University of Waterloo

AMATH 251

Numerical Approximations for Differential Equations

Nick Mitchell
David Yeghshatyan

K.G. Lamb

October 8, 2015

1 Quadratic Approximation

Numerical approximation of the equation $y' = xy$, $y(0) = 1$, with three different approximation methods.

Numerical Approximations			
Steps	Quadratic	RK2	Euler
1	1.5	1.5	1.0
2	1.58203	1.59961	1.25
4	1.62617	1.63422	1.41943
8	1.64219	1.64477	1.52401
$e^{\frac{1}{2}}$	1.64872	1.64872	1.64872

We see here that the Quadratic approximation is far superior to Euler's method, yet RK2 is still a slightly better approximation.

2 Showing that RK2 is a second order method

$$RK2: \qquad y_{j+1} = y_j + f(x_j + \frac{h}{2}, y^*)h, \quad where \quad y^* = y_j + f(x_j, y_j)\frac{h}{2}$$

$$Consider: \qquad f(x_j + \frac{h}{2}, y^*) = f(x_j + \frac{h}{2}, y_j + f(x_j, y_j)\frac{h}{2}) \quad where \quad y\prime(x) = f(x, y)$$

$$Gives \qquad = f(x_j + \frac{h}{2}, y_j + y'\frac{h}{2})$$

$$= f(x_j + \frac{h}{2}, y_j(x_j) + y'(x_j)\frac{h}{2})$$

$$Recall\ Taylor's\ Theorem: \qquad T(x) = \sum_{n=0}^{\infty} \frac{T^{(n)}(x_0)}{n!}(x - x_0)^n$$

$$let \quad x = x_j + \frac{h}{2}; x_0 = x_j; \qquad T(x_j + \frac{h}{2}) = \sum_{n=0}^{\infty} \frac{T^{(n)}(x_j)}{n!}(x_j + \frac{h}{2} - x_0)^n$$

$$= \sum_{n=0}^{\infty} \frac{T^{(n)}(x_j)}{n!}(\frac{h}{2})^n$$

$$= \frac{T^{(0)}(x_j)}{0!}(\frac{h}{2})^0 + \frac{T^{(1)}(x_j)}{1!}(\frac{h}{2})^1 + \bigcirc(h^2)$$

$$= T(x_j) + T'(x_j)(\frac{h}{2}) + \bigcirc(h^2)$$

$$So \qquad y_j + y'\frac{h}{2} = y(x_j + \frac{h}{2}) + \bigcirc(h^2)$$

$$And \qquad f(x_j + \frac{h}{2}, y^*) = f(x_j + \frac{h}{2}, y_j + y'\frac{h}{2})$$

$$= f(x_j + \frac{h}{2}, y(x_j + \frac{h}{2}) + \bigcirc(h^2))$$

$$= y'(x_j + \frac{h}{2}) + \bigcirc(h^2)$$

$$It\ follows\ that \qquad y_{j+1} = y_j + f(x_j + \frac{h}{2}, y^*)h,$$

$$= y_j + [y'(x_j + \frac{h}{2}) + \bigcirc(h^2)]h$$

$$= y_j + y'(x_j + \frac{h}{2})h + \bigcirc(h^3)$$

Therefore, RK2 is of error $\bigcirc(h^3)$ at each step.
Hence RK2 is order 2.

3 Initial Value Problems

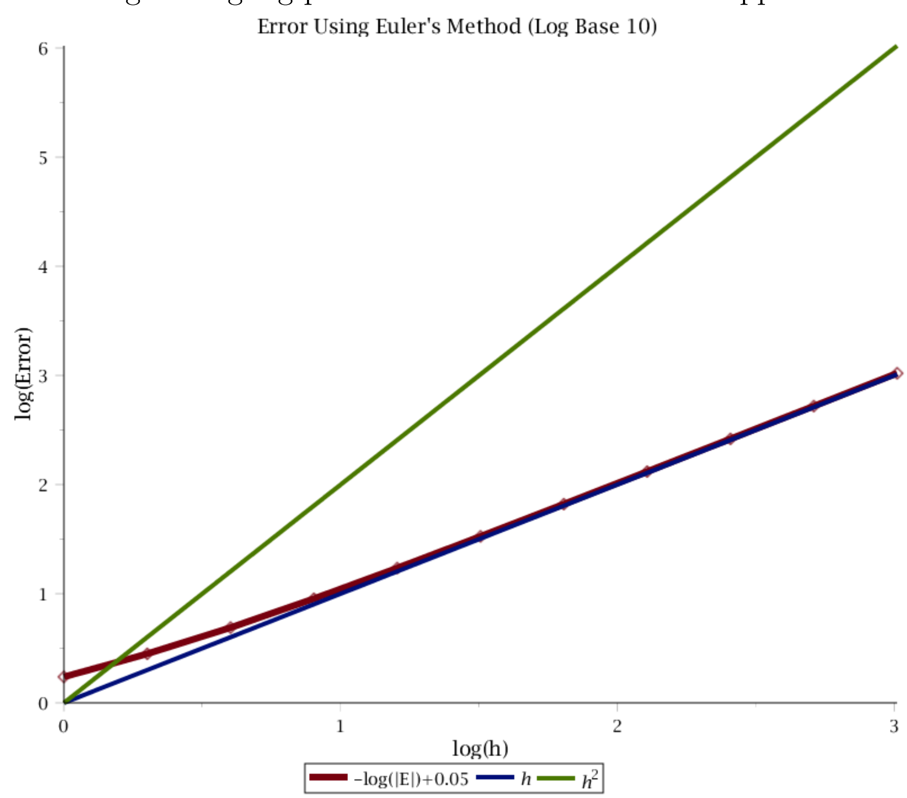
3.1 $y' = xy$, $y(0) = 1$

3.1.1 Analytic Solution

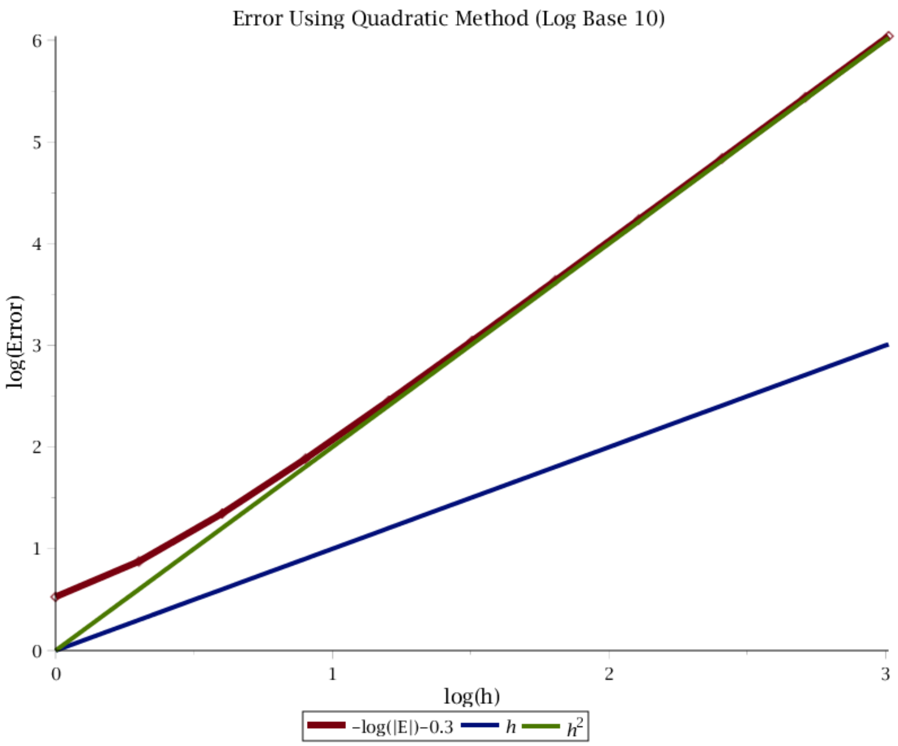
$$\begin{aligned}
y' &= \frac{dy}{dx} = xy \\
\frac{dy}{y} &= xdx \\
\int \frac{dy}{y} &= \int xdx \\
\ln(y) &= \frac{1}{2}x^2 + C \\
y &= e^{\frac{1}{2}x^2 + C} = e^{\frac{1}{2}x^2}e^{+C} \\
y(0) &= 1 \\
1 &= e^{\frac{1}{2}(0^2)}e^C = e^C \\
C &= 0 \\
y &= e^{\frac{1}{2}x^2}
\end{aligned}$$

3.1.2 Approximation Error Plots

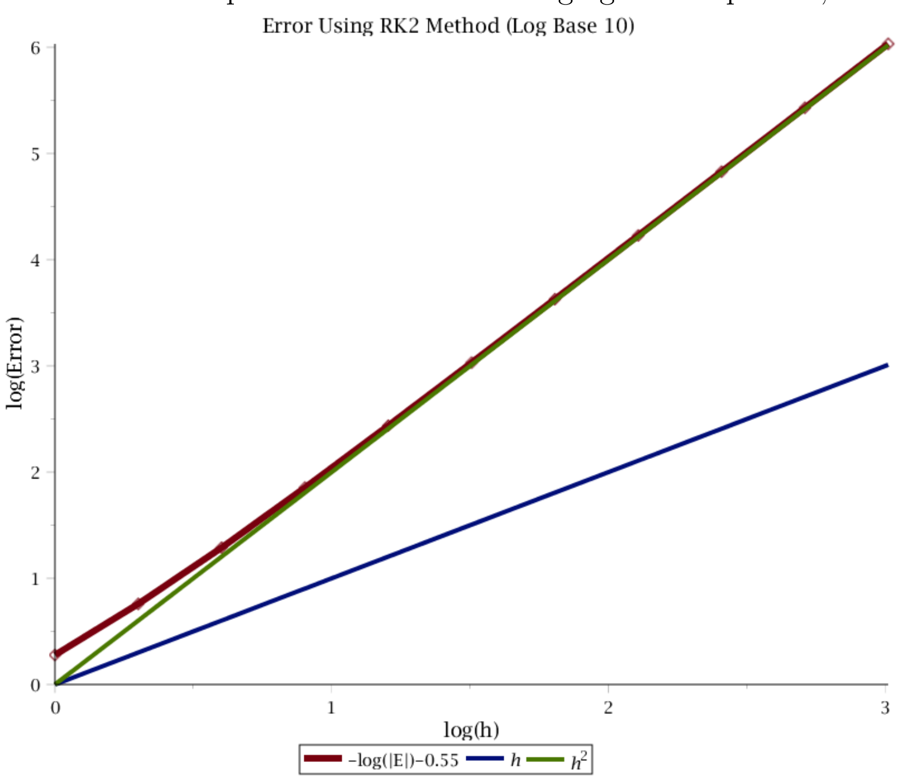
The following are log-log plots of the error of numerical approximations.



Here we see Euler's method converging to a slope of 1, thus the error is of order 1.



Here we see the quadratic method converging to a slope of 2, thus the error is of order 2.



Here we see the RK2 method converging to a slope of 2, thus the error is of order 2.

3.1.3 Alternative to RK2

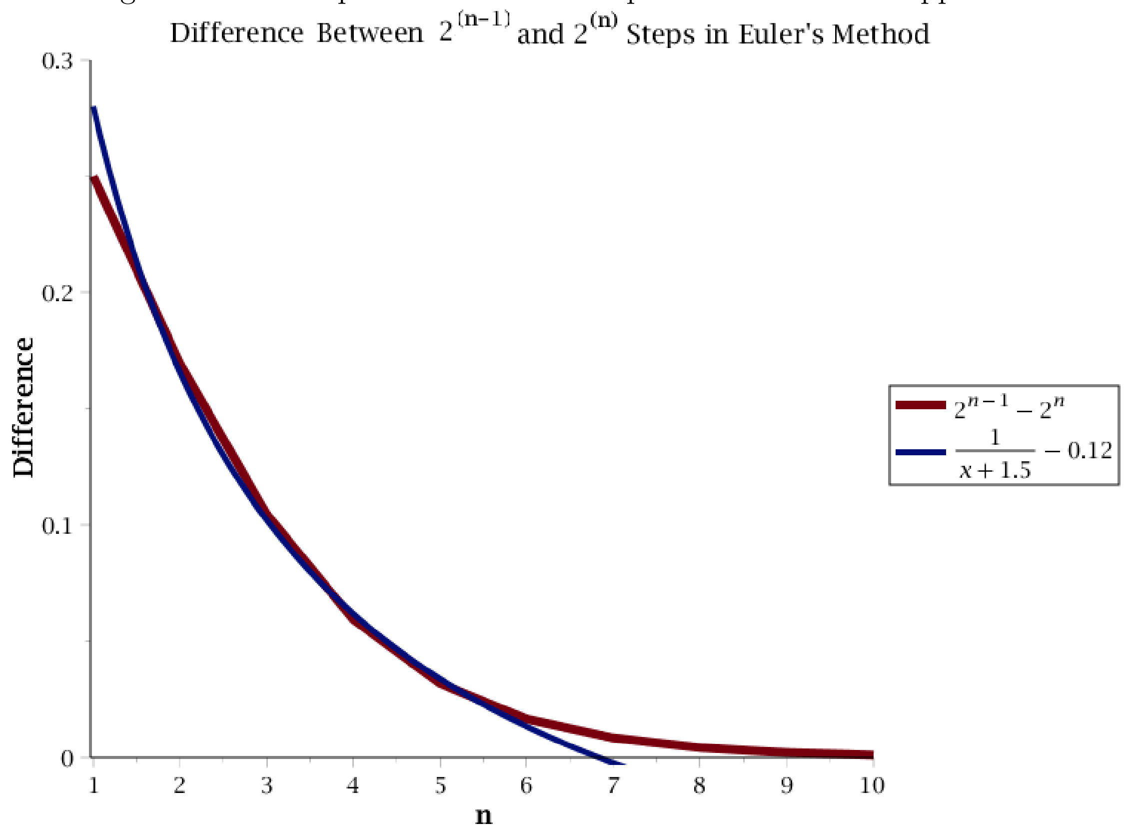
A modified version of RK2,
$y_j = y_{j-1} + \frac{h}{2} f(x_{j-1}, y_{j-1}) + \frac{h}{2} f(x_{j-1} + h, y_{j-1} + h[f(x_{j-1}, y_{j-1})])$
Produces different approximate values.

Numerical Approximations				
Steps	Alternate	RK2	Alt. Error	RK2 Error
1	1.5	1.5	0.14872	0.14871
2	1.61719	1.59961	0.03154	0.04911
4	1.64229	1.63422	0.00643	0.01450
8	1.64735	1.64477	0.00137	0.00103
16	1.64841	1.64769	0.00031	0.00395
32	1.64865	1.64846	0.00007	0.00026
$e^{\frac{1}{2}}$	1.64872	1.64872	0	0

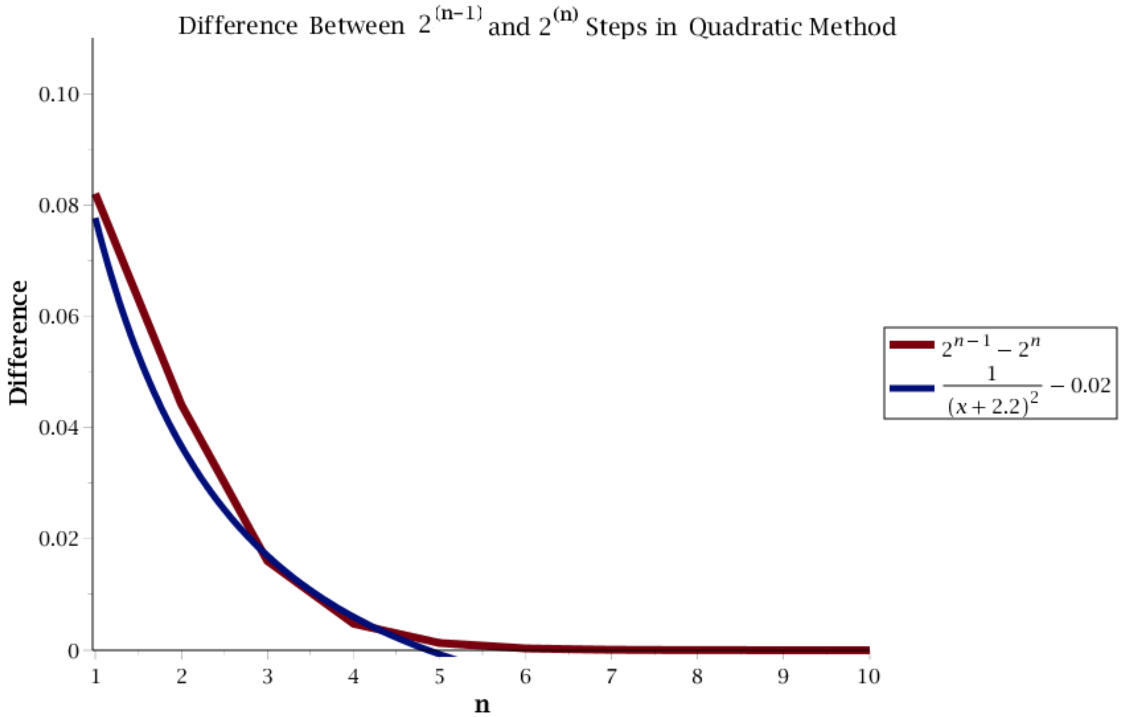
This method converges faster than RK2 for the given initial value problem.

3.1.4 Differences in Approximate Solutions

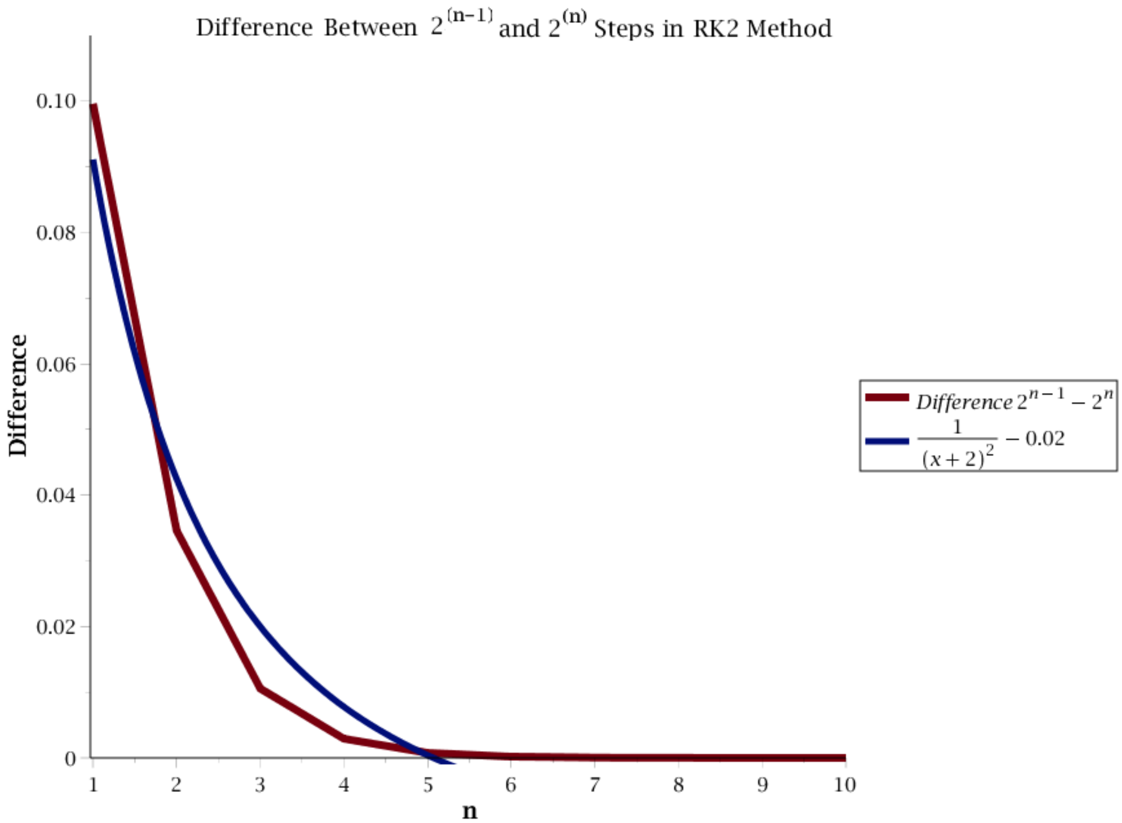
The following are difference plots between the step sizes in numerical approximations.



Here we see the differences in solutions for Euler's method tending to zero as a factor of x.



Here we see the differences in solutions for the quadratic method tending to zero as a factor of x^2.



Here we see the differences in solutions for RK2 tending to zero as a factor of x^2.

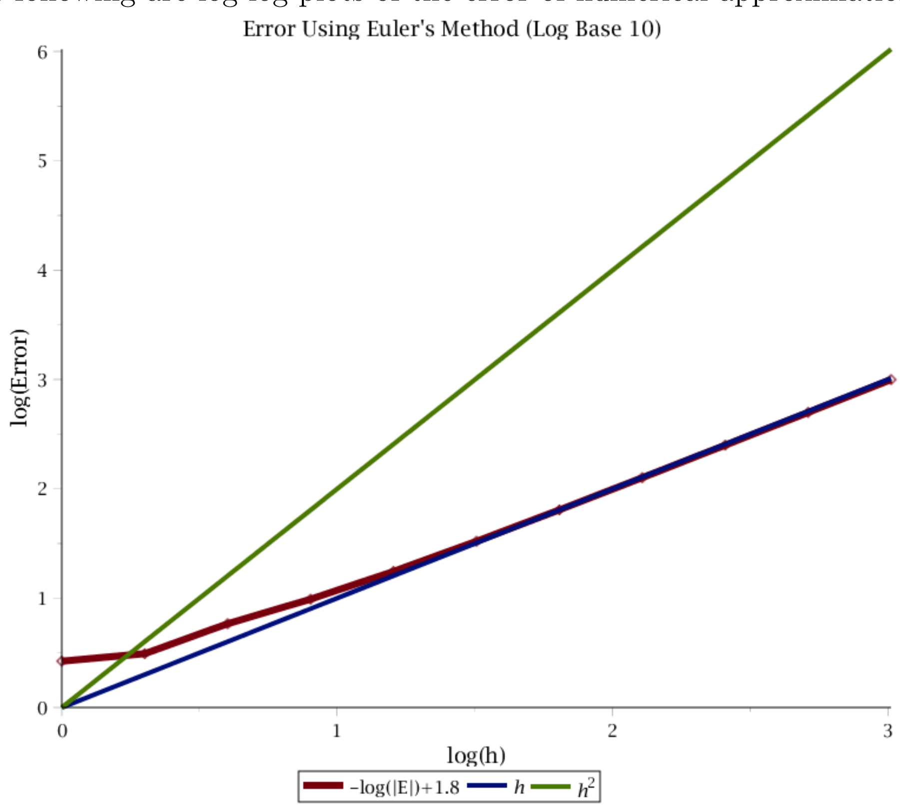
3.2 $y' = \frac{1}{2}y + 9\cos(3x) - \frac{3}{2}\sin(3x)$, $y(0) = 1$

3.2.1 Analytic Solution

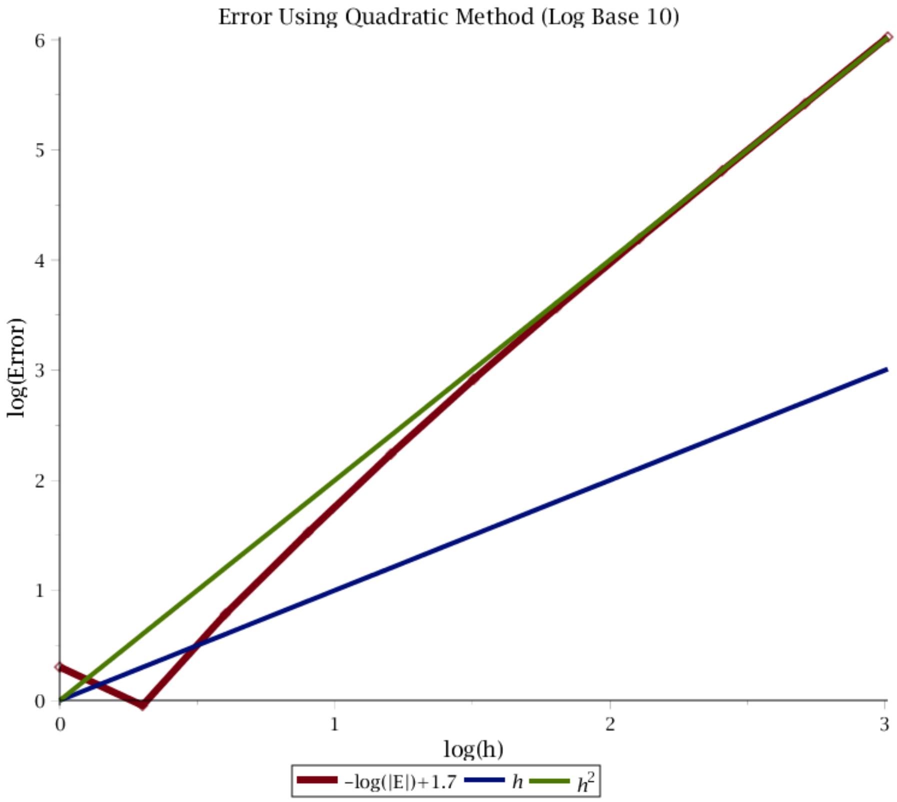
$$\begin{aligned}
y' &= \frac{1}{2}y + 9\cos(3x) - \frac{3}{2}\sin(3x) \\
y' - \frac{1}{2}y &= 9\cos(3x) - \frac{3}{2}\sin(3x) \quad ,let \quad u(x) = e^{\frac{-1}{2}x} \\
y'e^{\frac{-1}{2}x} - \frac{1}{2}ye^{\frac{-1}{2}x} &= [(9\cos(3x) - \frac{3}{2}\sin(3x)]e^{\frac{-1}{2}x} \\
y'u - yu' &= [(9\cos(3x) - \frac{3}{2}\sin(3x)]e^{\frac{-1}{2}x} \\
\frac{d}{dx}uy &= [(9\cos(3x) - \frac{3}{2}\sin(3x)]e^{\frac{-1}{2}x} \\
uy &= \int [(9\cos(3x) - \frac{3}{2}\sin(3x)]e^{\frac{-1}{2}x}dx \\
e^{\frac{-1}{2}x}y &= 3\sin(3x)e^{\frac{-1}{2}x} + C \\
y &= 3\sin(3x) + Ce^{\frac{1}{2}x} \quad , \quad y(0) = 1 \\
1 &= 3\sin(3(0)) + Ce^{\frac{1}{2}(0)} = C \\
y &= 3\sin(3x) + e^{\frac{1}{2}x}
\end{aligned}$$

3.2.2 Approximation Error Plots

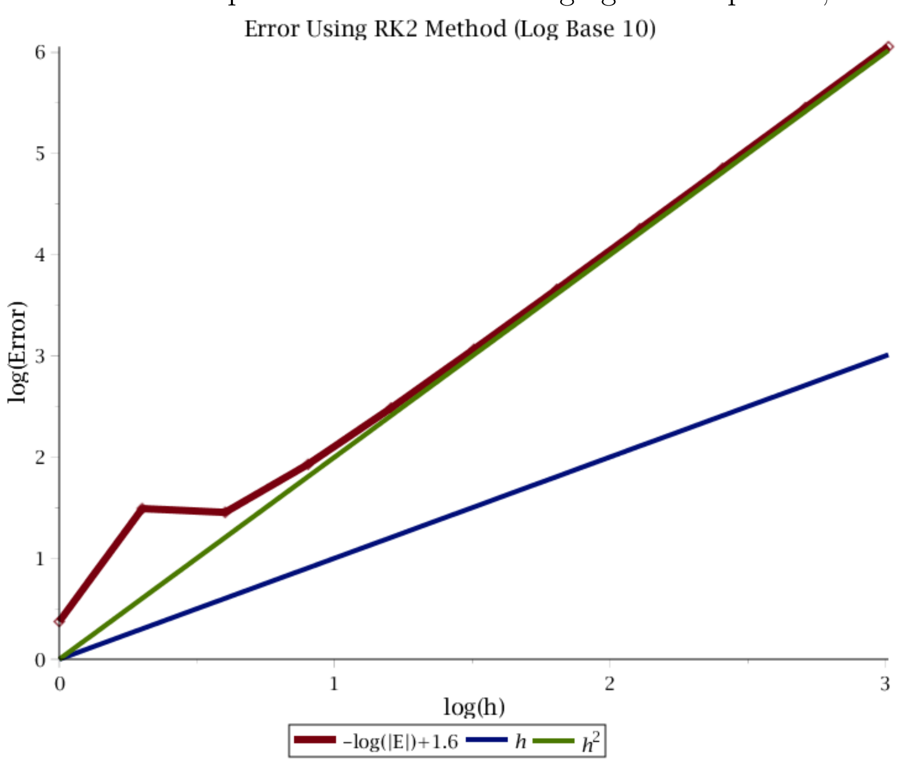
The following are log-log plots of the error of numerical approximations.



Here we see Euler's method converging to a slope of 1, thus the error is of order 1.



Here we see the quadratic method converging to a slope of 2, thus the error is of order 2.



Here we see the RK2 method converging to a slope of 2, thus the error is of order 2.

3.2.3 Alternative to RK2

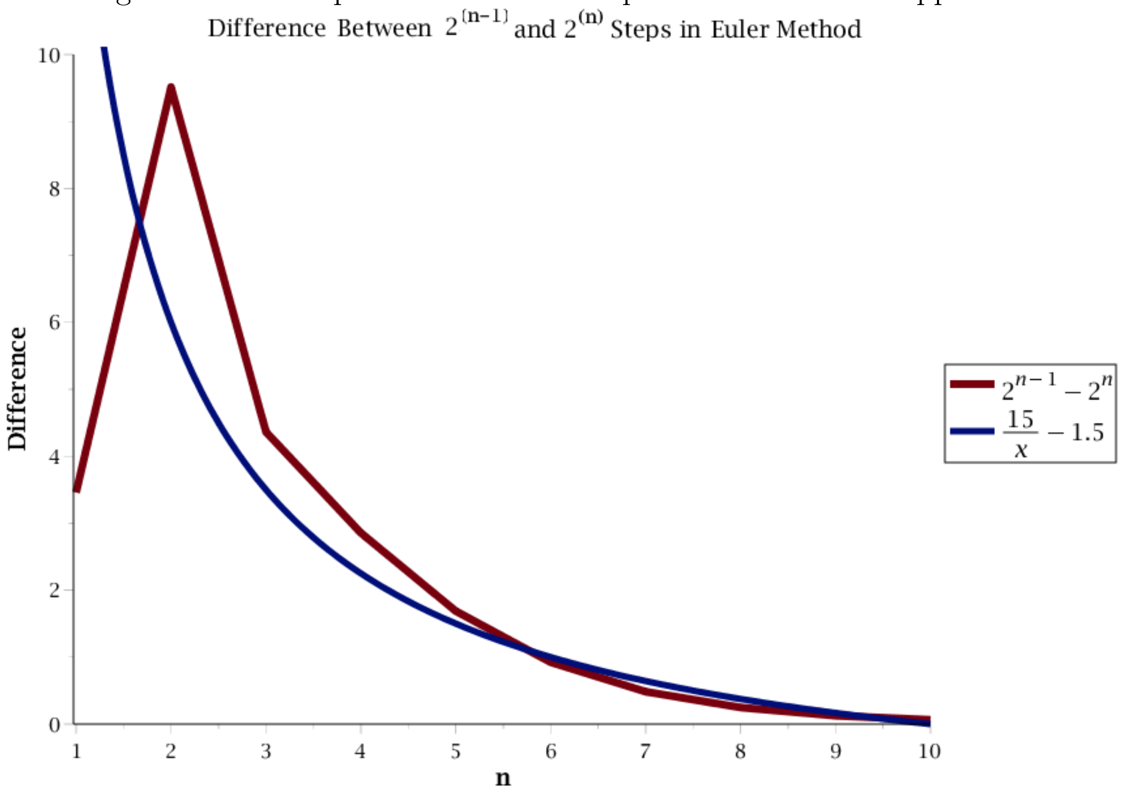
A modified version of RK2,
$y_j = y_{j-1} + \frac{h}{2} f(x_{j-1}, y_{j-1}) + \frac{h}{2} f(x_{j-1} + h, y_{j-1} + h[f(x_{j-1}, y_{j-1})])$
Produces different approximate values.

Numerical Approximations				
Steps	Alternate	RK2	Alt. Error	RK2 Error
1	24.14747	22.58240	18.42943	16.86435
2	20.28444	4.42807	14.56640	1.28998
4	7.12623	7.12280	1.40819	1.40476
8	6.05043	6.18877	0.33239	0.47073
16	5.80493	5.84905	0.08689	0.13100
32	5.74057	5.75235	0.02252	0.03430
Analytic Solution	5.71804	5.71804	0	0

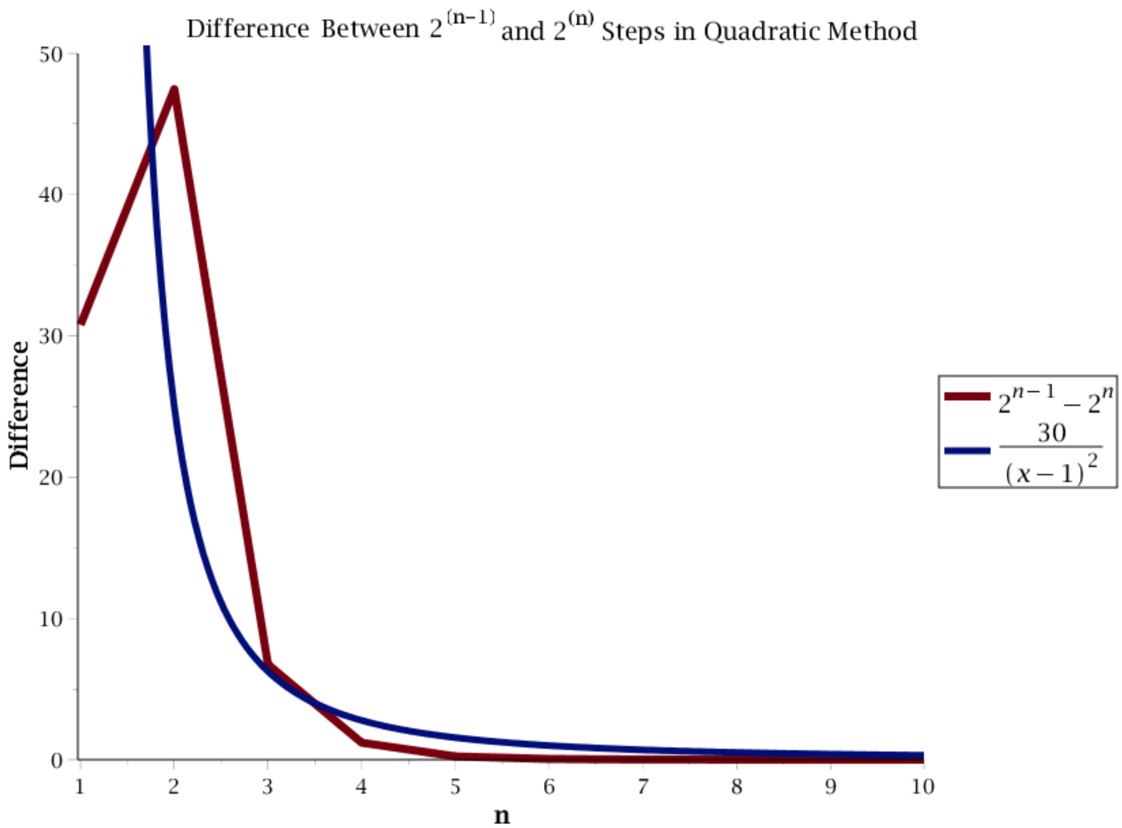
This method starts worse, but still converges faster than RK2 for the given initial value problem.

3.2.4 Differences in Approximate Solutions

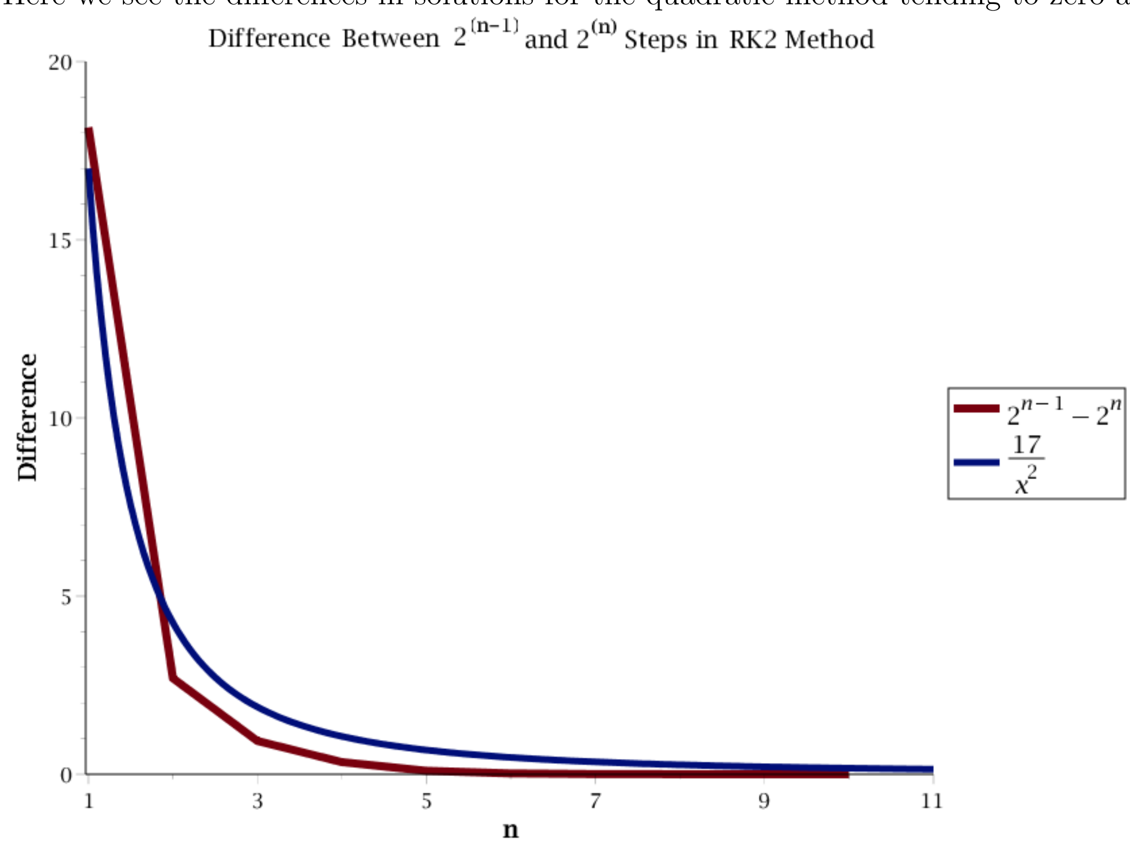
The following are difference plots between the step sizes in numerical approximations.



Here we see the differences in solutions for Euler's method tending to zero as a factor of x.



Here we see the differences in solutions for the quadratic method tending to zero as a factor of x^2.



Here we see the differences in solutions for RK2 tending to zero as a factor of x^2.

4 Appendix

```
#Python Code, Questions 1,3i)

#Given ODE
def f(x,y):
        return x*y

#Implicit derivative of the ODE, with respect to x
def Df(x,y):
        return y+(x*x*y)


# The following are all numerical approximations, utilizing loops
# s is the number of steps

def Alt(s):
        y=1
        x=0
        h=(1/s)
        n=0
        while n<s:
                y=y+(h/2)*(f(x,y)+f(x+h,y+h*f(x,y)))
                x=x+h
                n=n+1
        return y

def Euler(s):
        y=1
        x=0
        h=(1/s)
        n=0
        while n<s:
                y=y+(h*f(x,y))
                x=x+h
                n=n+1
        return y

def Quad(s):
        y=1
        x=0
        h=(1/s)
        n=0
        while n<s:
                y=y+(h*f(x,y))+((1/2)*(h*h)*Df(x,y))
                x=x+h
                n=n+1
        return y
```

```python
def RK2(s):
        y=1
        x=0
        h=(1/s)
        n=0
        while n<s:
                y=y+(h*f(x+(h/2),y+(h/2)*f(x,y)))
                x=x+h
                n=n+1
        return y

#Python Code, Question 3ii)

import math

#Given ODE
def f(x,y):
        return ((1/2)*y)+(9*math.cos(3*x))-((3/2)*math.sin(3*x))

#Implicit derivative of the ODE, with respect to x
def Df(x,y):
        return (1/2)*f(x,y)-(27*math.sin(3*x))-((9/2)*math.cos(3*x))

# The following are all numerical approximations, utilizing loops
# s is the number of steps

def Alt(s):
        y=1
        x=0
        h=(3/s)
        n=0
        while n<s:
                y=y+(h/2)*(f(x,y)+f(x+h,y+h*f(x,y)))
                x=x+h
                n=n+1
        return y

def Euler(s):
        y=1
        x=0
        h=(3/s)
        n=0
        while n<s:
                y=y+(h*f(x,y))
                x=x+h
                n=n+1
        return y
```

```python
def Quad(s):
        y=1
        x=0
        h=(3/s)
        n=0
        while n<s:
                y=y+(h*f(x,y))+((1/2)*(h*h)*Df(x,y))
                x=x+h
                n=n+1
        return y

def RK2(s):
        y=1
        x=0
        h=(3/s)
        n=0
        while n<s:
                y=y+(h*f(x+(h/2),y+(h/2)*f(x,y)))
                x=x+h
                n=n+1
        return y
```