Chapter 4. A simple method for solving first order equations numerically

We shall discuss a simple numerical method suggested in the introduction, where it was applied to the second order equation associated to a falling object. We shall also deal with the problem of estimating error in these approximate solutions. This method has the virtue of being easy to understand, but it is very inefficient. Later we shall see more practical methods.

1. Euler's method

Suppose we want to find approximate values for the solution of the differential equation

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

with initial condition

$$y(t_0) = y_0 \ .$$

The differential equation tells us that the instantaneous rate of change of y at time t can be calculated in terms of y and t alone. Now if Δt is any small interval of time we know that as a first order approximation we can write

$$y(t + \Delta t) \doteq y(t) + \Delta t \cdot y'(t)$$

in the sense that the difference between the right and left sides, for a fixed value of t, is essentially of order $(\Delta t)^2$. If y is a solution of the differential equation, this tells us that

$$y(t + \Delta t) \doteq y(t) + \Delta t \cdot f(t, y(t)) \ .$$

If $t_1 = t_0 + \Delta t$ we know therefore that

$$y(t_1) \doteq y_1 = y(t_0) + \Delta t \cdot f(t_0, y(t_0)) = y_0 + \Delta t \cdot f(t_0, y_0) \ .$$

We can apply this reasoning once again. If $t_2 = t_1 + \Delta t = t_0 + 2\Delta t$ we obtain a further estimate

$$y(t_2) \doteq y_2 = y_1 + \Delta t \cdot f(t_1, y_1) \ .$$

Etc. If $t_{n+1} = t_n + \Delta t = t_0 + (n+1)\Delta t$ we get an estimate y_{n+1} for $y(t_{n+1})$ from the estimate y_n for $y(t_n)$:

$$y_{n+1} = y_n + \Delta t \cdot f(t_n, y_n) \ .$$

This technique for finding approximate values for the solution of a first order differential equation is the simplest of several similar ones. Each of them proceeds in this stepwise fashion, obtaining an estimate for $y(t + \Delta t)$ from that for $y(t)$. This one is called **Euler's method**. It has some great virtues: • It is simple to carry out. Doing it by hand is not impossible (although tedious and hence error prone). • It can be easily implemented on a computer, for example with a spread sheet. • The step from one estimate to the next is intuitive.

The calculations are easy to set up in a spreadsheet, equally easy to program, not even too bad to do a few steps by hand. Here is a sample run for the equation & initial conditions

$$y' = y - t, \quad y(0) = 0.5 \ .$$

with step size $\Delta t = 0.1$, in the range $[0, 1]$.

t	y	$f(t, y)$
0.000000	0.500000	0.500000
0.100000	0.550000	0.450000
0.200000	0.595000	0.395000
0.300000	0.634500	0.334500
0.400000	0.667950	0.267949
0.500000	0.694744	0.194745
0.600000	0.714220	0.114220
0.700000	0.725641	0.025641
0.800000	0.728206	-0.071794
0.900000	0.721026	-0.178974
1.000000	0.703129	-0.296871

Exercise 1.1. *Carry out two steps of Euler's method to estimate $y(1)$, for the differential equation & initial condition*

$$y' = y - t, \quad y(0) = 0.5 \ .$$

Then four steps.

2. Euler's method and slope fields

Euler's method has a simple geometric interpretation. Solving a differential equation

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

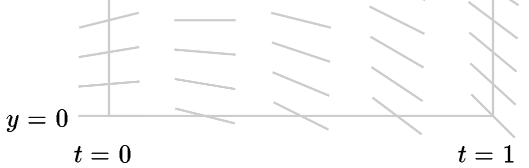
means, geometrically, finding the graph of a function $y = y(t)$ whose slope at any point (t, y) is equal to $f(t, y)$. We can picture this by drawing a number of small 'slope segments' in the (t, y) plane, the segment at (t, y) having slope $f(t, y)$. These segments suggest to the eye what solutions of the differential equation look like. Euler's method has a direct interpretation in terms of such slope fields. In the figure below we are looking at the differential equation & initial condition

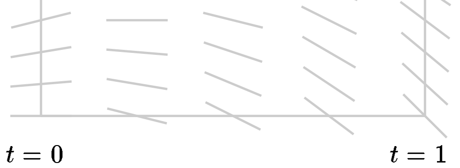
$$y' = y - t, \quad y(0) = 0.5 \ .$$

Note that this is a linear equation and has the explicit solution

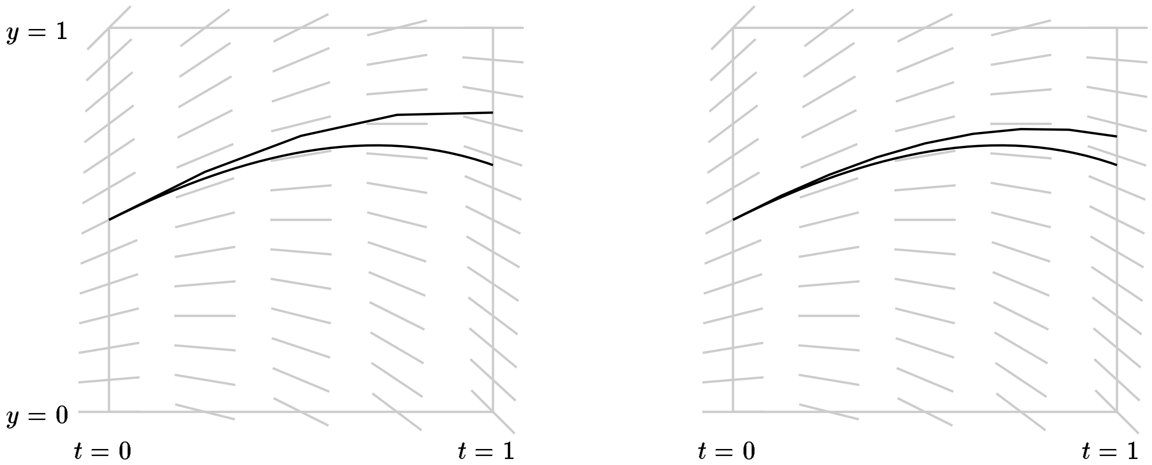
$$y = (t + 1) - 0.5\, e^t \ .$$

The differential equation tells us how to draw the slope field and the initial condition tells us where the graph starts.





At the starting point we know that the slope of the graph is equal to $f(0, 0.5) = 0.5$. This means that the tangent line to the graph at the starting point is the line $y_*(t) = 0.5 + 0.5\,t$. This tangent line and the graph of $y(t)$ will lie very close to each other for small values of t, and we can then use y_* as an estimate for $y(t)$ for those values of t. After a while, however, it will cease to be a useful approximation. How can we get a better one? We pick some interval Δt and at $t = \Delta t$ we modify the tangent line. We must modify it by using only information available to us in the calculation so far, and in view of this it seems reasonable to use as the new approximation the straight line through $(\Delta t, y_*(\Delta t))$ whose slope agrees with the slope field at that point.

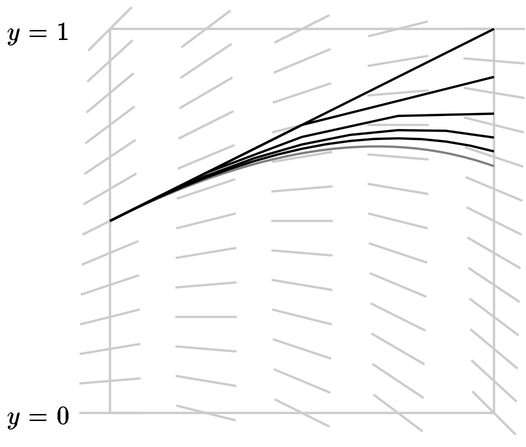


We then make new breaks in the approximation after every interval of size Δt. As we take smaller and smaller values of Δt we get better approximations to the graph of the solution.

3. The error in Euler's method

We expect that the error in the approximation we get from Euler's method decreases as we let the **step size** Δt get smaller. We can get a good idea of how the error depends on the choice of Δt by plotting in one picture the approximations we get for different values of Δt. Here we let $\Delta t = 1$, $1/2$, $1/4$, etc. again with the differential equation

$$y' = y - t, \quad y(0) = 0.5\ .$$



It looks very much as though we halve the error as we halve the step size. We can verify this in more detail by comparing estimates for $y(1)$ with the true value which we know to be $(t+1) - 0.5\,e^t$ at $t = 1$ which is equal to $2 - 0.5\,e = 0.640859$.

N	estimated $y(1)$	true $y(1)$	error
2	0.875000	0.640859	0.234141
4	0.779297	...	0.138438
8	0.717108	...	0.076249
16	0.681036	...	0.040177
32	0.661505	...	0.020646
64	0.651328	...	0.010468
128	0.646130	...	0.005271
256	0.643504	...	0.002645
512	0.642184	...	0.001325
1024	0.641522	0.640859	0.000663

This example leads to a guess which turns out to be true:

- *The error in Euler's method for estimating $y(t)$ with a given differential equation and initial condition of y at a fixed value of t is proportional to the step size chosen.*

The effect of this is to make Euler's method extremely inefficient for serious calculation. It requires an enormous amount of calculation to achieve reasonable accuracy. Very roughly speaking, the amount of work it takes to achieve a certain level of accuracy is proportional to the accuracy you want. Thus if it takes N steps to get 1 digit of accuracy (answer within $1/10$), it will take $10\,N$ to get an extra valid digit (within $1/100$), $100\,N$ to get 3 valid digits, . . . , and $100000\,N$ to get 6.

The reason for the way error and step size interact for Euler's method is not hard to understand at least informally. Euler's method uses the estimate

$$y(t + \Delta t) \doteq y(t) + \Delta t \cdot y'(t) + \text{ terms of order } (\Delta t)^2 \ .$$

That means that the error in making each step is essentially of order $(\Delta t)^2$. But for a fixed interval across which we want to calculate, the number of steps necessary is proportional to the inverse of Δt, which makes it plausible that the overall error is of order $(1/\Delta t)(\Delta t)^2 = \Delta t$.

Exercise 3.1. *If we use Euler's method to solve*

$$y' = -y^2 + t, \quad y(0) = 0.5$$

we get the following estimates:

Δt	$y(1)$
0.100000	0.682953
0.050000	0.693979
0.025000	0.699202

Approximately how many steps would it take to estimate $y(1)$ correctly to 6 decimals, using Euler's method?

4. Illustrating errors graphically

We can use this to estimate the error in Euler's method even when we don't know the exact answer! For example, suppose we look at the equation

$$y' = xy + 1, \quad y(0) = 1$$

This is a linear equation, but if we were to use a standard formula for the solution we would see it produces an integral we cannot find explicitly. Here is the estimate we get for $y(1)$ with various values of N:

N	$y(1)$
4	2.65515137
8	2.83657060
16	2.94189329
32	2.99898700
64	3.02876204
128	3.04397338
256	3.05166223
512	3.05552774
1024	3.05746579

And here is a graph of the pairs $(\Delta, y(1))$.

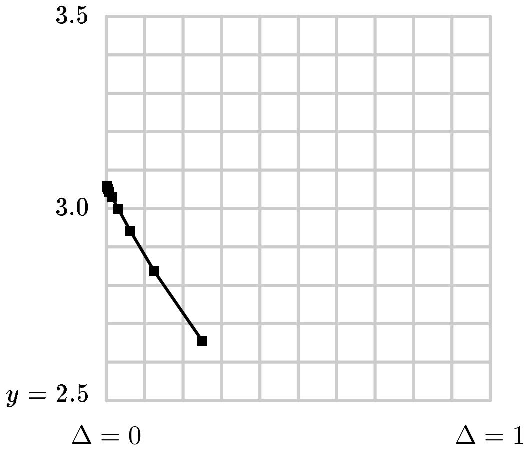


Figure 4.1. *A plot of $y(1)$ versus step size.*

What we can see from this plot is that the points $(\Delta, y(1))$ are to a good approximation on a straight line. This is in complete agreement with the principle I stated above, that the error is proportional to Δ. But we can also see that to some extent we can predict what the end of the line is. This is because we can estimate from each successive pair what the slope of the purported straight line is—it ought to be about

$$\frac{y_N - y_{N-1}}{\Delta_N - \Delta_{N-1}}$$

for each N. Of course this is only an approximation, so we don't expect any of these estimates to be exact. Here is what we get in the runs above:

N	$\Delta_N = 1/N$	$y_N(1)$	$y_N(1) - y_{N-1}(1)$	estimated slope
4	0.25000000	2.65515137	$*$	$*$
8	0.12500000	2.83657060	0.18141923	-1.45135383
16	0.06250000	2.94189329	0.10532269	-1.68516309
32	0.03125000	2.99898700	0.05709371	-1.82699868
64	0.01562500	3.02876204	0.02977504	-1.90560247
128	0.00781250	3.04397338	0.01521134	-1.94705210
256	0.00390625	3.05166223	0.00768885	-1.96834562
512	0.00195312	3.05552774	0.00386551	-1.97913877
1024	0.00097656	3.05746579	0.00193806	-1.98457249

The rough proportionality certainly shows up here, and the last column tells us roughly what the constant of proportion is.

- *One run of Euler's method gives you no idea of the error involved.*
- *Two runs with values y_1 and y_2 for step sizes Δ_1 and Δ_2 give you the estimate*

$$\left[\frac{y_1 - y_2}{\Delta_1 - \Delta_2}\right] \Delta$$

for the error with step size Δ.

Exercise 4.1. *If we use Euler's method to solve*

$$y' = -y^2 + t, \quad y(0) = 0.5$$

we get the following estimates:

Δt	$y(1)$
0.100000	0.682953
0.050000	0.693979
0.025000	0.699202

Using the data above, what is the best estimate you can make for $y(1)$?

5. A program to carry out Euler's method

The following Java program will do to carry out Euler's method, in this case for the differential equation & initial condition

$$y' = -y + t, \quad y(0) = 0.5\ .$$

It is very primitive, and requires recompiling each time any of the initial conditions, step size, or differential equation are changed.

To compile this: **`javac euler.java`**. To run it: **`java euler`**.

```
public class euler {

    public static void main(String arg[]) {
        int N = 10;

        double h = 1.0/N;
        double x0 = 0;
        double y0 = 0.5;

        double x = x0, y = y0;

        for (int i=0;i < N;i++) {
            y += h*f(x, y);
            x += h;
            System.out.println("x, y = " + x + ", " + y);
        }
    }

    static double f(double t, double y) {
        return(-y + t);
    }
}
```