MATH 475B

Homework 1

Q1:

Question 1

$\frac{dx}{dt} = x - \sin(t)$; $x(0)=0$ $\Longrightarrow$ $a(t)=1$, $b(t) = -\sin(t)$

$a(t) = 1 \Rightarrow A(t) = \int a(t)\,dt = t$

linear first order ODE

$$x(t) = K \cdot e^{A(t)} + e^{A(t)} \int^t e^{-A(s)} \cdot b(s) \cdot ds$$

$$= K \cdot e^t + e^t \left(\int^t -e^{-s} \sin(s)\, ds \right)$$

$$\int^t -e^{-s} \sin(s)\,ds = e^{-t}\sin(t) + \int e^{-s}\cos(s)\,ds$$

$$= e^{-t}\sin(t) + e^{-t}\cos(t) + \int^t e^{-s} \sin(s)\,ds$$

$$2\int^t e^{-s}\sin(s)\,ds = e^{-t}\sin(t) + e^{-t}\cos(t)$$

$$\Rightarrow \int^t -e^{-s}\sin(s)\,ds = \frac{\sin(t)+\cos(t)}{2e^t}$$

$$x(t) = K \cdot e^t + \cancel{e^t}\left(\frac{\sin(t)+\cos(t)}{2\cancel{e^t}}\right) = Ke^t + \frac{\sin(t)+\cos(t)}{2}$$

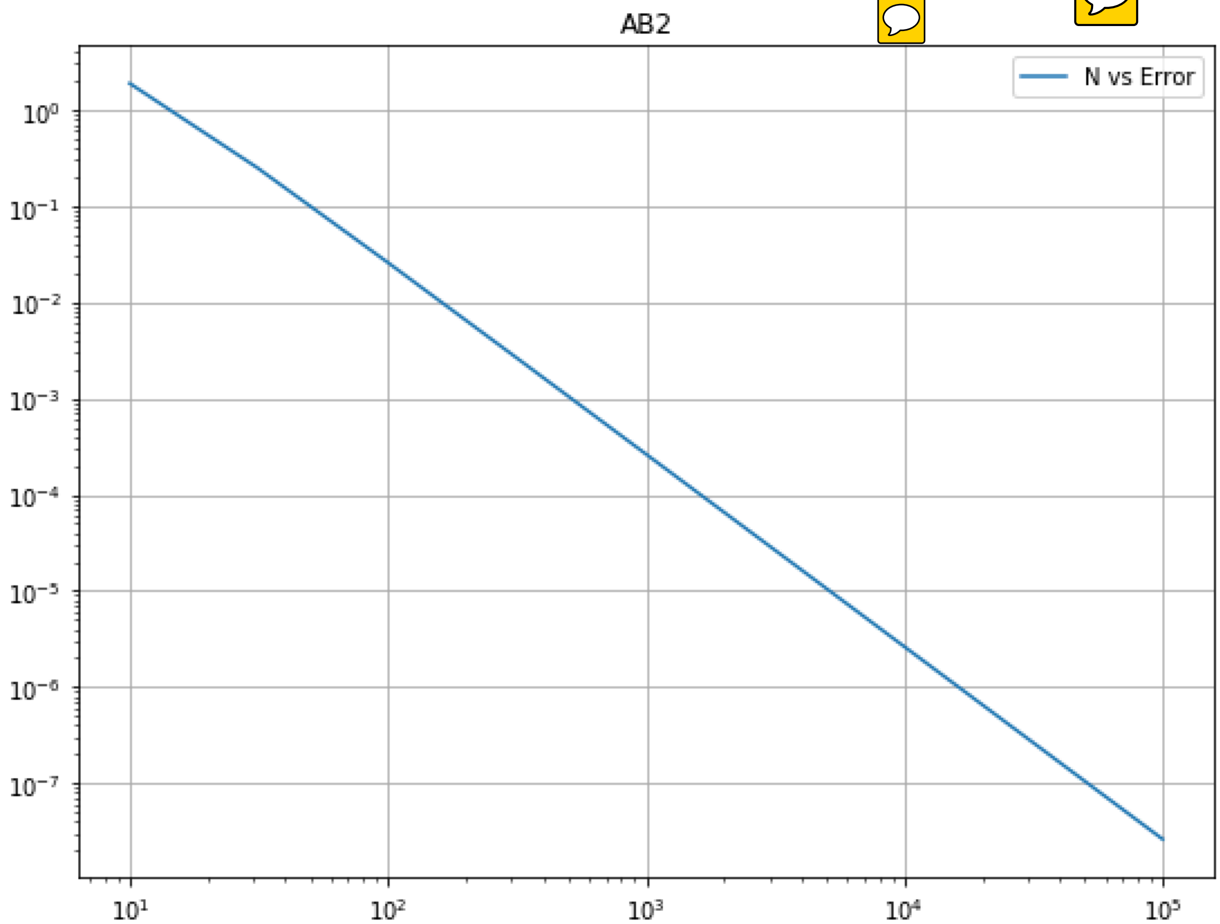
Finding K: $x(0)=0$

$$0 = Ke^0 + \frac{\sin(0)+\cos(0)}{2} = \frac{1}{2}$$

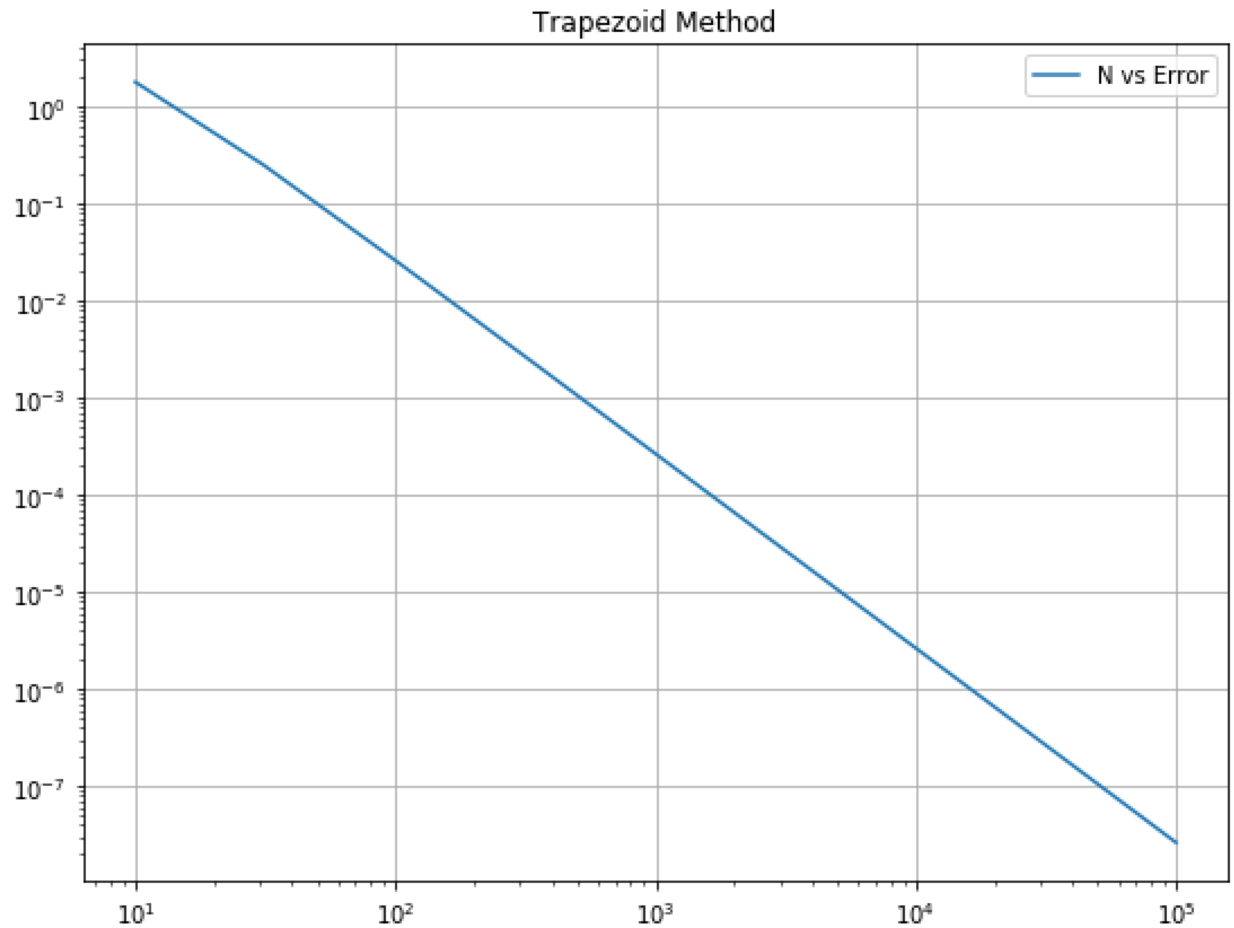
$K = -\frac{1}{2}$

$$\boxed{x(t) = \frac{-e^t + \sin(t) + \cos(t)}{2}}$$

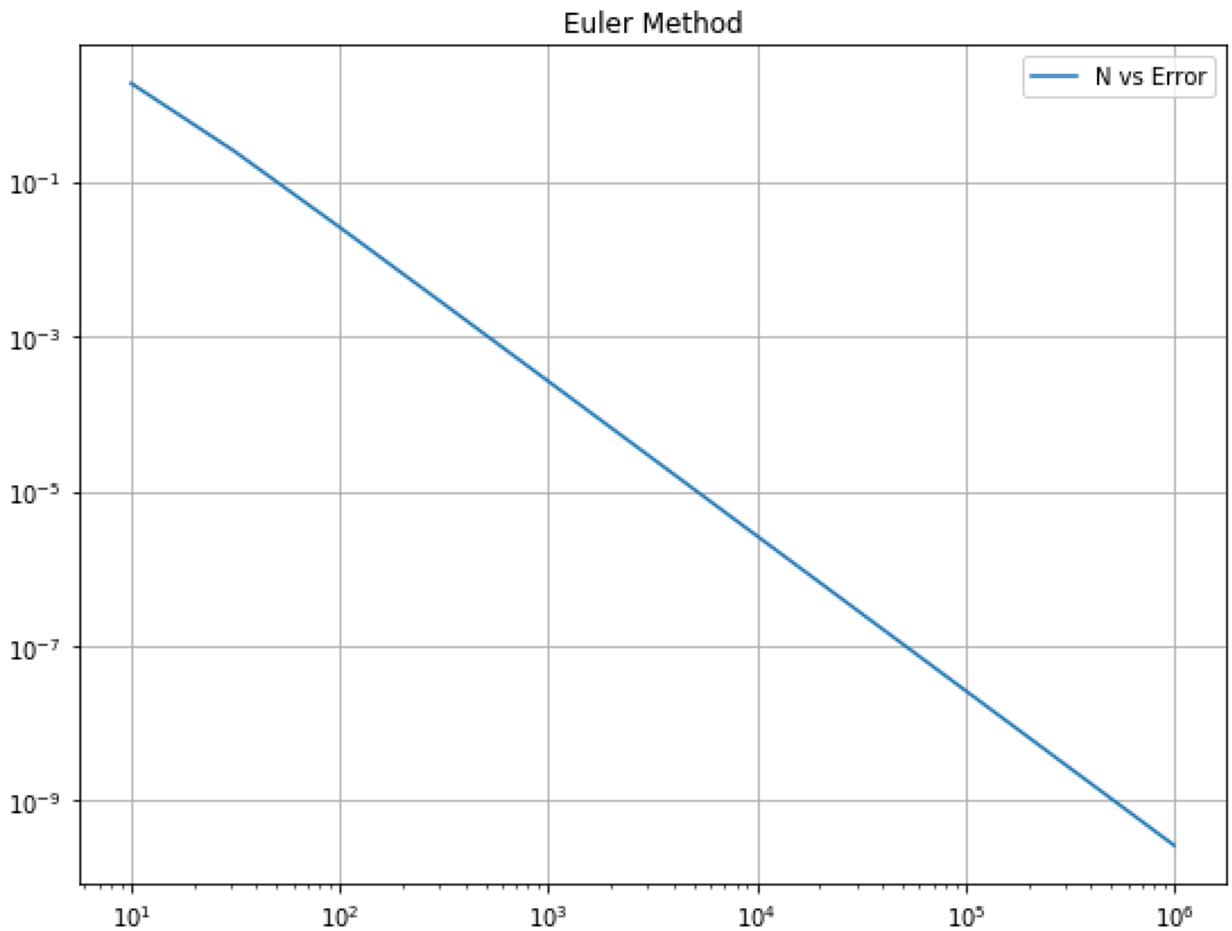
Q2: For this problem, I first implemented the AB2 method. I solved for the x values and t values and then I created a function to calculate the error, based on the error that was shown on the assignment document. This is a second order method. I chose to use 9 N values in this problem and as N increases, the error decreases linearly. The order of Forward Euler is 2.



Q3: The graph of the trapezoidal method is shown below. It is pretty much the same as the Forward Euler method shown above. The values differed minimally, but the slope is the same. The order of trapezoidal method is 2.



Q4: This one is similar to Q2, but it splits N into smaller subintervals. This calculation used both forward Euler and AB2, solving for x1 using forward Euler and inputting that into the Nlist (list of N values). This method is also of order 2. The difference for this one though is that when we are calculating the error for all the values N, we are substituting x1 for x1 computed through the forward Euler method.



Q5: To solve for b_{-1}, b_0, b_1, I used matrices. Using gauss-elimination, I determined the values oto be:

$b_{-1} = 5/12$

$b_0 = 8/12$ -or- $2/3$

$b_1 = 1/12$

The steps are shown below:

	X_1	X_2	X_3	b
1	1	1	1	1
2	-1	1	3	0
3	-2	1	-2	0

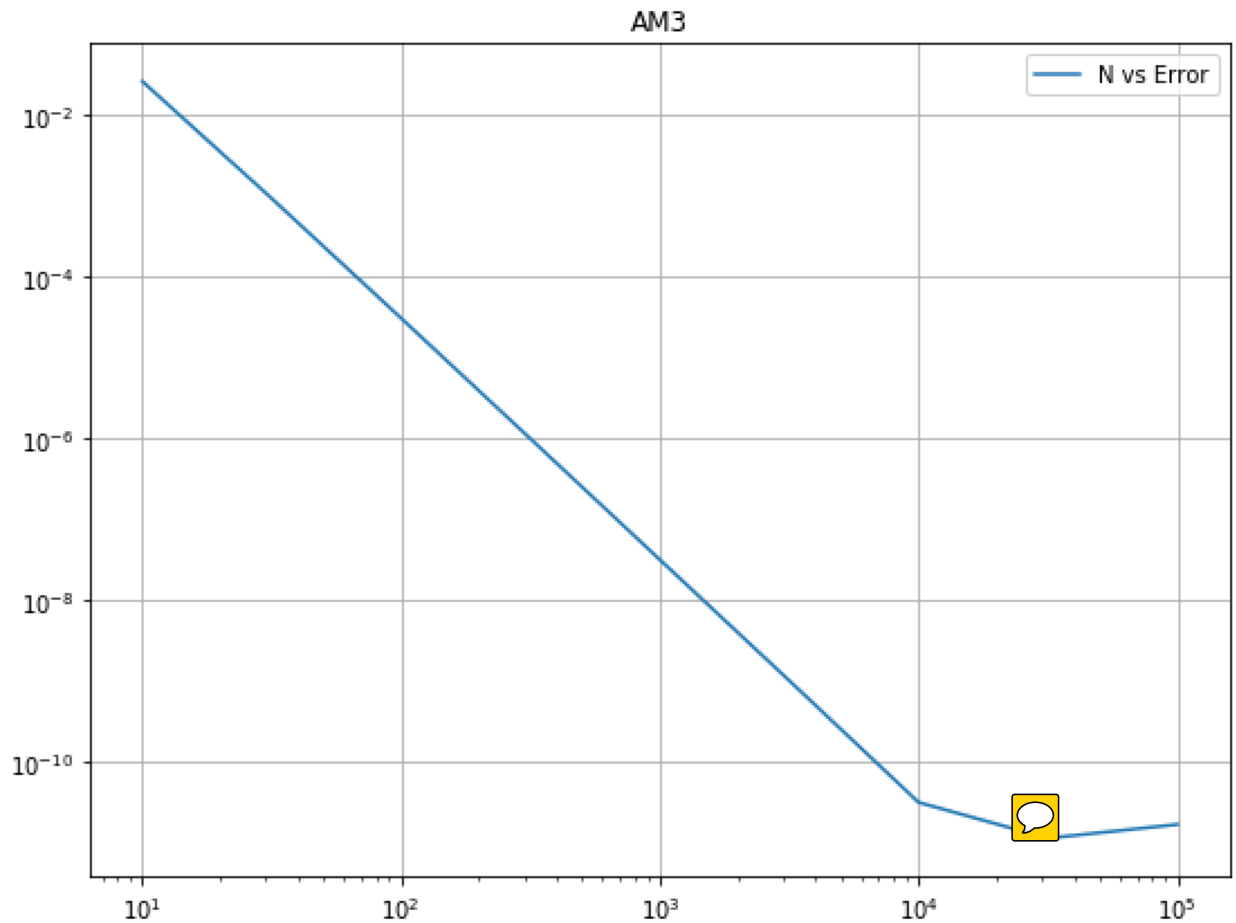
	X_1	X_2	X_3	b
1	1	1	1	1
2	0	2	4	1
3	0	3	0	2

	X_1	X_2	X_3	b
1	1	1	1	1
2	0	1	2	1/2
3	0	3	0	2

	X_1	X_2	X_3	b
1	1	0	-1	1/2
2	0	1	2	1/2
3	0	0	-6	1/2

	X_1	X_2	X_3	b
1	1	0	-1	1/2
2	0	1	2	1/2
3	0	0	1	-1/12

	X_1	X_2	X_3	b
1	1	0	0	5/12
2	0	1	0	2/3
3	0	0	1	-1/12



Q6: The error depends on the number of discretization points because the graph is not completely linear. The graph is linear for a certain number of points, but after a while (in this case when N = 10,000) the graph starts to be less linear and eventually the error value starts to increase again. When looking at the graph as a whole, the alpha value (order) was about 2.4, but when we reduced the number of N values included, the order was the true value of what It should be, 3.

I also had to solve for x1, and the number that I got for x1 was -0.0545171299774. This small value changed the graph from having an order of 2 (when x1=0) to having an order of 3.

Edited:
I solved for the x1 value in AM3 by using the trapezoid method as shown in a screenshot of my code below.

```
def AM3(x0, x1, N, delta):
    xnplus1 = lambda xn, tn, xnminus1, tnminus1, tnplus1: (xn + delta*(bminus1*-np.sin(tnplus1) + b0*(xn-np.sin(tn)) + b
    tValues = [0.0, delta]
    xValues = [x0, x(delta)]
    for n in np.arange(2*delta, np.pi+delta, delta):
        tValues.append(n)
        xValues.append(xnplus1(xValues[-1], tValues[-2], xValues[-2], tValues[-3], tValues[-1]))
    return [np.array(tValues), np.array(xValues)]
print x(delta)
ts,xs = AM3(x0, x1, N, delta)
calculateError(ts, xs)

Nlist = [int(np.sqrt(10)**(i+2)) for i in range(9)]
#print Nlist
errorList = []

for N in Nlist:
    delta = np.pi/N
    x1 = trapezoid(x0, delta, delta/N)
    ts, xs = AM3(x0, x1, N, delta)
    error = calculateError(ts, xs)
    errorList.append(error)
```