Silesian University of Technology
Faculty of Automatic Control,
Electronic and Computer Science
Major: Makro sem.4

Gliwice
Academic year 2018/2019

Numerical Methods
Laboratory Exercise

Ex.10 Aproximate Solving of Ordinary Differential Equqtions

Group 2, Section 4
Mateusz Warmuz
Szymon Skorupa

Exercise preformed on: 23.05.2019

1. Analytical solution of the first order equation

$$y' = -2y + 1e^{-2x}$$
$$y' + 2y = 0 \quad compl\ eq$$
$$y' = -2y$$
$$\frac{y'}{y} = -2 \quad or \quad y = 0$$
$$\int \frac{y'}{y} dx = \int -2dx$$
$$ln\left|y\right| = -2x + C_1$$
$$\left|y\right| = e^{-2x+C_1}$$
$$y = \pm e^{C_1} e^{-2x}$$
$$y = C_2 e^{-2x} \quad or \quad y = 0$$
$$y_H = Ce^{-2x}$$
$$y' = C'(x)e^{-2x} + C(x)e^{-2x}(-2)$$
$$y' = C'(x)e^{-2x} - 2C(x)e^{-2x}$$
$$C'(x)e^{-2x} - 2C(x)e^{-2x} + 2C(x)e^{-2x} = 1e^{-2x}$$
$$C'(x)e^{-2x} = e^{-2x}$$
$$C'(x) = 1$$
$$C(x) = \int 1dx$$
$$C(x) = x$$
$$y = xe^{-2x} + Ce^{-2x} \quad GS\ of\ diff.\ eq$$
$$-10 = 0 \cdot e^{-2*0} + C \cdot e^{-2*0}$$
$$C = -10$$
$$y = xe^{-2x} - 10e^{-2x} \quad particular\ solution\ of\ GS$$

2. Program codes

Euler

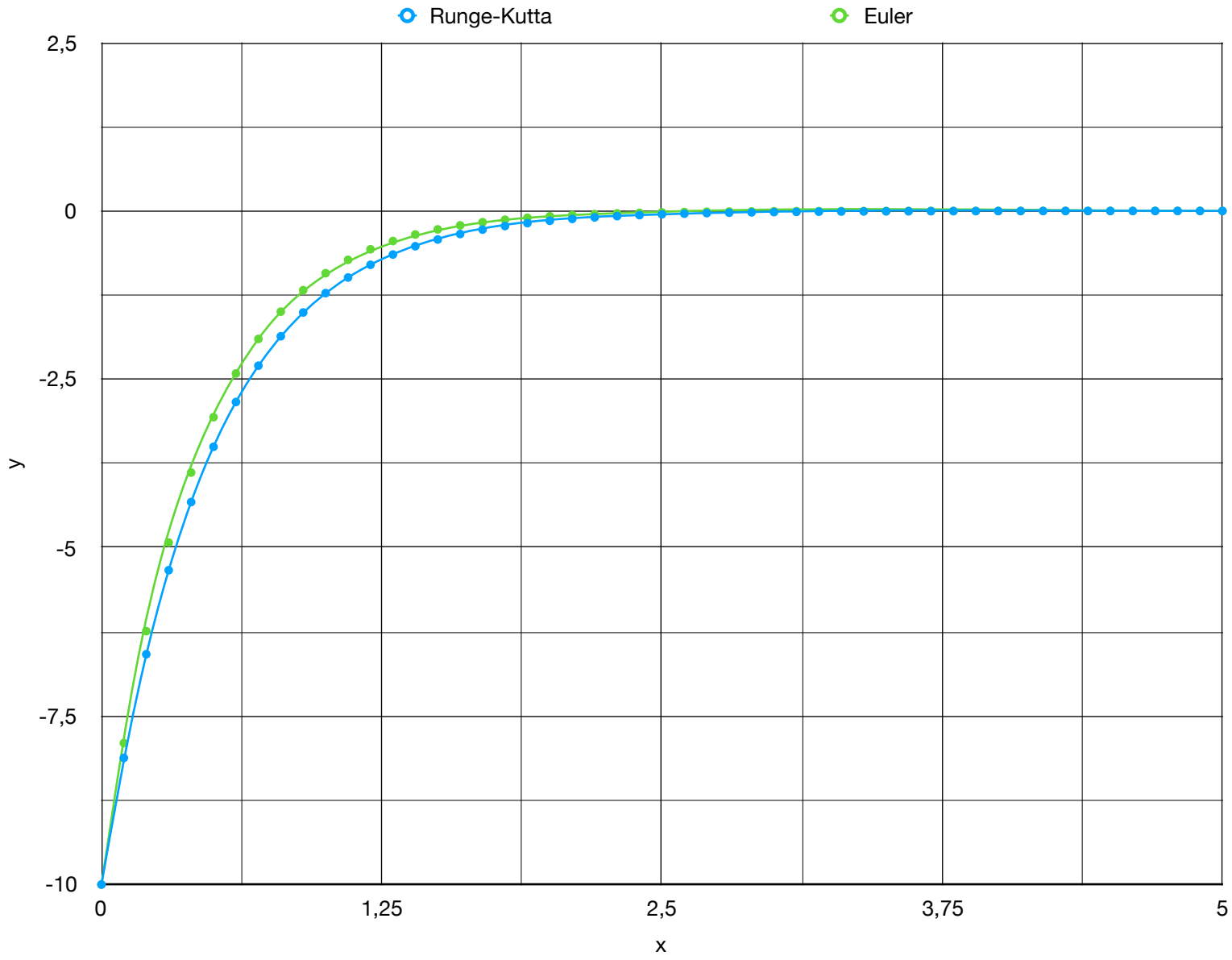
```
#include <iostream>
#include <cmath>
using std::cout;
using std::cin;
using std::endl;

double function(double x, double y){
    return -2*y+1*exp(-2*x);
}

double euler(double x, double y, double h, int n){
    double* arguments;
    double* values;
    double* triangle;

    arguments = new double [n+1];
    values = new double [n+1];
    triangle = new double [n];

    values[0]=y;
    for(int i =0; i<n ; i++){
        arguments[i]=x+i*h;
        triangle[i]=h*function(arguments[i],values[i]);
        values[i+1]=values[i]+triangle[i];
        cout<<values[i]<<endl;
    }
    return* values;
}

int main() {
    double x=0;
    double y=-10;
    double h=0.1;
    int n=51;
    euler(x,y,h,n);
}
```

```
#include <cmath>
#include <iostream>

double function(double x, double y){
    return -2*y+1*exp(-2*x);
}

double k(double x, double y, double h){
    double k[4] = {0};
    double deltaY = 0;

    k[0] = h*function(x,y);
    k[1] = h*function(x+0.5*h, y+0.5*k[0]);
    k[2] = h*function(x+0.5*h, y+0.5*k[1]);
    k[3] = h*function(x+h, y+k[2]);

    deltaY = (k[0] + 2*k[1] + 2*k[2] + k[3])/6;

    return deltaY;
}


int main() {

    double x0 = 0.1;
    double y0 = -10;
    double h = 0.1;        \\step
    double n = 50;
    std::cout << y0 << std::endl;
    for(int i = 1; i <= n; i++){
        k(x0, y0, h);
        y0 += k(x0, y0, h);
        x0 = i*h;
        std::cout << y0 << std::endl;
    }

}
```

3. Programs' outputs

	Runge-Kutta	Euler		Runge-Kutta	Euler
x	y		x	y	
0	-10	-10	2,6	-0,040927	-0,0169068
0,1	-8,1203	-7,9	2,7	-0,0330566	-0,0129738
0,2	-6,58133	-6,23813	2,8	-0,0266948	-0,00992738
0,3	-5,33348	-4,92347	2,9	-0,0215532	-0,00757211
0,4	-4,32176	-3,88389	3	-0,0173984	-0,00575494
0,5	-3,50159	-3,06218	3,1	-0,0140417	-0,00435607
0,6	-2,83675	-2,41296	3,2	-0,0113303	-0,00328192
0,7	-2,29788	-1,90025	3,3	-0,00914045	-0,00245938
0,8	-1,86116	-1,49554	3,4	-0,00737221	-0,00183146
0,9	-1,50727	-1,17624	3,5	-0,00594469	-0,00135379
1	-1,22052	-0,924463	3,6	-0,00479246	-0,000991847
1,1	-0,988197	-0,72603	3,7	-0,00386262	-0,000718819
1,2	-0,799998	-0,569749	3,8	-0,00311241	-0,00051393
1,3	-0,647558	-0,446727	3,9	-0,00250727	-0,000361099
1,4	-0,524096	-0,349955	4	-0,00201924	-0,000247906
1,5	-0,424117	-0,273883	4,1	-0,00162575	-0,000164778
1,6	-0,343162	-0,214127	4,2	-0,00130857	-0,000104357
1,7	-0,277621	-0,167226	4,3	-0,00105296	-6,0999E-05
1,8	-0,224565	-0,130443	4,4	-0,000847021	-3,03887E-05
1,9	-0,181622	-0,101622	4,5	-0,000681144	-9,23761E-06
2	-0,146869	-0,0790607	4,6	-0,000547572	4,95089E-06
2,1	-0,118747	-0,061417	4,7	-0,000440043	1,40647E-05
2,2	-0,0959942	-0,047634	4,8	-0,000353505	1,95241E-05
2,3	-0,0775885	-0,0368795	4,9	-0,000283881	2,23922E-05
2,4	-0,0627013	-0,0284984	5	-0,000227883	2,34589E-05
2,5	-0,0506619	-0,0219758			

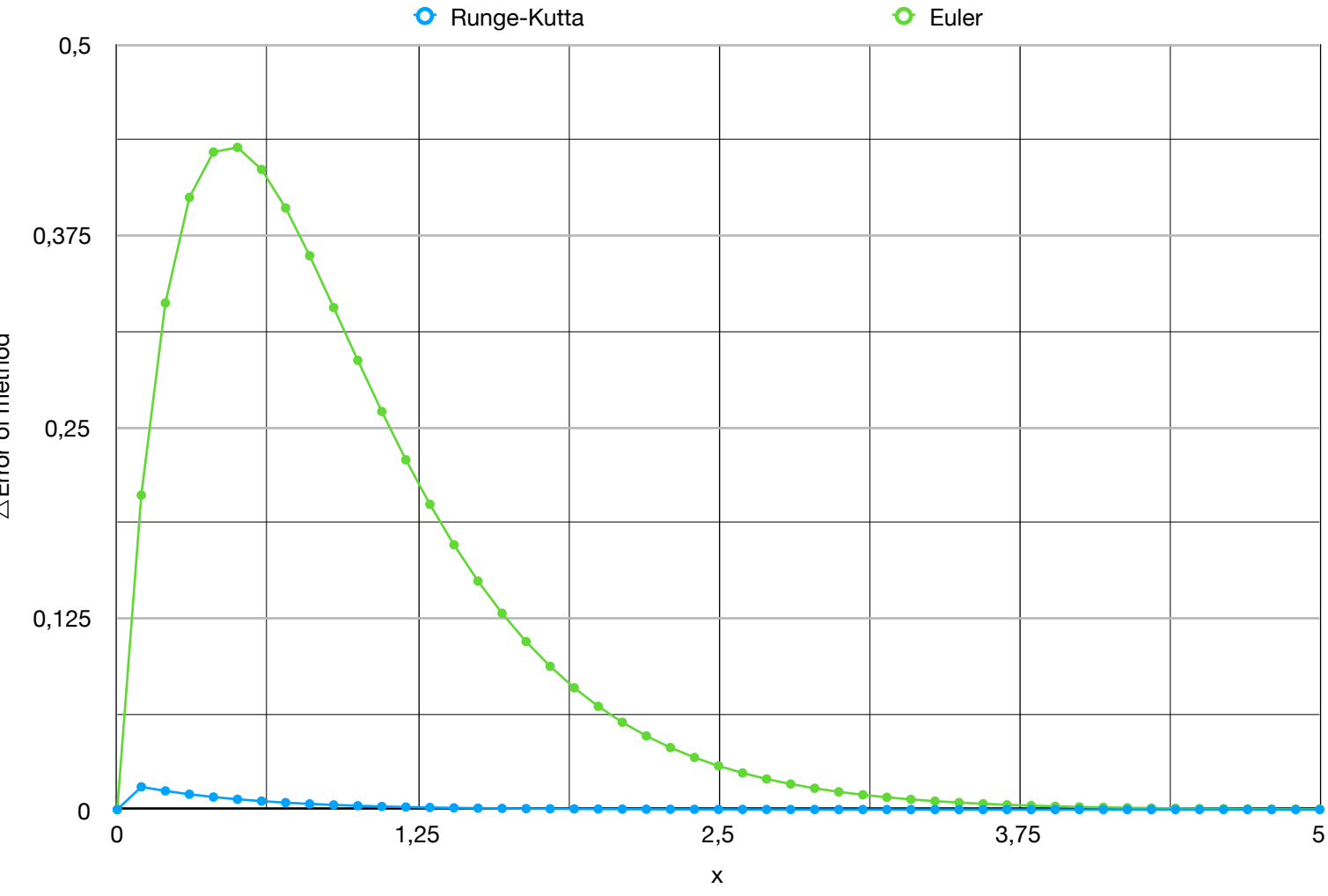
4. Graph of solutions obtained using Euler and Runge-Kutta method



5. Analysis of both method's error

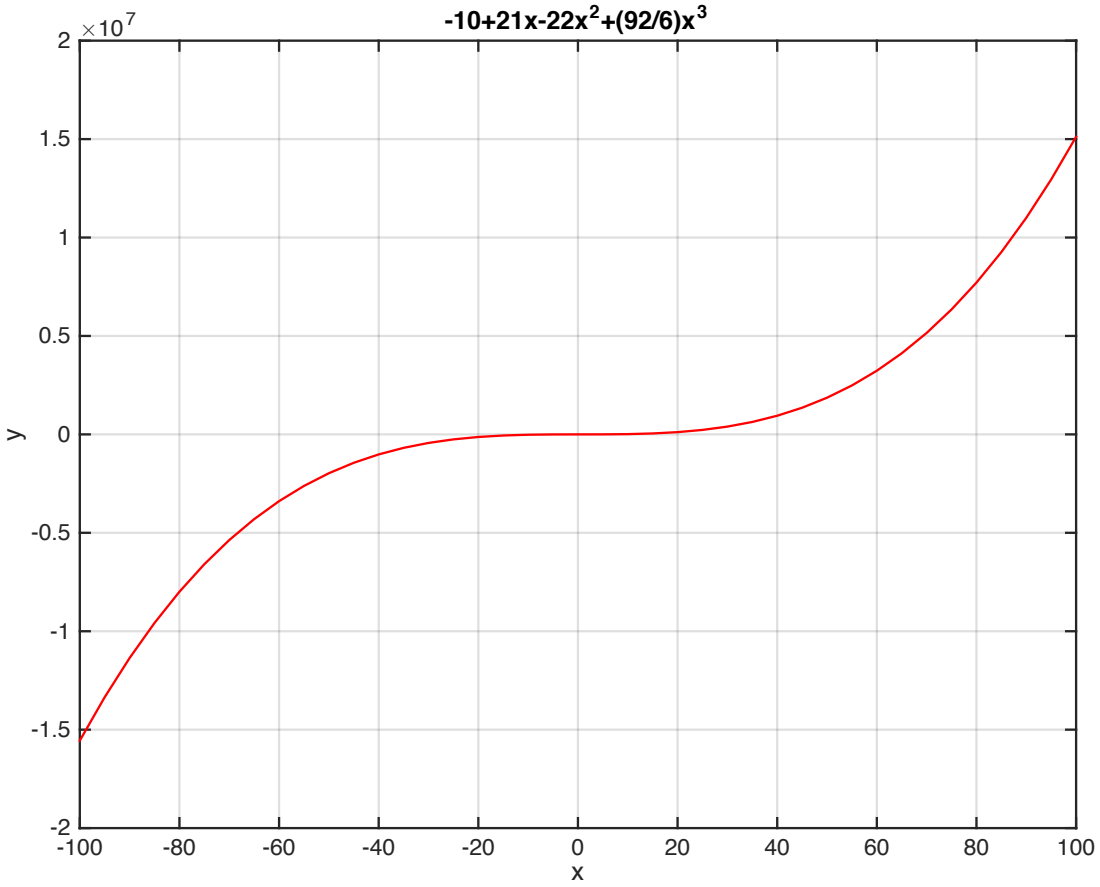
	Runge-Kutta	Euler	Analytical	△Runge-Kutta	△Euler
x	y				
0	-10	-10	-10	0	0
0,1	-8,1203	-7,9	-8,1054345	0,0148655	0,2054345
0,2	-6,58133	-6,23813	-6,5691365	0,0121935	0,3310065
0,3	-5,33348	-4,92347	-5,3234729	0,0100071	0,4000029
0,4	-4,32176	-3,88389	-4,3135581	0,0082019	0,4296681
0,5	-3,50159	-3,06218	-3,4948547	0,0067353	0,4326747
0,6	-2,83675	-2,41296	-2,8312256	0,0055244	0,4182656
0,7	-2,29788	-1,90025	-2,2933518	0,0045282	0,3931018
0,8	-1,86116	-1,49554	-1,8574480	0,0037120	0,3619080
0,9	-1,50727	-1,17624	-1,5042199	0,0030501	0,3279799
1	-1,22052	-0,924463	-1,2180175	0,0025025	0,2935545
1,1	-0,988197	-0,72603	-0,9861481	0,0020489	0,2601181
1,2	-0,799998	-0,569749	-0,7983180	0,0016800	0,2285690
1,3	-0,647558	-0,446727	-0,6461801	0,0013779	0,1994531
1,4	-0,524096	-0,349955	-0,5229665	0,0011295	0,1730115
1,5	-0,424117	-0,273883	-0,4231901	0,0009269	0,1493071
1,6	-0,343162	-0,214127	-0,3424025	0,0007595	0,1282755
1,7	-0,277621	-0,167226	-0,2769981	0,0006229	0,1097721
1,8	-0,224565	-0,130443	-0,2240545	0,0005105	0,0936115
1,9	-0,181622	-0,101622	-0,1812033	0,0004187	0,0795813
2	-0,146869	-0,0790607	-0,1465251	0,0003439	0,0674644
2,1	-0,118747	-0,061417	-0,1184651	0,0002819	0,0570481
2,2	-0,0959942	-0,047634	-0,0957633	0,0002309	0,0481293
2,3	-0,0775885	-0,0368795	-0,0773991	0,0001894	0,0405196
2,4	-0,0627013	-0,0284984	-0,0625461	0,0001552	0,0340477
2,5	-0,0506619	-0,0219758	-0,0505346	0,0001273	0,0285588
2,6	-0,040927	-0,0169068	-0,0408226	0,0001044	0,0239158
2,7	-0,0330566	-0,0129738	-0,0329710	0,0000856	0,0199972
2,8	-0,0266948	-0,00992738	-0,0266246	0,0000702	0,0166972
2,9	-0,0215532	-0,00757211	-0,0214956	0,0000576	0,0139235
3	-0,0173984	-0,00575494	-0,0173513	0,0000471	0,0115963
3,1	-0,0140417	-0,00435607	-0,0140031	0,0000386	0,0096470
3,2	-0,0113303	-0,00328192	-0,0112986	0,0000317	0,0080167
3,3	-0,00914045	-0,00245938	-0,0091145	0,0000260	0,0066551

3,4	-0,00737221	-0,00183146	-0,0073509	0,0000213	0,0055195
3,5	-0,00594469	-0,00135379	-0,0059272	0,0000175	0,0045734
3,6	-0,00479246	-0,000991847	-0,0047781	0,0000143	0,0037863
3,7	-0,00386262	-0,000718819	-0,0038509	0,0000117	0,0031321
3,8	-0,00311241	-0,00051393	-0,0031028	0,0000096	0,0025889
3,9	-0,00250727	-0,000361099	-0,0024994	0,0000079	0,0021383
4	-0,00201924	-0,000247906	-0,0020128	0,0000065	0,0017649
4,1	-0,00162575	-0,000164778	-0,0016205	0,0000053	0,0014557
4,2	-0,00130857	-0,000104357	-0,0013042	0,0000043	0,0011999
4,3	-0,00105296	-0,0001	-0,0010494	0,0000036	0,0009884
4,4	-0,000847021	-0,00003	-0,0008441	0,0000029	0,0008137
4,5	-0,000681144	-0,00001	-0,0006788	0,0000024	0,0006695
4,6	-0,000547572	0,00000	-0,0005456	0,0000020	0,0005506
4,7	-0,000440043	0,00001	-0,0004384	0,0000016	0,0004525
4,8	-0,000353505	0,00002	-0,0003522	0,0000013	0,0003717
4,9	-0,000283881	0,00002	-0,0002828	0,0000011	0,0003052
5	-0,000227883	0,00002	-0,0002270	0,0000009	0,0002505



6. Taylor Series

$$y^{(1)}(x) = -2y + e^{-2x} \qquad y(0) = -10$$
$$y^{(1)}(0) = -2 \cdot (-10) + e^{-2 \cdot 0} = 20 + 1 = 21$$
$$y^{(2)}(x) = -2y^{(1)}(x) - 2e^{-2x}$$
$$y^{(2)}(0) = -2 \cdot 21 - 2e^{-2 \cdot 0} = -44$$
$$y^{(3)}(x) = -2y^{(2)}(x) + 4e^{-2x}$$
$$y^{(3)}(0) = -2 \cdot (-44) + 4e^{-2 \cdot 0} = 92$$
$$y(x) \approx y(0) + y^{(1)}(0)x + \frac{1}{2}y^{(2)}(0)x^2 + \frac{1}{6}y^{(3)}(0)x^3$$
$$\approx -10 + 21x - 22x^2 + \frac{92}{6}x^3$$



7. Conclusions:

When testing both given methods, euler and runge-kutta against the analytical solution we can clearly see that runge-kutta method follows the ideal function way better than the euler. Especially when testing for lower x the runge-kutta method performs way better than euler.