

Math 2280 - Project 1 Writeup

Dylan Zwick

Spring 2009

This is my sample writeup for our first class project. Your writeup obviously doesn't need to be exactly like this, but it should contain more or less the same information.

1 Euler's Method

For the first part of our project I wrote an implementation of Euler's method in Maple (using code provided for you in the handout) and then used this implementation to estimate the values of the "famous" numbers e , π , and $\ln(2)$ by recognizing them as values of the solutions to certain differential equations at given points. Each time the total x distance from the initial x value to the estimated x value was 1, and I began by taking 50 steps, and then doubling the step size for each iteration until the estimate settled down. For the Euler's method estimates settling down meant no change in the first three decimal places.

The code used to implement Euler's method (given in the handout) is:

```
EulersMethod := proc(x0,y0,h,n) local xk, yk, k;  
  xk := x0;  
  yk := y0;  
  k := 0;  
  while k < n do  
    k := k+1;  
    yk := yk + h*f(xk,yk);  
    xk := xk + h;
```

```
end do ;
[ xk , yk ] ;
end ;
```

I first used this code to estimate the value of e as being the solution to the initial value problem:

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

$$y(0) = 1$$

at the point $x = 1$.

I first set the function f to be $f(x, y) = y$ by entering in the command:

```
f := (x, y) -> y;
```

then using these initial conditions I recorded the values:

Command	h	n	Value
<i>EulersMethod</i> (0, 1, .02, 50)	.02	50	2.692
<i>EulersMethod</i> (0, 1, .01, 100)	.01	100	2.705
<i>EulersMethod</i> (0, 1, .005, 200)	.005	200	2.712
<i>EulersMethod</i> (0, 1, .0025, 400)	.0025	400	2.715
<i>EulersMethod</i> (0, 1, .00125, 800)	.00125	800	2.717
<i>EulersMethod</i> (0, 1, .000625, 1, 600)	.000625	1,600	2.717

So, my estimate using Euler's method for the value of e out to three decimal places is 2.717. The actual value out to three decimal places is 2.718.

The next famous value I estimated was $\ln(2)$, which is the solution to the initial value problem:

$$\frac{dy}{dx} = \frac{1}{x}$$

$$y(1) = 0$$

at the point $x = 2$.

I first set the function f to be $f(x, y) = \frac{1}{x}$ by entering in the command:

```
f := (x, y) -> 1/x;
```

then using these initial conditions I recorded the values:

Command	h	n	Value
<i>EulersMethod</i> (0, 1, .02, 50)	.02	50	0.698
<i>EulersMethod</i> (0, 1, .01, 100)	.01	100	0.696
<i>EulersMethod</i> (0, 1, .005, 200)	.005	200	0.694
<i>EulersMethod</i> (0, 1, .0025, 400)	.0025	400	0.694

So, my estimate using Euler's method for the value of $\ln(2)$ out to three decimal places is 0.694. The actual value out to three decimal places is 0.693.

The next famous value I estimated was π , which is the solution to the initial value problem:

$$\begin{aligned}\frac{dy}{dx} &= \frac{4}{1+x^2} \\ y(0) &= 0\end{aligned}$$

at the point $x = 1$.

I first set the function f to be $f(x, y) = \frac{4}{1+x^2}$ by entering in the command:

```
f := (x, y) -> 4/(1+x^2);
```

then using these initial conditions I recorded the values:

Command	h	n	Value
<i>EulersMethod</i> (0, 1, .02, 50)	.02	50	3.162
<i>EulersMethod</i> (0, 1, .01, 100)	.01	100	3.152
<i>EulersMethod</i> (0, 1, .005, 200)	.005	200	3.147
<i>EulersMethod</i> (0, 1, .0025, 400)	.0025	400	3.144
<i>EulersMethod</i> (0, 1, .00125, 800)	.00125	800	3.143
<i>EulersMethod</i> (0, 1, .000625, 1, 600)	.000625	1,600	3.142
<i>EulersMethod</i> (0, 1, .0003125, 3, 200)	.0003125	3,200	3.142

So, my estimate using Euler's method for the value of π out to three decimal places is 3.142. The actual value out to three decimal places is 3.142.

2 Improved Euler's Method

For the next part of the project I did the same thing, only instead of using Euler's method I used an implementation of the improved Euler's method, and instead of calculating out the estimate to three decimal places, I calculated the estimate out to five decimal places.

The code for implementing the improved Euler's method I used was:

```
ImprovedEulersMethod := proc(x0,y0,h,n)
local xk, yk, k, k1, uk, k2;
xk := x0;
yk := y0;
k := 0;
while k < n do
k := k+1;
k1 := f(xk,yk);
uk := uk + h*k1;
k2 := f(xk+h,uk);
yk := yk + h*(1/2)*(k1+k2);
xk := xk + h;
end do;
[xk,yk];
end;
```

Using this improved implementation, and the same methodology as I used with Euler's method, I found the following estimates for the value of e :

Command	h	n	Value
<i>ImprovedEulersMethod</i> (0, 1, .02, 50)	.02	50	2.71810
<i>ImprovedEulersMethod</i> (0, 1, .01, 100)	.01	100	2.71824
<i>ImprovedEulersMethod</i> (0, 1, .005, 200)	.005	200	2.71827
<i>ImprovedEulersMethod</i> (0, 1, .0025, 400)	.0025	400	2.71828
<i>ImprovedEulersMethod</i> (0, 1, .00125, 800)	.00125	800	2.71828

So, our estimate for e to five decimal places is 2.71828. The actual value to five decimal places is 2.71828. Nice!

Estimating $\ln(2)$ using the improved Euler's method I obtained the following:

Command	h	n	Value
<i>ImprovedEulersMethod</i> (0, 1, .02, 50)	.02	50	0.69317
<i>ImprovedEulersMethod</i> (0, 1, .01, 100)	.01	100	0.69315
<i>ImprovedEulersMethod</i> (0, 1, .005, 200)	.005	200	0.69315

Here the method converged rather quickly to the estimate of 0.69315 for the value of $\ln(2)$. The actual value to five decimal places is 0.69315. Sweet!

Estimating π using the improved Euler's method I obtained the following:

Command	h	n	Value
<i>ImprovedEulersMethod</i> (0, 1, .02, 50)	.02	50	3.14152
<i>ImprovedEulersMethod</i> (0, 1, .01, 100)	.01	100	3.14158
<i>ImprovedEulersMethod</i> (0, 1, .005, 200)	.005	200	3.14159
<i>ImprovedEulersMethod</i> (0, 1, .0025, 400)	.0025	400	3.14159

So, my estimate for π out to five decimal places is 3.14159. The actual value to five decimal places is 3.14159. Excellent!

3 The Runge-Kutta Method

Finally, I wrote an implementation of the Runge-Kutta method, and used this implementation to estimate the values of our three famous numbers out to *nine* decimal places.

The code for my implementation of the Runge-Kutta method is:

```
RungeKuttaMethod := proc(x0,y0,h,n)
local xk, yk, k, k1, k2, k3, k4;
xk := x0;
yk := y0;
k := 0;
while k < n do
k := k+1;
k1 := f(xk,yk);
k2 := f(xk+(1/2)*h,yk+(1/2)*h*k1);
k3 := f(xk+(1/2)*h,yk+(1/2)*h*k2);
k4 := f(xk+h,yk+h*k3);
yk := yk + (h/6)*(k1+2*k2+2*k3+k4);
xk := xk + h;
end do;
[xk,yk];
end;
```

Using this implementation and our previous methodology, I obtained the following estimate for the value of e :

Command	h	n	Value
<i>RungeKuttaMethod</i> (0, 1, .02, 50)	.02	50	2.718281822
<i>RungeKuttaMethod</i> (0, 1, .01, 100)	.01	100	2.718281831
<i>RungeKuttaMethod</i> (0, 1, .005, 200)	.005	200	2.718281827
<i>RungeKuttaMethod</i> (0, 1, .0025, 400)	.0025	400	2.718281810
<i>RungeKuttaMethod</i> (0, 1, .00125, 800)	.00125	800	2.7182823
<i>RungeKuttaMethod</i> (0, 1, .000625, 1600)	.000625	1,600	2.718281799

OK, what's going on here? My values aren't settling down, and they're even getting farther apart the more steps I take. Well, what's happening here is that I've got a problem with rounding error, and that doesn't

go away with more steps. It looks like, with the rounding limitations of Maple, the best estimate for the value of e I can get with this approach is out to 8 decimal places. This estimate is 2.71828182, which is correct. The value of e to nine decimal places is 2.718281728.

For my estimate of the value of $\ln(2)$ to nine decimal places I obtained the following:

Command	h	n	Value
<i>RungeKuttaMethod</i> (0, 1, .02, 50)	.02	50	0.693147181
<i>RungeKuttaMethod</i> (0, 1, .01, 100)	.01	100	0.693147180
<i>RungeKuttaMethod</i> (0, 1, .005, 200)	.005	200	0.693147181
<i>RungeKuttaMethod</i> (0, 1, .0025, 400)	.0025	400	0.693147181

So, this time the values settled down, and my estimate for $\ln(2)$ out to nine decimal places is 0.693147181, which is correct. Hooray!

Finally, for my estimate of the value of π out to nine decimal places using the Runge-Kutta method I obtained the following:

Command	h	n	Value
<i>RungeKuttaMethod</i> (0, 1, .02, 50)	.02	50	3.141592656
<i>RungeKuttaMethod</i> (0, 1, .01, 100)	.01	100	3.141592650
<i>RungeKuttaMethod</i> (0, 1, .005, 200)	.005	200	3.141592658
<i>RungeKuttaMethod</i> (0, 1, .0025, 400)	.0025	400	3.141592657
<i>RungeKuttaMethod</i> (0, 1, .00125, 800)	.00125	800	3.141592654
<i>RungeKuttaMethod</i> (0, 1, .000625, 1600)	.000625	1,600	3.141592646

Again, we're running into round problems, and so it looks like the best estimate we can come up with using this approach is out to eight decimals, 3.14159265, which is correct. The value of π to nine decimals is 3.141592654.

So, I implemented Euler's method, the improved Euler's method, and the Runge-Kutta method in Maple and used these methods to calculate estimates of famous numbers out to three, five, and nine decimal places respectively. The nine decimal place estimates ran into problems with rounding error, but other than that the methods worked well, especially the improved Euler's method and the Runge-Kutta method.