

Math 2280 - Lecture 10

Dylan Zwick

Summer 2013

I've decided not to focus much on computer and numerical methods in this class, and so consequently we will not be covering sections 2.5 or 2.6 of the textbook, nor will we have any computer projects. However, I would be guilty of instructor negligence if I were to teach an introductory differential equations class without at least mentioning the grandfather of all numerical methods - Euler's method. This simple, and yet foundational, numerical approximation method will be the subject of today's lecture.

This lecture corresponds with section 2.4 from the textbook. The assigned problems for this section are:

Section 2.4 - 1, 5, 9, 26, 30

Euler's Method

For the first order differential equations we've seen so far, most of them have had the form:

$$\frac{dy}{dx} = f(x, y)$$

and most of them we've been able to solve using one technique or another. Don't get too comfortable. Differential equations for which we've got explicit solutions are really the exception, rather than the rule, and even for relatively simple looking differential equations it may be impossible to figure out the solution. For example:

$$\frac{dy}{dx} = e^{-x^2}$$

has no solution $y = f(x)$ where $f(x)$ is an elementary function. By elementary function we mean a function that can be expressed in terms of the standard functions (exponentials, cosines, logarithms, polynomials, etc...) from calculus.

Take heart, all is not lost. Even in situations where we cannot figure out the explicit solution, we can frequently construct approximations. One of the oldest, and foundational, methods used to figure out approximate solutions is called "Euler's Method", named after the (extremely) prolific and influential mathematician Leonhard Euler.¹

The Euler's Method Algorithm

The idea behind Euler's method is pretty simple. We're given an initial value problem:

$$\begin{aligned}\frac{dy}{dx} &= f(x, y) \\ y(x_0) &= y_0\end{aligned}$$

and we want to construct an approximate solution. We construct the approximate solution by starting with our initial condition, and taking the slope at that point $f(x_0, y_0)$. We assume that slope will be constant over a small change in x , and so we move forward a small distance h in the x -direction. The size of h is called the "step size" for our implementation.

If we move an amount h in the x -direction, and our slope is $f(x_0, y_0)$, then we move an amount $h \times f(x_0, y_0)$ in the y -direction. This gives us an approximation for the value of our solution for the input value $x_0 + h$, namely $y_0 + h \times f(x_0, y_0)$. We call this approximated point (x_1, y_1) . Then from there, we just continue in the same fashion.²

¹It is sometimes joked that for all the formulas and techniques in undergraduate math that have a named attached to them, like "Laplace transforms", the rule for assigning the name is that credit is given to the first mathematician *after Euler* to discover the idea.

²Wash, rinse, repeat...

What we get is a sequence of line segments that approximate our solution curve. If our step size is very small, this sequence of line segments looks more and more like a curve, and (in theory) they get closer and closer to our actual solution curve.

As a first, simple example, let's say we start with the differential equation:

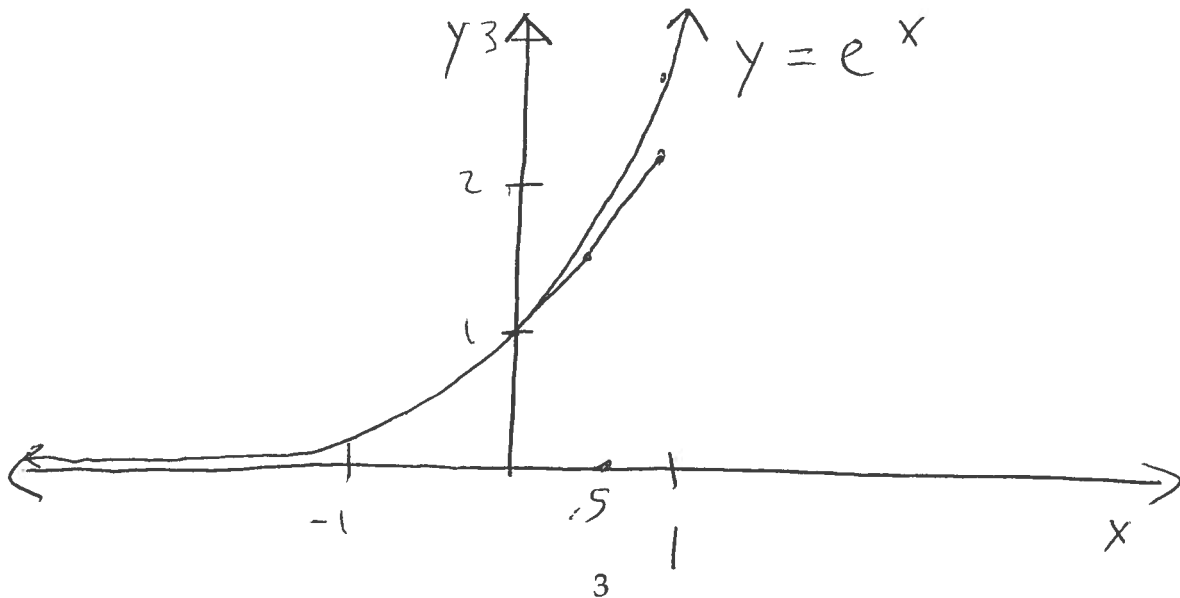
$$\frac{dy}{dx} = y$$

$$y(0) = 1.$$

This initial value problem has solution $y(x) = e^x$. Let's check out what Euler's method predicts for the solution at $x = 1$ using a step size of $h = .5$. Applying the Euler's method algorithm we get the following table:

n	x_n	y_n	$f(x_n, y_n)$	e^{x_n}
0	0	1	1	1
1	.5	1.5	1.5	1.65
2	1	2.25	2.25	2.72

So, we can see that Euler's method gives an O.K. solution here, but the approximation isn't great.



If we do this again, but use a smaller step size $h = .1$, we have to do quite a few more calculations, but our estimate improves:

n	x_n	y_n	$f(x_n, y_n)$	e^{x_n}
0	0	1	1	1
1	.1	1.1	1.1	1.11
2	.2	1.21	1.21	1.22
3	.3	1.33	1.33	1.35
4	.4	1.46	1.46	1.49
5	.5	1.61	1.61	1.65
6	.6	1.77	1.77	1.82
7	.7	1.95	1.95	2.01
8	.8	2.14	2.14	2.23
9	.9	2.36	2.36	2.46
10	1	2.59	2.59	2.72

So, as we can see, by taking a smaller step size we get closer to the actual value, although again we're a little bit off. If we took an even smaller step size, say $h = .01$, we'd get even closer.

Now, this example has been with a very simple differential equation, for which we've already got the solution, but it illustrates the method. For more complicated problems with many more steps these computations can get very, very tedious and time consuming. It's work best given to a machine.

Example - The initial value problem

$$y' = 2y$$

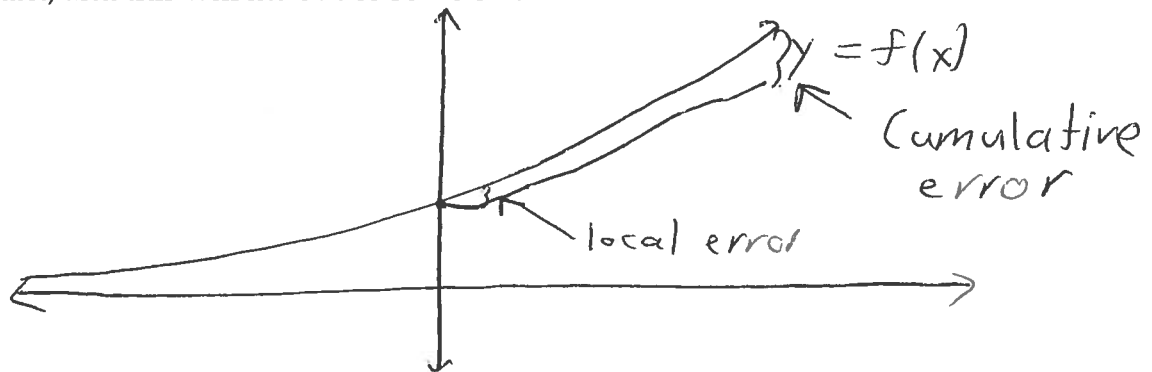
$$y(0) = \frac{1}{2}$$

has the exact solution $y = \frac{1}{2}e^{2x}$. Apply Euler's method twice to approximate this solution on the interval $[0, \frac{1}{2}]$, first with step size $h = 0.25$, then with step size $h = 0.1$. Compare the three-decimal-place values of the two approximations at $x = \frac{1}{2}$ with the value $y(\frac{1}{2})$ of the actual solution.

Sources of Error

There are two major sources of error in the use of Euler's method, called *local error* and *cumulative error*.

Local error is the result of our assumption that the slope is constant over our small step size h . Now, if our function $f(x, y)$ is continuous and our step size is small, this isn't an unreasonable assumption, but it's not exact, and this will introduce some error.



The other source of error is cumulative error. Because in each of our steps we introduce some local error, the starting points from which we calculate the slopes for each step are also not quite right, and so the slopes we calculate are not quite right, and so this introduces even more error. The overall cumulative effect of this error is called, not terribly creatively, cumulative error. It's meant to represent the total error (distance) of our approximate solution from the actual solution.

Now, frequently we don't know what the actual solution is, and so we just want to know that our approximation is within some range of values of the actual solution. The study of this type of situation along with related situations is a topic for an entire class on error analysis. A necessary class if you want to be an engineer, but I must admit it sounds like a tremendously boring one to me.

Finally, even if you make very certain that you're getting close to the actual result (taking a very small step size) this can introduce problems. The first is that even for a computer extremely small step sizes can lead to long computation times, but the other problem can be more pernicious. The computer rounds, and this rounding inevitably introduces some error. So, if you're dealing with very small step sizes, you're dealing with very small numbers, and these numbers are frequently rounded, which introduces errors that can accumulate over time. In fact, problems of this nature led to the first results in what we now call "chaos theory".