THE INTRODUCTION WITH MONEY EXAMPLE

It has always been of great interest to me how to make defining and solving differential equations easier to learn and to use in practice. This short introduction lays out my thinking on how to accomplish that.

What is one of the first real-life problems you think of. As an electrical engineer you might think something related to V=IR, like a capacitor charging. But how about something important to everyone, how fast will my money grow?

It is easy to find formulas, but where does the growth come from?

Think of the amount of money in your bank account in terms of a rate of change. That is, the instantaneous amount of money,say B, in your account is dB/dt = interest rate times the amount of money in the accout.

$$\frac{d}{dt}(B) = rate\ per\ period \cdot Money$$

$$\frac{d}{dt}(B) = rate \cdot B(t)$$

A typical rate might be 6 percent per year. The rule of 72 says that it will take about 72/6 or 12 years to double your money. To see rapid growth in this example let's have the rate be 18 percent per time period, a year in this case. How do you solve for the actual time it takes to double your money?

Well, can search the internet for it, but that won't let you solve other problems of interest. So, how to get started?

The equation above says that dB the change in bank balance divided by the change in time t is equal to something. Change over change reminds me of rise over run in the basis description of a line. It is called the point-slope form. It is often written as

$y - y0 = m \cdot (x - x0)$ where m is the slope (rise/run).

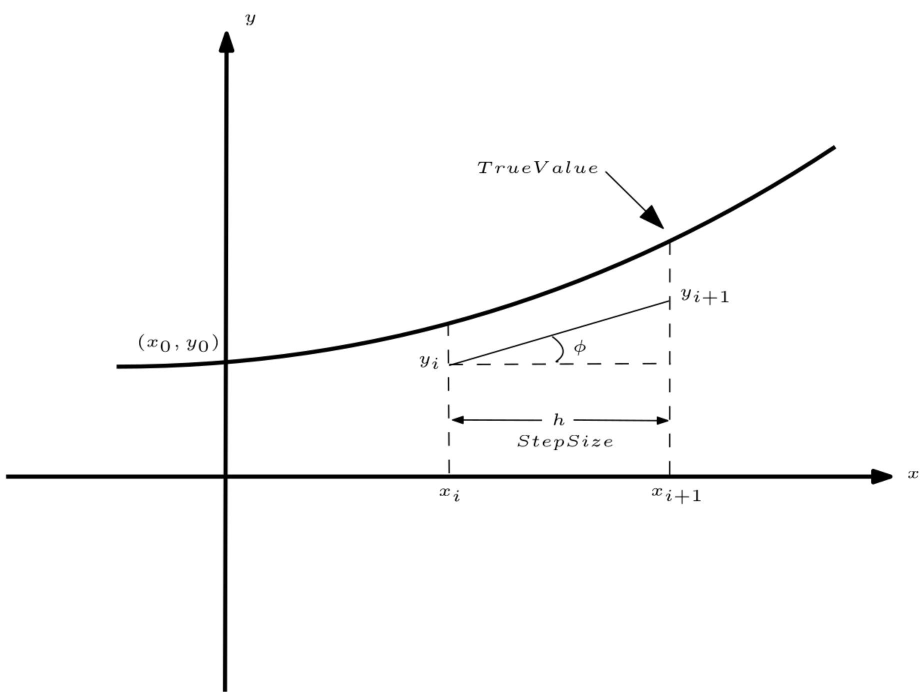
Now, let's let $h = x - x0$ (an increment along the x axis) and the slope $m = f(x0, y0)$. Now we have

$y = y0 + f(x0, y0)$ or

$$y_{n+1} = y_i + h \cdot f(x_i, y_i)$$

This last expression is referred to as Euler's Method. It is a simple iterative process and can be implemented on standard equations or differential equations with initial conditions, as well will soon see.

For reference, here is a graphical view of the Euler Method.



Now, we have a step-by-step method to update the solution to a differential equation. Before we implent that solution, how can we visualize not only that solution but all solutions for a differential equation? From above, remember that everything to the right of the equals sign is nothing more than a slope. So, for every point in an (x, y) plane we can plot the slope or $f(x, y)$ or its slope.

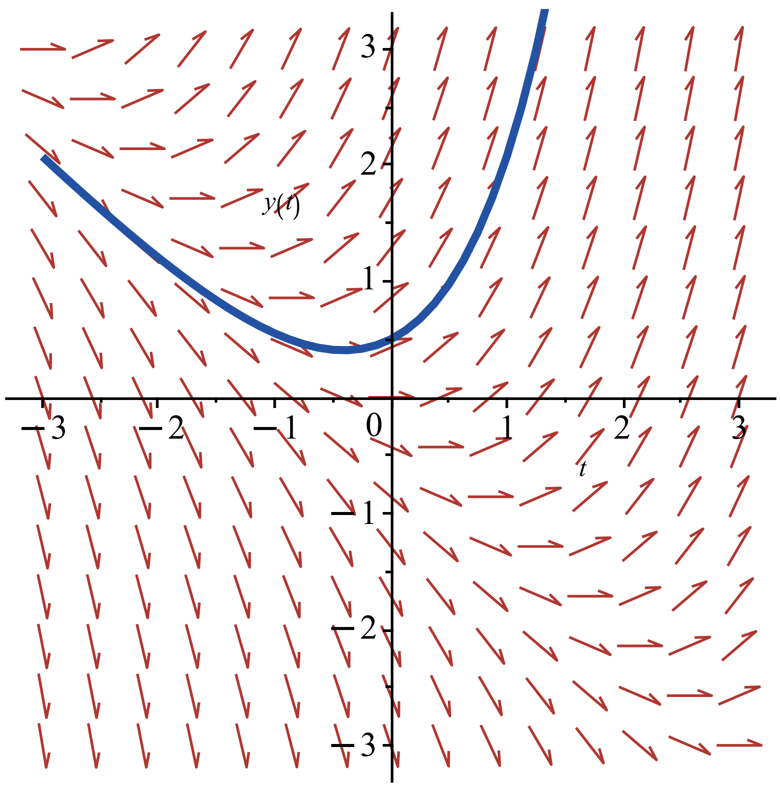
These plots are called direction fields and are very helpful in understanding the behavior of a solution to a differential equation.

Let's pick a differential equation that is easy to visualize and plot by hand if wanted.

Let $\frac{d}{dt}(y) = x + y$ which is straightforward to plot. We wil use Maple here to plot the direction field. Both MATLAB and Mathematica can also plot direction fields.

restart

with(*DETools*) :
DEplot(*diff* (*y*(*t*), *t*) = *y*(*t*) + *t*, *y*, *t* = −3 ..3, *y* = −3 ..3, *dirfield* = [15, 15], [*y*(0) = 0.5], *size* = [300, 300]);



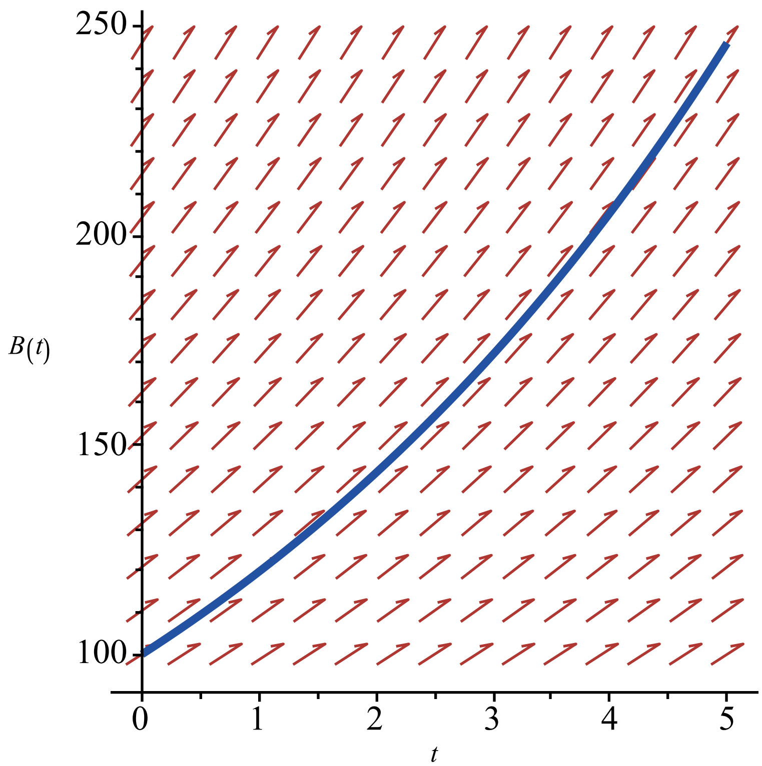
Here, you can easily see where the direction field vectors have zero slope, i.e. where $x + y = 0$. The solutions start on the left, cross the $y - axis$, then go upwards. The blue line is simply the unique solution where the initial condition is $y = 0.5$. What this means it that, once you choose where the solution crosses the $y - axis$, you have set the unique solution for the differential equation.

What does the direction field look like for our money problem? The initial balance in the account is 100.

restart
with(*DETools*) :
eq1 := *diff*(*B*(*t*), *t*) = 0.18·*B*(*t*);

$$eq1 := \frac{d}{dt} B(t) = 0.18\, B(t) \quad \textbf{(1)}$$

ics := [*B*(0) = 100] :
DEplot(*eq1*, *B*(*t*), *t* = 0 ..5, *ics*, *dirfield* = [15, 15], [*B*(0) = 100], *size* = [300, 300]);



Time for a quick sidebar on economics formulas and compounding periods. Our solution to this differential equation corresponds to continous compounding in the financial world. Let's get ahead of ourselves and make sure we get the same result as if you searched for continous compounding online which would be $P = P_0 \cdot e^{r \cdot t}$ where P_0 is the inital balance, r is the rate of interest and t is time. In our case $r = 0.18$ and $t = 4$.

B_solution := *dsolve*({*B*(0) = 100, *l*(**1**)});

$$B\_solution := B(t) = 100\, e^{\frac{9t}{50}} \quad \textbf{(2)}$$

sol1 := *unapply*(*rhs*(*B_solution*), *t*) : *evalf*(*sol1*(4)); = 205.4433211

This result matches the result obtained from $P = P_0 \cdot e^{r \cdot t}$.

$$P = P_0 \, e^{r\,t} \qquad \textbf{(3)}$$

Now let's create a Maple-based Euler's Method solution for differential equations with initial conditions. Euler is built into Maple and other Computer Algebra Systems (CAS), but by creating our own we can better control and understand the calculations.

In Maple it will be convenient to define an Euler procedure. This procedure is called with some input information and buckets for the outputs data that can be plotted.

To get started, we need to define the interval that we are interested in. We will call these variables b and a, where $b > a$. The step size h needs to be calculted. We do that with the interval endpoints and the total number of steps desired.

Euler's method procedure

```
restart

Euler := proc( f, b, a, n_steps, y0, X, Y)
local h, n :
 h := (b - a)/n_steps :

X[0] := a :
Y[0] := y0;  # inital condition

 # printf("  %3d  %18.12f  %18.12f\n", n, X[n], Y[n] ) :
 for n from 0 to (n_steps - 1)  do

 X[n + 1] := a + (n + 1) · h :
 Y[n + 1] := Y[n] + h · f(X[n], Y[n]) :

 # printf("  %3d  %18.12f  %18.12f\n", n + 1, X[n + 1], Y[n + 1] ) :

 end do:
 end proc:
```

Everything to the right of $\frac{d}{dt}(Y)$ defines the function (slope) we need to calculate. Define the function here

$f := (t, B) \rightarrow evalf(0.18 \cdot B);$

$$f := (t, B) \mapsto evalf(0.18 \cdot B) \tag{4}$$

$Digits = 16:$

Initial conditions

$a := 0 : b := 5 : y0 := 100 :$

$n\_steps := 10 :$ # *number of steps*

Places to put x and y from calling Euler

$X := Array(0 .. n\_steps) :$

$Y := Array(0 .. n\_steps) :$

Here is where we call Euler's method. The results are returned in the X and Y vectors. Following that call we define the exact solution. We can then compare our Euler results with the exact solution. In real life you may not have an exact solution. The maximum error between the Euler result and the exact solution is determined. Both solutions are then plotted.

$Euler(f, b, a, n\_steps, y0, X, Y) :$

$exacty := t \rightarrow 100 \cdot e^{0.18 \cdot t};$

$$exacty := t \mapsto 100 \cdot e^{0.18 \cdot t} \tag{5}$$

$maxerr := 0 :$

$h := \frac{(b - a)}{n\_steps};$

$$h := \frac{1}{2} \tag{6}$$

for n **from** 0 **to** $(n\_steps - 1)$ **do**

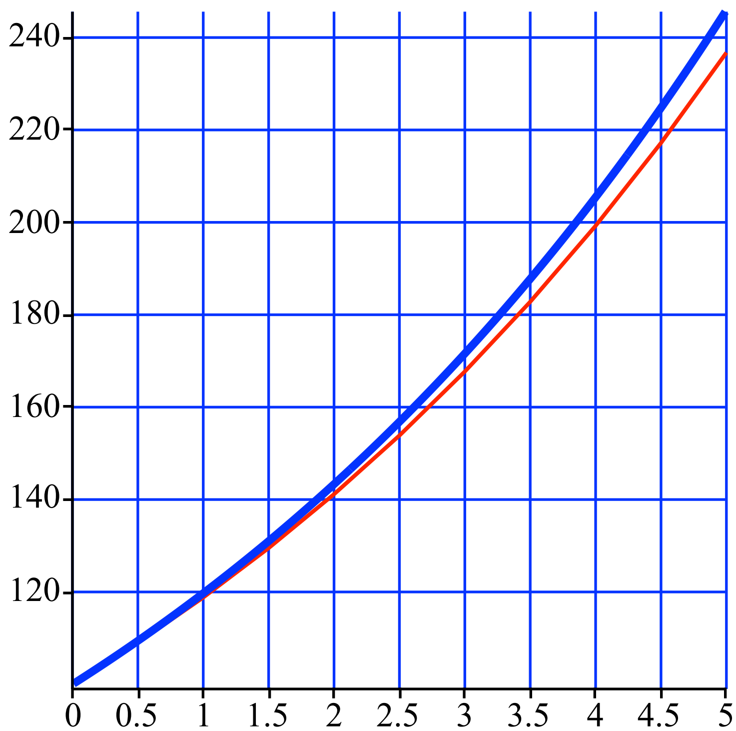
$maxerr := \max(maxerr, \text{abs}(exacty(n \cdot h) - Y[n]));$

end do:

$evalf(maxerr)$

$$7.6014707 \tag{7}$$

```
with( plots ) :
plot1 := plot( exacty, a ..b, size = [300, 300], axis = [gridlines = [10, color = blue]], color = blue,
    thickness = 3 ) :
plot2 := plot( X, Y, color = red ) :
plots:-display( {plot1, plot2}, size = [300, 300], axis = [gridlines = [10, color = blue]] );
```



The exact solution is the blue line. With 10 steps the Euler curve is not too bad and the error close to 4% at $t = 4$. Let's run it again with more steps.

```
n_steps := 100 : # number of steps
X1 := Array( 0 ..n_steps ) :
Y1 := Array( 0 ..n_steps ) :
Euler( f, b, a, n_steps, y0, X1, Y1 ) :
maxerr := 0 :
```

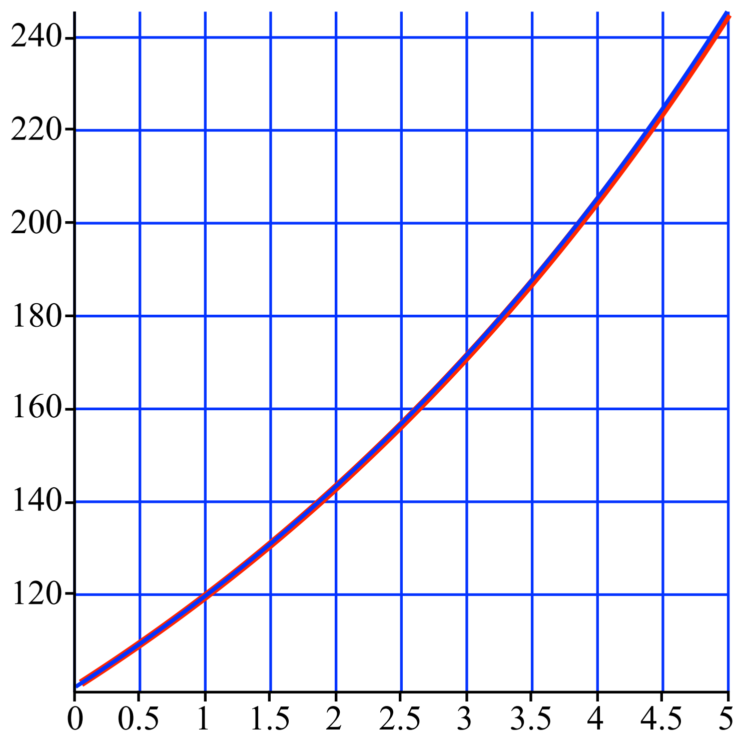
$h := \frac{(b - a)}{n\_steps};$

$$h := \frac{1}{20} \qquad \textbf{(8)}$$

for n **from** 0 **to** $(n\_steps - 1)$ **do**
$maxerr := \max(maxerr, \text{abs}(exacty(n \cdot h) \ - Y1[n]) \ );$
end do:
$evalf(maxerr)$

$$0.9695839 \tag{9}$$

$with(plots)$:
$plot1 := plot(exacty, a\,..b, size = [300, 300], axis = [gridlines = [10, color = blue]], color = blue, thickness = 1)$:
$plot2 := plot(X1, Y1, color = red, thickness = 3)$:
$plots\text{:-}display(\{plot2, plot1\}, size = [300, 300], axis = [gridlines = [10, color = blue]])$;



Now the error is much less with the increased number of steps in the same interval. In real life, the differential equation might stiff in the interval of interest. That means for a small increment in t there is a large jump in the y direction.

Improved Euler methods have been developed to help with numerical stability and convergence, especially for stiff systems. There are many other numerical methods such as the Runge-Kutta 4th order algorithm. As its name implies it uses 4 calculations to come up with each estimate.

One last thing. We have shown that our money will double in approximately 4 years. If you divide 72 the interest rate in percent, you have the rule of 72. In our case, $\frac{72}{18} = 4$ years. This rule of 72 comes in handy when evaluated investment performance.