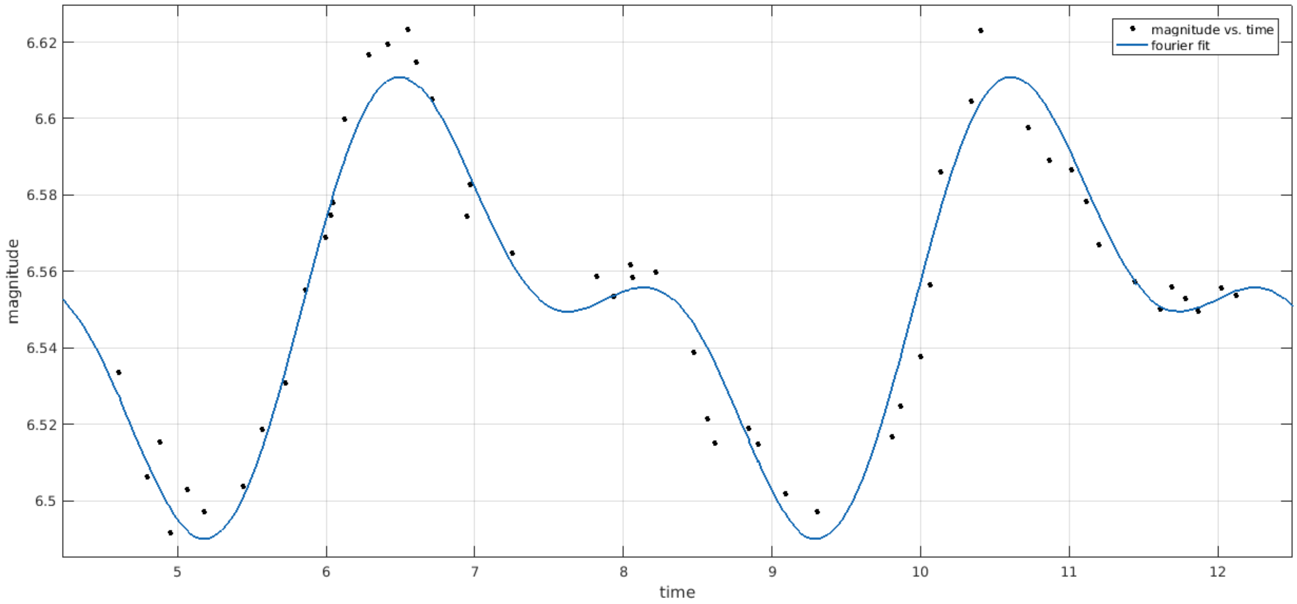
1. I used the curve fitting tool in Matlab to a fourier series with two terms (ie. y(x) = a0 + a1*cos(x*w) + b1*sin(x*w) + a2*cos(2*x*w) + b2*sin(2*x*w) as the resulting function). This resulted in the following values for a0, a1, a2, b1, b2 and w with 95% confidence interval in parentheses:

```
       a0 =       6.551  (6.549, 6.554)
       a1 =    -0.01973  (-0.02688, -0.01259)
       b1 =    -0.03736  (-0.04305, -0.03167)
       a2 =     0.02363  (0.017, 0.03025)
       b2 =     0.01372  (0.004723, 0.02271)
       w =       1.527  (1.504, 1.549)
```

Fitted function together with the original data points:



This is the best matlab can do in terms of residual so I couldn’t get a better residual by manually changing the periods.

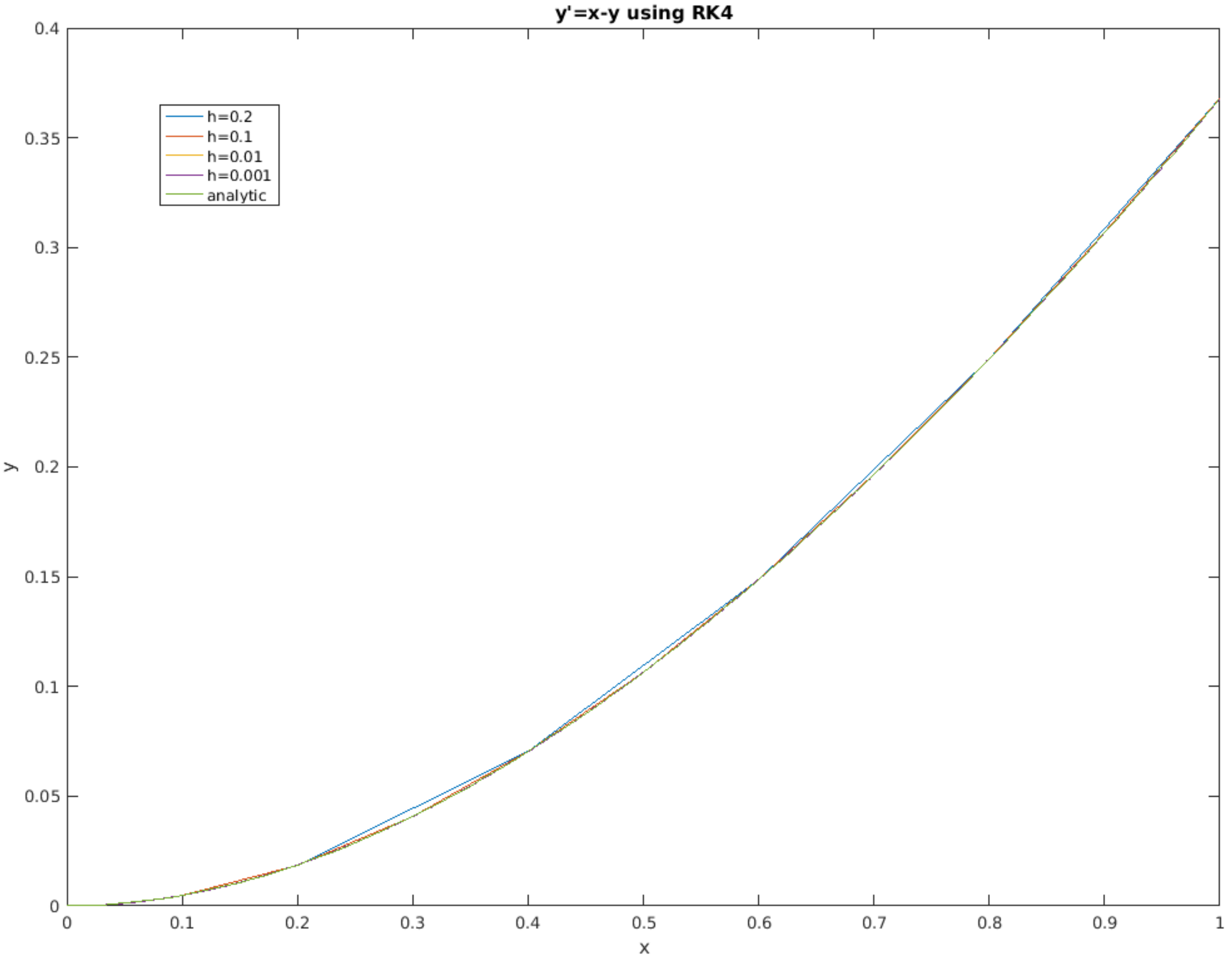
2&3. Plots on next page. Differences are so small that all runs but ones with h=0.2 are very difficult to tell apart. Outputs with numbers are also shown below. From them, it is easy to see that accuracy of Euler worsens faster when h gets larger whereas for RK4 the effect is much smaller.

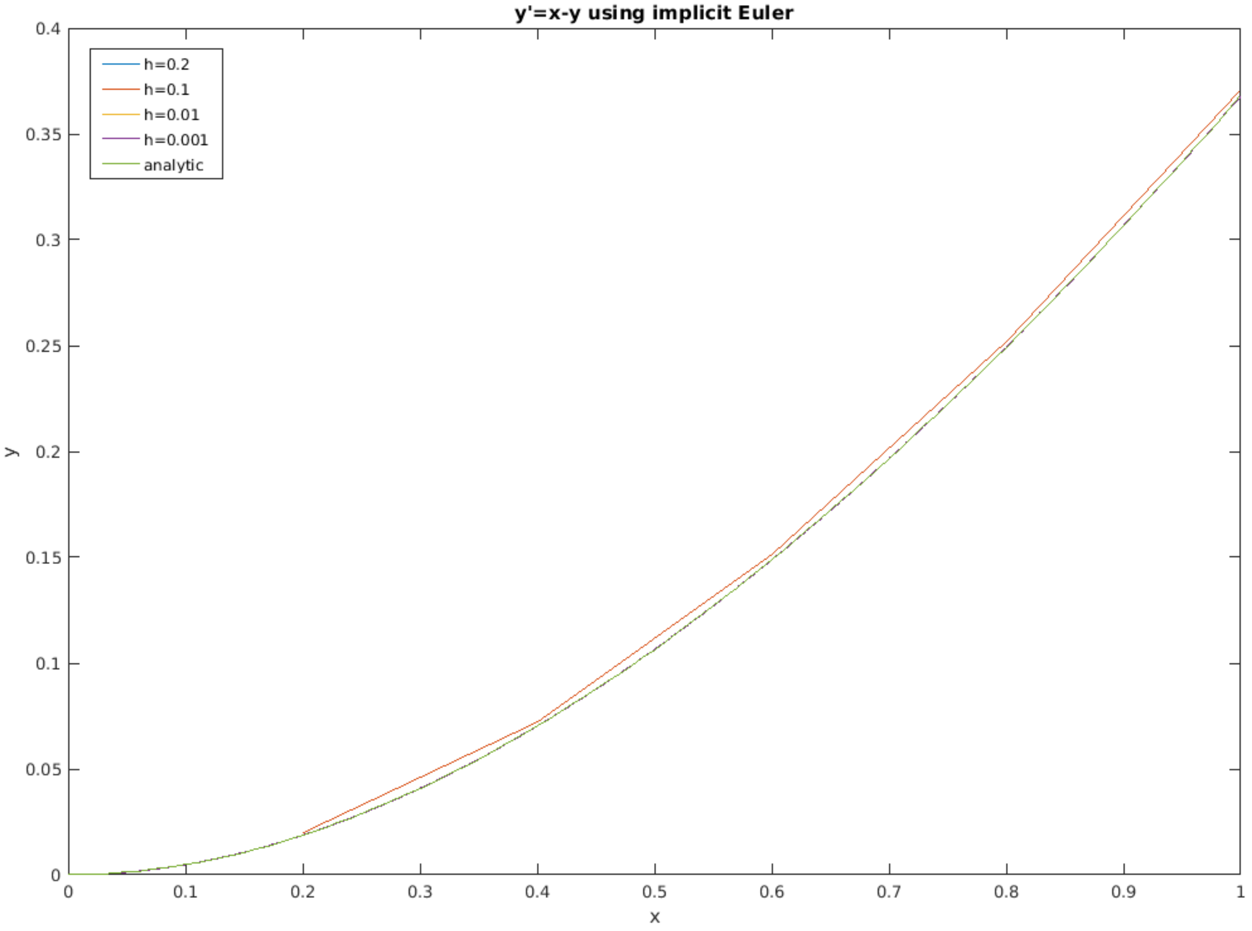
```
2a) h=0.200000:     0.370740
    time spent on 1 000 000 runs:  0.119084 s
2a) h=0.100000:     0.368541
    time spent on 1 000 000 runs:  0.175714 s
2a) h=0.010000:     0.367886
    time spent on 1 000 000 runs:  1.767559 s
2a) h=0.001000:     0.367880
    time spent on 1 000 000 runs:  17.667138 s

3) h=0.200000: 0.367885
    time spent on 1 000 000 runs:  0.136812 s
3) h=0.100000: 0.432871
    time spent on 1 000 000 runs:  0.328597 s
3) h=0.010000: 0.367879
    time spent on 1 000 000 runs:  3.088388 s
3) h=0.001000: 0.367879
    time spent on 1 000 000 runs:  30.908662 s
```

The output also includes timing results using standard C function clock(). We can see that both functions seem to scale as O(n) as their code would suggest, but RK has larger constant factor and is therefore slower. This is expected, as RK4 evaluates df(x, y) four times per loop whereas my implementation of implicit Euler does that only twice. In general RK4 needs more operations per loop and the number of loops is same for both RK4 and Euler for same h.

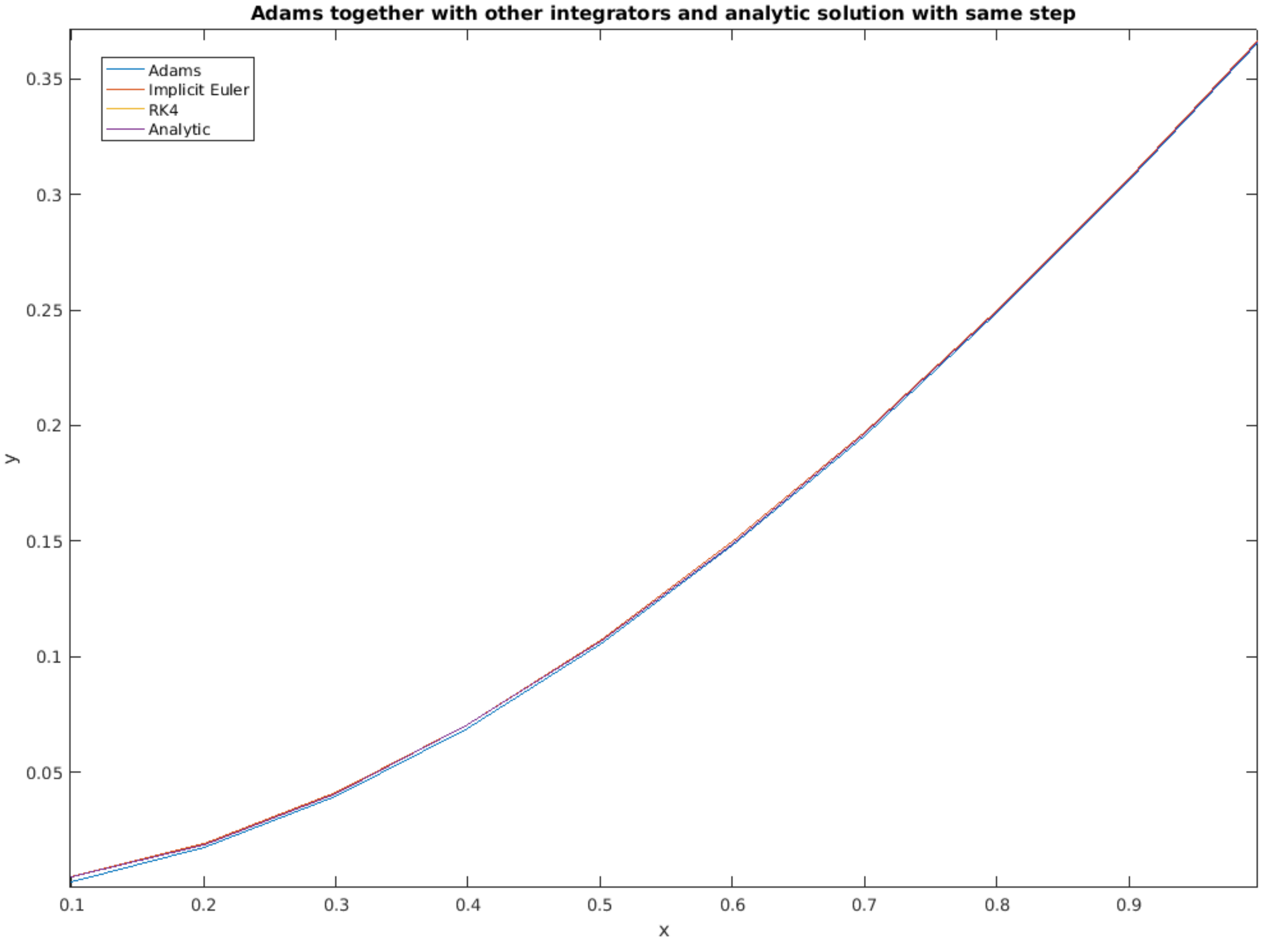
RMS error of the numerically integrated curve versus the analytical one evaluated in same values of x (corresponding to the square of the vertical difference in curves):

```
Euler:
h=0.2:    0.0052
h=0.1:    0.0016
h=0.01:   4.4663e-05
h=0.001:  1.0048e-06

RK4:
h=0.2:    1.0357e-05
h=0.1:    6.0006e-07
h=0.01:   1.3697e-07
h=0.001:  4.4986e-07
```


y'=x-y using RK4
h=0.2
h=0.1
h=0.01
h=0.001
analytic
y
x
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


y'=x-y using implicit Euler
h=0.2
h=0.1
h=0.01
h=0.001
analytic
y
x
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

4)   I used integration step for three known points given in slides (ode pdf page 26) and corrector as given in same pdf page 28 (lower slide, lower equation). Latter actually kind of has four terms, but I figured that’s ok because I can use the three I know from previous steps and then fourth I got from predictor. I programmed everything from scratch using C, my code is in same file as ex 2 and 3 functions. Below is plot produced with Adams predictor-corrector together with other implemented functions with same step size h=0.1.


Adams together with other integrators and analytic solution with same step
Adams
Implicit Euler
RK4
Analytic
y
x