point: **round-off error.** Computers cannot store an infinite amount of information, and so for a given real number will be represented by a truncated version of the actual number. In other words, $\pi$ on a computer is only given to a certain number of digits (this number is sometimes referred to as the machine-precision). Hence, we cannot just continually shrink the step size ad infinitum, as we will eventually introduce nontrivial round-off error in the extremely small numbers being computed with.
3 Modified Euler and Runge-Kutta Methods

In this section we review improvements to the Euler scheme (2). In particular, we look at the Heun’s method (also known as modified Euler) and the popular fourth-order Runge-Kutta (RK) method (known as RK4). RK methods fall into a class of numerical methods called single-step methods. Single-step methods for solving \( y' = f(t, y) \) require only a knowledge of the numerical solution \( y_n \) in order to compute the next value \( y_{n+1} \). These should be compared to the so-called multistep methods, which require several past values to find the next approximation. Both approaches have their relative advantages and disadvantages.

We work our way to Heun’s method and more general RK methods by revisiting the Taylor series derivation of Euler’s method. Recall the initial value problem we deal with is

\[
\begin{aligned}
\begin{cases}
y' &= f(t, y) \\
y(t_0) &= Y_0 
\end{cases}
\end{aligned}
\]

We want to approximate the exact solution to (14) on \([t_0, t]\) for \( t > t_0 \). As before, we let \( h > 0 \), and partition \([t_0, t]\) into \( N = N(h) \) subintervals \([t_0, t_1], [t_1, t_2], \ldots, [t_N, t]\) where

\[
t_j = t_0 + jh, \quad j = 0, 1, \ldots, N.
\]

We assume that the exact solution \( Y(t) \) to (14) on \([t_0, t]\) is \( r + 1 \)-times continuously differentiable. Using Taylor’s theorem, we expand \( Y(t_{n+1}) = Y(t_n + h) \) about \( t_n \) and find

\[
Y(t_{n+1}) = Y(t_n) + hY'(t_n) + \frac{h^2}{2}Y''(t_n) + \cdots + \frac{h^r}{r!}Y^{(r)}(t_n) + \frac{h^{r+1}}{(r+1)!}Y^{(r+1)}(\xi_n)
\]

for some \( t_n \leq \xi_n \leq t_{n+1} \). Every derivative \( Y^{(j)} \) for \( 1 \leq j \leq r \) can be computed by using the initial value problem (14). I.e., \( Y'(t_n) = f(y(t_n), Y(t_n)) \),

\[
Y''(t_n) = (\partial_t f + f \partial_y f)(t_n, Y(t_n)),
\]

and so forth. Thus, we can truncate the Taylor series to obtain a numerical scheme that generalizes Euler’s method:

\[
y_{n+1} = y_n + \sum_{j=0}^{r} \frac{h^{j+1}}{(j+1)!} \left( \frac{d^j f}{dt^j} \right) (t_n, y_n), \quad t_{n+1} = t_n + h, \quad y_0 = Y_0.
\]

(15)

For \( r = 1 \), (15) reduces to Euler’s method (2).

The RK methods are closely related to the Taylor series expansion (15), however they circumvent the need to calculate the derivatives of \( f \). From here one can go about deriving the RK methods. This is a little tedious and beyond the scope of these notes. Thus we simply record two special cases of RK methods: Heun’s method and RK4.

The first is Heun’s method, a second-order RK method. The idea is to first calculate an intermediate value \( \tilde{y}_{n+1} \) and then use this to average the vector field \( f \) to have a better approximation to the slope of \( Y \) over the interval \([t_n, t_{n+1}]\). Heun’s method reads

\[
y_{n+1} = y_n + \frac{h}{2} [f(t_n, y_n) + f(t_{n+1}, \tilde{y}_{n+1})], \quad t_{n+1} = t_n + h, \quad y_0 = Y_0.
\]

(16)

where

\[
\tilde{y}_{n+1} = y_n + hf(t_n, y_n)
\]

(17)

Here \( \tilde{y}_{n+1} \) is an intermediate value that we calculate to find a better approximation \( y_{n+1} \) to \( Y(t_{n+1}) \). We emphasize that Heun’s method is a second-order numerical scheme; i.e., the total accumulated error \( E_n = Y(t_n) - y_n \) is \( O(h^2) \).

We take this fact for granted.

The single-step fourth-order RK method, known as RK4, reads

\[
y_{n+1} = y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4), \quad t_{n+1} = t_n + h, \quad y_0 = Y_0.
\]

(18)
where

\[ k_1 = f(t_n, y_n) \]  \hspace{1cm} (19)
\[ k_2 = f \left( t_n + \frac{h}{2}, y_n + \frac{k_1}{2} \right) \]  \hspace{1cm} (20)
\[ k_3 = f \left( t_n + \frac{h}{2}, y_n + \frac{k_2}{2} \right) \]  \hspace{1cm} (21)
\[ k_4 = f \left( t_n + h, y_n + k_3 \right) \].  \hspace{1cm} (22)

In words, RK4 specifies the next approximation \( y_{n+1} \) by computing a weighted-average of the vector field \( f \) in the window \([t_n, t_{n+1}] \times [y_n, y_n + k_3]\). Here \( k_1 \) is the increment based on the slope at the beginning of the interval, using \( y \) (Euler’s method); \( k_2 \) is the increment based on the slope at the midpoint of the interval, using \( y \) and \( k_1 \); \( k_3 \) is again the increment based on the slope at the midpoint, but now using \( y \) and \( k_2 \); \( k_4 \) is the increment based on the slope at the end of the interval, using \( y \) and \( k_3 \). We emphasize that RK4 is a fourth-order numerical scheme; i.e., the total accumulated error \( E_n = Y(t_n) - y_n \) is \( O(h^4) \). We take this fact for granted without proof. Figure 4 compares the global truncation error \( E_n \) between the methods discussed above.

**Figure 4:** The above Figure highlights how the maximum error changes as a function of the step size \( h \) between several different methods for some initial value problem. The blue curve is Euler’s method, the orange curve is a second-order Runge-Kutta method (similar to modified Euler), and the yellow curve is RK4. One can see how the error in RK4 almost completely diminishes around a step size of \( h \approx 10^{-4} \). This is impressive when compared to Euler method. However, note that as the step size is lower below \( 10^{-4} \) the error in RK4 begins to grow. This is likely due to round-off error in the computation.
4 Problems and Exercises

Throughout we consider the initial value problem

$$\begin{cases}
\dot{y} = f(t, y) \\
y(0) = Y_0
\end{cases}$$

on the interval $[0, T]$, $T > 0$.

1) Consider the following variant of Euler’s method for solving (23), known as **backward Euler**:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \quad t_{n+1} = t_n + h = nh.$$ 

Backwards Euler is an *implicit* numerical method (meaning we have to solve an algebraic equation to find the unknown $y_{n+1}$), whereas (forward) Euler’s method is an *explicit* numerical method. Below are some questions that highlight some striking differences between forward Euler and backward Euler. These differences are related to the *stability* of a numerical method. Let $f(t, y) = -2y$, $Y_0 = 1$.

a) Without a computer, apply Euler’s method to this IVP with $T = 10$ and $h = 1$. Plot the approximate solution over the exact solution. Discuss the resulting behaviour.

b) Without a computer, apply backward Euler to this IVP with $T = 10$ and $h = 1$. Plot the approximate solution over the exact solution. How does it compare to the result from part (a)?

[Remark: If we use a smaller step size (say, $h = 0.1$), then forward Euler gives the right qualitative behaviour.]

2) Consider the following Taylor series method for solving (23):

$$y_{n+1} = y_n + hf(t_n, y_n) + \frac{h^2}{2} (\partial_t f + f \partial_y f)(t_n, y_n), \quad t_{n+1} = t_n + h = nh.$$ 

a) Write a Matlab function implements this method for $f(t, y) = -2y^2 + 1/(1 + t^2)$, $Y_0 = 0$, with variable interval and step size. The code should output $|E_n|$ at each time step, as well as plot the exact solution versus the approximation.

b) Compare the results to those of Euler’s method used in Example 1 (revisited).

c) Use this function to experiment and determine the order of the method.

d) Using a strategy similar to the one we did for Euler’s method in §2, prove that the order of the method is what you inferred in part (b).

3) Write a Matlab function that implements the modified Euler method for $f(t, y) = y(1-y)$, $Y_0 = 0.1$, with variable interval and step size. The code should output $|E_n|$ at each time step, as well as plot the exact solution versus the approximation. By running some experiments with this code, verify that this is a second-order method.

4) Write a Matlab function that implements RK4 for $f(t, y) = te^{-y^2}$, $Y_0 = 0$, with variable interval and step size. The code should output $|E_n|$ at each time step, as well as plot the exact solution versus the approximation. By running some experiments with this code, verify that this is a fourth-order method.
5 Appendix

function [tsteps, err]=EulerDemo0(N,T)
% This function applies Euler’s method to the problem
% \( y' = \frac{1}{1+t^2} - 2y^2 \), \( y(0) = 0 \)
% on the interval \([0,T]\)
% using N steps.
% The output is the error at each time step, along with a graph.
% The blue points mark the Euler approximation
% the black curve is the exact solution.

y0=0;
h=T/N;
t=[0:h:T];
Y=yexact(t);
y(1)=y0;

for n=1:N
  y(n+1)=y(n)+h*f(t(n),y(n));
end

plot(t,Y,'k','LineWidth',2)
hold on
plot(t,y,'r--o')
hold off
err=abs(Y-y);
tsteps=t;

function f=f(t,y)
f=1/(1+power(t,2))-2*power(y,2);

function y=yexact(t)
y=t./(1+t.*t);

References