Euler's Method for Approximation of Ordinary Differential Equations

Alexander MacKenzie

6 December 2018

1 Introduction

When approaching a first order ordinary differential equation, it may be necessary to approximate the solution rather than solve through integration. This can be due to a number of reasons such as the equation is too difficult or complex to find a solution, or perhaps there simply does not exist a solution. Leonhard Euler, a Swiss mathematician, devised a simple solution known in numerical analysis as Euler's Method: $w_0 = y_0, w_{i+1} = w_i + hf(t_i, w_i)$ [1]. This method works by splitting an interval of time (denoted as t) into equal pieces, or steps (denoted as h), and evaluating the differential equation according to the steps, checking the slope after each step and moving according to the new slope [1].

2 Summary of Results

We will be looking at the first order differential equation $f(t, y) = ty + t^3$ which has an exact solution of $y(t) = 3e^{t^2/2} - t^2 - 2$. If we run Euler's method with 5 steps on this equation with an initial value of $y = 1$, a time interval from $[0, 1]$ we can see in figure 1 and table 1 that quite a bit of error is introduced by even the second step. By the time $t = 1$ the approximation is off by an error of 0.3155. This is because Euler's method is first order convergent, indicating that the smaller the step size the closer the approximation should be since we are evaluating the slope at each step. Increasing the number of steps to 10, shown in figure 1 and table 2, gives us an approximation that is twice as accurate while taking twice as long to calculate. Increasing the number of steps to 100, shown in figure 3 and the abridged table 3, gives us a more accurate approximation of the differential equation, accurate to a tolerance of 10^-2 error.

3 Discussion

Euler's Method for approximating first order ordinary differential equations is useful but it takes a very long time to reach any true semblance of accuracy. If

the problem that you are faced with requires an accurate approximation, then Euler's Method is not likely to be the best option. Although it is computationally simple, it requires far too many steps to reach an accurate approximation being first order convergent, and is therefore quite unwieldy; especially if being computed by hand. There are much faster methods, such as Heun's Method (also known as the explicit Trapezoid Method), and the The Runge-Kutta family of methods [1]. Heun's Method greatly increases the accuracy to a second order convergence by making use of trapezoids in the direction of the slope to minimize error [1]. The Runge-Kutta order four, which further improves on Heun's Method, is fourth order convergent increasing the accuracy with far few steps needed [1].

Depending on how accurate of a result is needed, Heun's Method or one of the Runge-Kutta methods may be a better choice due to the speed at which they converge. However, since both are derived from Euler's method [1], it remains an incredibly important formula that can be used to roughly approximate first order ordinary differential equations with relative ease.

References

[1] Timothy Sauer. *Numerical Analysis.* Always learning. Pearson, 2012.

4 Appendix

```
#Euler's Method for Initial Value Problems
#@Alexander MacKenzie

#to use:
#       edit functions FofX and FPrimeofX to match your problem
#       run the whole script
#For results with comparison if F(x) can be found
#       execute EulersMethodCompare(initial value, number of steps,
                                    lower t bound, upper t bound)
#For results with approximation only
#       execute EulersMethod(initial value, number of steps,
                             lower t bound, upper t bound)


#setting functions
FPrimeofX = function(t, y){
        t*y+t^3
}
FofX = function(t){
        (3*exp(1)^((t^2)/2))-(t^2)-2
}


#Euler's Method
EulerStep = function(t,w,h){
        w+h*FPrimeofX(t,w)       #wi+1=wi+hf(ti,wi)
}
EulersMethodCompare = function(y,n,inter1,inter2) {
        t=c()
        w=c(y)  #w0=y0
```

```
        y=c()
        error=c()
        h=(inter2-inter1)/n      #finding step size

        for(i in 0:n) t=c(t,h*i)         #setting t to a vector
        for(i in 1:n){
                w=c(w,EulerStep(t[i],w[i],h))    #Euler Step

        }
        for(i in 0:n+1){
                y=c(y,FofX(t[i]))        #setting y to a vector
                error=c(error,abs(y[i]-w[i]))    #setting error
                                                   to a vector
        }

        plot(t, y, main="Euler's Method", ylab="y", xlab="t",
             col="red", type="l")        #graph
        points(t, w, col="blue")
        legend("topleft", legend=c("y Exact", "y' Approximated"),
             col=c("red", "blue"), lty=1:3, cex=1)

        table <- data.frame(t,w,y,error)         #table
        print(table)
}
EulersMethod = function(y,n,inter1,inter2) {
        t=c()
        w=c(y)  #w0=y0

        h=(inter2-inter1)/n      #finding step size

        for(i in 0:n) t=c(t,h*i)         #setting t to a vector
        for(i in 1:n){
                w=c(w,EulerStep(t[i],w[i],h))    #Euler Step

        }

        plot(t, w, main="Euler's Method", ylab="y", xlab="t",
               col="blue", type="p")     #graph
        legend("topleft", legend=c("y' Approximated"),
                     col=c("blue"), lty=3, cex=1)

        table <- data.frame(t,w)         #table
        print(table)
}
```

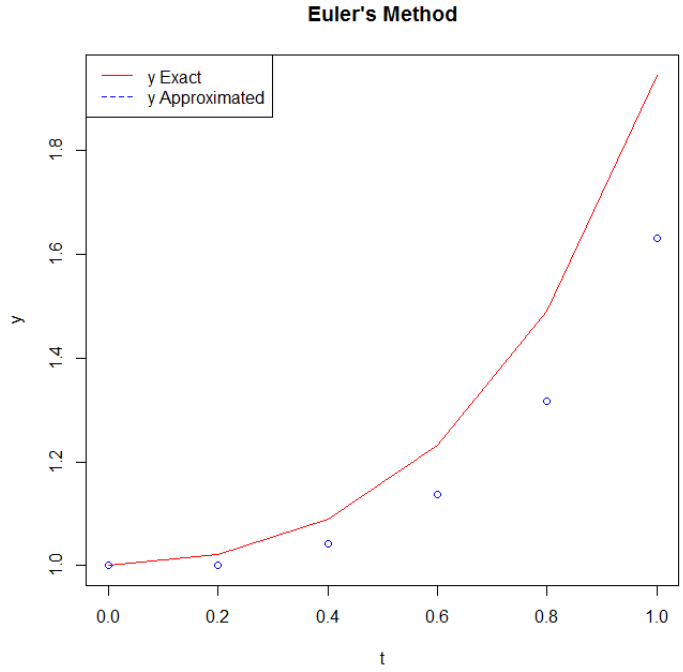

Figure 1: Plot of Euler's approximation using y' versus the exact value of y where $h = 0.2$

	t	w	y	error
1	0.0	1.000000	1.000000	0.00000000
2	0.2	1.000000	1.020604	0.02060402
3	0.4	1.041600	1.089861	0.04826120
4	0.6	1.137728	1.231652	0.09392409
5	0.8	1.317455	1.491383	0.17392793
6	1.0	1.630648	1.946164	0.31551559

Table 1: Table showing $h = 0.2$ as a step size

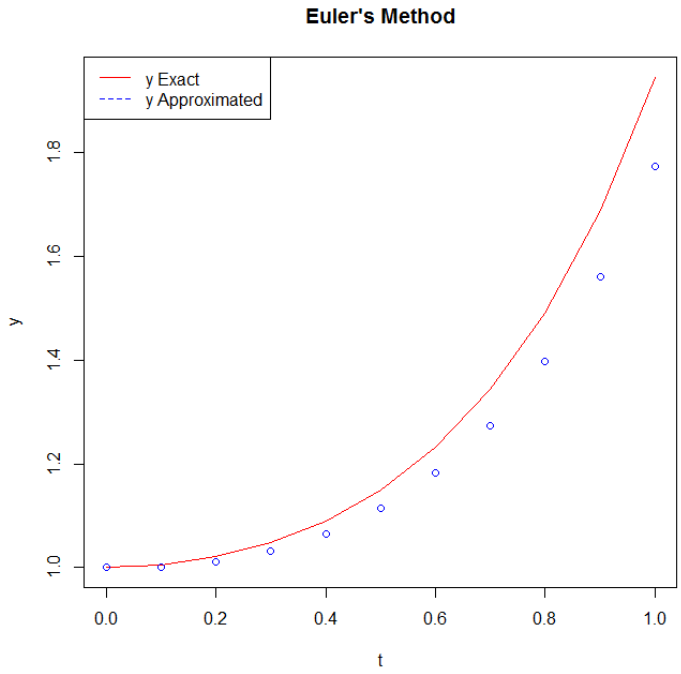

Figure 2: Plot of Euler's approximation using y' versus the exact value of y where $h = 0.1$

	t	w	y	error
1	0.0	1.000000	1.000000	0.000000000
2	0.1	1.000000	1.005038	0.005037563
3	0.2	1.010100	1.020604	0.010504020
4	0.3	1.031102	1.048084	0.016981580
5	0.4	1.064735	1.089861	0.025126143
6	0.5	1.113724	1.149445	0.035720897
7	0.6	1.181911	1.231652	0.049741404
8	0.7	1.274425	1.342864	0.068438613
9	0.8	1.397935	1.491383	0.093448193
10	0.9	1.560970	1.687908	0.126937593
11	1.0	1.774357	1.946164	0.171806613

Table 2: Table showing $h = 0.1$ as a step size

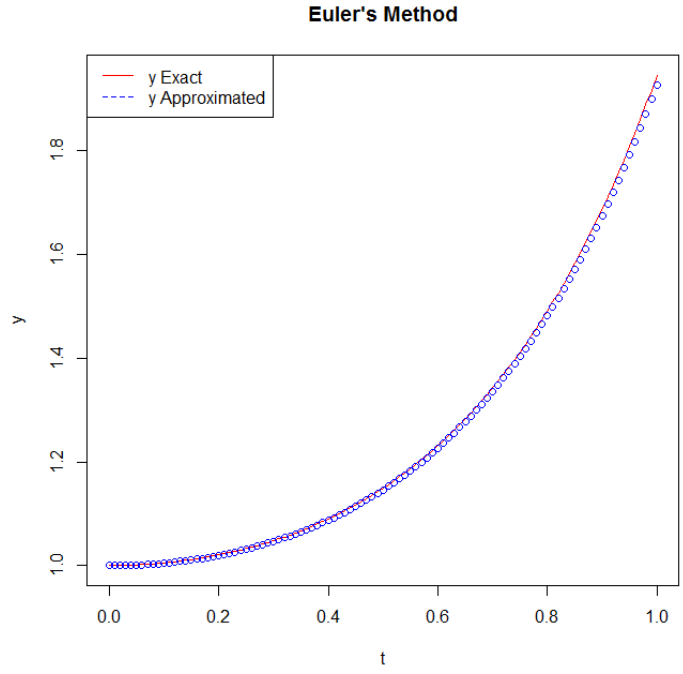

Figure 3: Plot of Euler's approximation using y' versus the exact value of y where $h = 0.01$

	t	w	y	error
1	0.00	1.000000	1.000000	0.000000e+00
2	0.01	1.000000	1.000050	5.000375e-05
3	0.02	1.000100	1.000200	1.000500e-04
4	0.03	1.000300	1.000450	1.501938e-04
5	0.04	1.000600	1.000801	2.004902e-04
...	...	...	...	...
96	0.95	1.792312	1.808321	1.600974e-02
97	0.96	1.817912	1.834425	1.651288e-02
98	0.97	1.844212	1.861243	1.703136e-02
99	0.98	1.871227	1.888793	1.756567e-02
100	0.99	1.898977	1.917094	1.811629e-02
101	1.00	1.927480	1.946164	1.868372e-02

Table 3: Table showing $h = 0.01$ as a step size