METODY NUMERYCZNE

LISTA 7

WIOLETTA ŁUPKOWSKA, 244831

CZWARTEK, 9:15

1.1)

```
Klasa umożliwiająca całkowanie równań różniczkowych zwyczajnych pierwszego
rzędu metodą Eulera:
-
import java.util.ArrayList;

public class Euler {

    Licz licz= new Licz();


    public ArrayList liczEuler(double przedzialL, double przedzialU, double krok){
        ArrayList<Double> listXEuler = new ArrayList();
        ArrayList<Double> listXTrue = new ArrayList();
        ArrayList<Double> listT = new ArrayList();
        double t=przedzialL;
        double xTrue=0;
        double xEulerPrevious=1;
        double xEulerNext;
        do{
            listT.add(t);
            xTrue=licz.xTrue(xTrue,t);
            listXTrue.add(xTrue);
            xEulerNext= xEulerPrevious + licz.ft(xEulerPrevious,t)*krok;
            listXEuler.add(xEulerNext);
            xEulerPrevious=xEulerNext;
            t+=krok;
        }while(t<=przedzialU);
        System.out.println("ListXTrue: ");
        System.out.println(listXTrue);
         return listXEuler;

    }

}
```

```
public class Licz implements Function {

    @Override
    public double xTrue (double x){  //tutaj całke
        double xTrue= -0.5*Math.pow(x,4) + 4*Math.pow(x,3)- 10*Math.pow(x,2) +
8.5*x + 1;  //rownanie z listy 6
        //double xTrue= -0.0000797121*Math.pow(x, 21/5)-
0.0611845*Math.pow(x,3)+9.81*x; //skoczek
        //double xTrue= Math.exp(1/3*Math.pow(x,3)-1.1*x); // zad.5
        return xTrue;
    }
    @Override
    public double ft(double x, double t){
        double ft= -2*Math.pow(t,3)+ 12*Math.pow(t,2) -20*t+ 8.5;
        //double ft=9.81-
12.5/68.1*(Math.pow(t,2)+8.3/(Math.pow(46,2.2))*Math.pow(t,3.2));
        //double ft=x*Math.pow(t,2)-1.1*x;
        return ft;
    }
}
```

```
import java.util.ArrayList;

public interface Function {

     //ArrayList liczEuler(double przedzialL, double przedzialU, double krok);
     double xTrue(double t);
     double ft(double x, double t);
}
```

2.1)

Klasa umożliwiająca całkowanie równań różniczkowych zwyczajnych pierwszego rzędu zmodyfikowaną metodą Eulera:

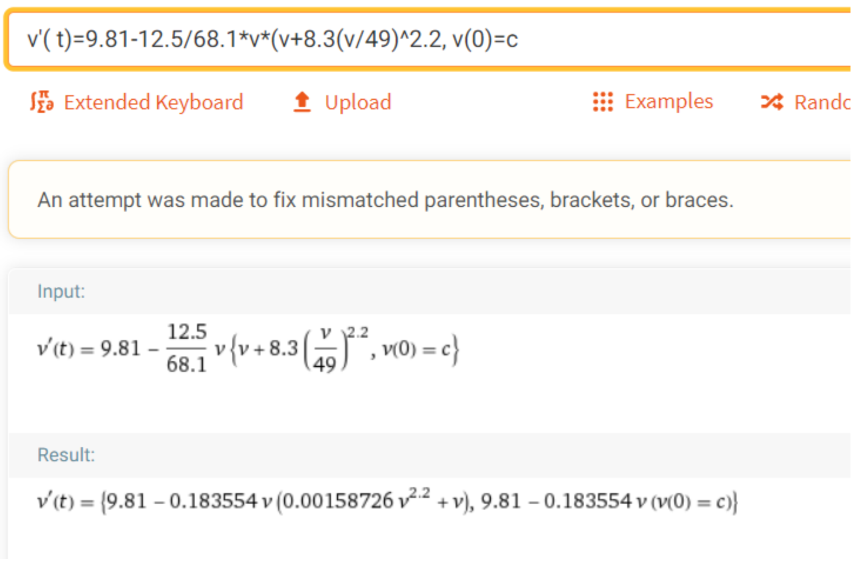
```
import java.util.ArrayList;

public class ZmodyfikowanyEuler {

        Licz licz= new Licz();


    public ArrayList liczEuler(double przedzialL, double przedzialU, double krok){
         ArrayList<Double> listXEuler = new ArrayList();
         ArrayList<Double> listXTrue = new ArrayList();
         ArrayList<Double> listT = new ArrayList();

        double tSrodkowy;
        double t=przedzialL;
        double xEulerNext;
        double xTrue=0;
        double xEulerPrevious=1; //==0
        do{

            tSrodkowy=t+krok/2;
            double x_temp = xEulerPrevious+licz.ft(xEulerPrevious,t)*krok/2;
            listT.add(t);
            xTrue=licz.xTrue(xTrue, t);
            listXTrue.add(xTrue);
            xEulerNext = xEulerPrevious + licz.ft(x_temp, tSrodkowy)*krok;
            listXEuler.add(xEulerNext);
            xEulerPrevious=xEulerNext;
            t+=krok;
        }while(t<=przedzialU);
        System.out.println("ListXTrue: ");
        System.out.println(listXTrue);
        System.out.println("t: ");
        System.out.println(listT);
    return listXEuler;

    }


}
```

3.1)

```
import java.util.ArrayList;

public class Test {

    static Euler euler = new Euler();
    static ZmodyfikowanyEuler zmodyfikowanyEuler = new ZmodyfikowanyEuler();
    static ArrayList<Double> listXEuler = new ArrayList();
    static ArrayList<Double> listXZmodyfikowanyEuler = new ArrayList();

    public static void main(String[] args) {

        listXEuler = euler.liczEuler(0,4,0.5);
        listXZmodyfikowanyEuler = zmodyfikowanyEuler.liczEuler(0,4,0.5);
        System.out.println("Euler: ");
        System.out.println(listXEuler);
        System.out.println("Zmodyfikowany Euler: ");
        System.out.println(listXZmodyfikowanyEuler);
    }
}
```

//tutaj poprawione//

```
public class Licz implements Function {

    @Override
    public double xTrue (double x, double t){ //calka, rozwiazanie
        double xTrue= -0.5*Math.pow(x,4) + 4*Math.pow(x,3)- 10*Math.pow(x,2) +
8.5*x + 1;  //rownanie z listy 6
        //double xTrue= (0-53.4449)*exp(-0.183554*t)+53.4449; //skoczek //zerowe
warunki poczaktkowe?
        // double xTrue= (9.81-0.183554*x*(0.00158726*v^2.2+v);
        //double xTrue=Math.exp(1./3.*(Math.pow(t,3)) - 1.1*x); // zad.5.//ok
        return xTrue;
    }


    @Override
    public double ft(double x, double t){  //pierwotna, to przyblizam
        double ft= -2*Math.pow(x,3)+ 12*Math.pow(x,2) -20*x+ 8.5;
        //double xTrue= 9.81+12.5/68.1*x
        //double ft=9.81-
12.5/68.1*(Math.pow(t,2)+8.3/(Math.pow(46,2.2))*Math.pow(t,3.2));
        //double ft=x*Math.pow(t,2)-1.1*x;
        return ft;
    }
}
```


v'(t)=9.81-12.5/68.1*v*(v+8.3(v/49)^2.2, v(0)=c

An attempt was made to fix mismatched parentheses, brackets, or braces.

Input:

$$v'(t) = 9.81 - \frac{12.5}{68.1} v \left\{v + 8.3 \left(\frac{v}{49}\right)^{2.2}, v(0) = c\right\}$$

Result:

$$v'(t) = \{9.81 - 0.183554\, v\, (0.00158726\, v^{2.2} + v),\ 9.81 - 0.183554\, v\, (v(0) = c)\}$$

4.1)

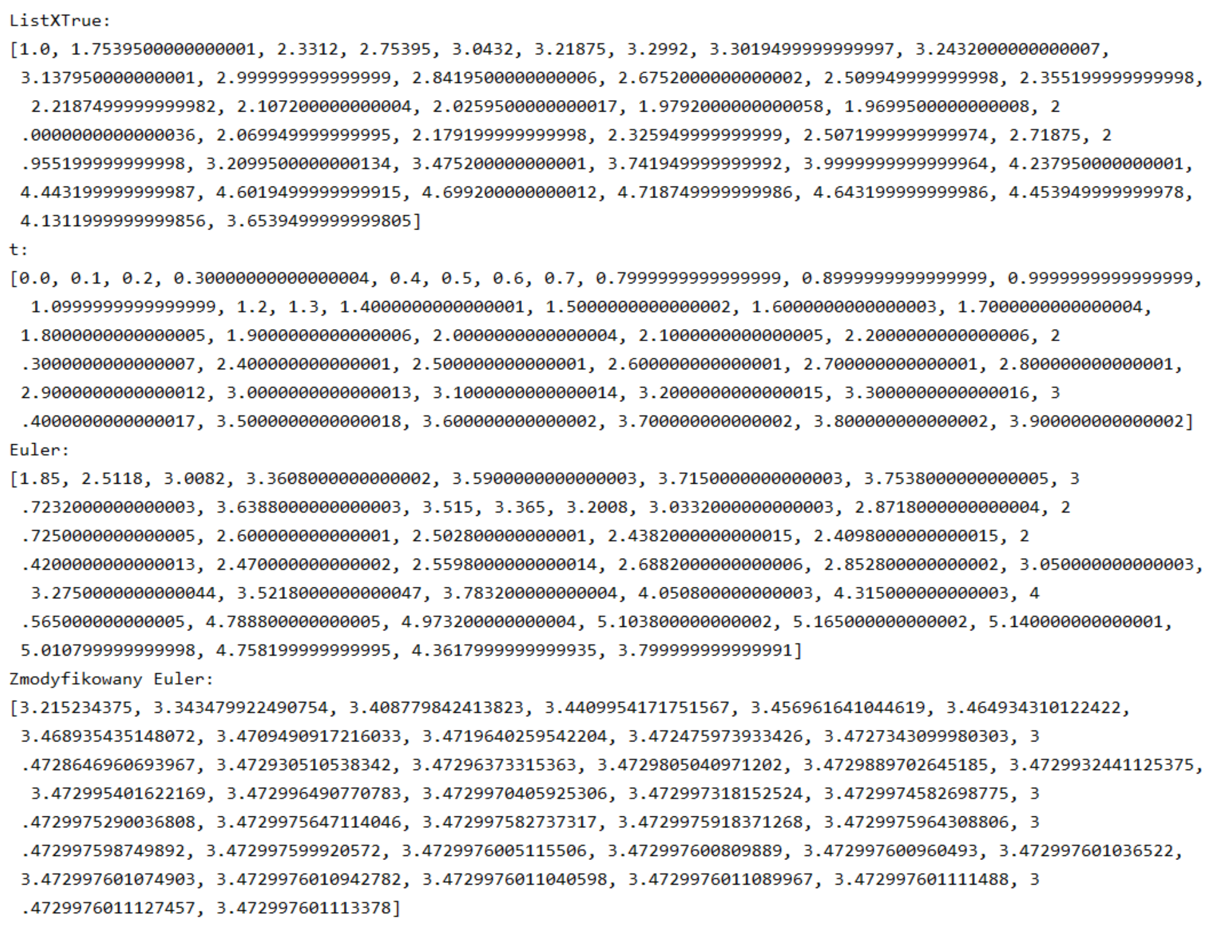
```
public class Licz implements Function {

    @Override
    public double xTrue (double x, double t){ //calka, rozwiazanie
        //double xTrue= -0.5*Math.pow(x,4) + 4*Math.pow(x,3)- 10*Math.pow(x,2) +
8.5*x + 1;  //rownanie z listy 6
        double xTrue= (0-53.4449)*Math.exp(-0.183554*t)+53.4449; //skoczek
//zerowe warunki poczaktkowe?
        // double xTrue= (9.81-0.183554*x*(0.00158726*v^2.2+v);
        //double xTrue=Math.exp(1./3.*(Math.pow(t,3)) - 1.1*x); // zad.5.//ok
        return xTrue;
    }


    @Override
    public double ft(double x, double t){  //pierwotna, to przyblizam
        ///double ft= -2*Math.pow(x,3)+ 12*Math.pow(x,2) -20*x+ 8.5;
        double ft= 9.81+12.5/68.1*x;
        //double ft=9.81-
12.5/68.1*(Math.pow(t,2)+8.3/(Math.pow(46,2.2))*Math.pow(t,3.2));
        //double ft=x*Math.pow(t,2)-1.1*x;
        return ft;
    }
}
```

5.1a)

```
public class Licz implements Function {

    @Override
    public double xTrue (double x, double t){ //calka, rozwiazanie
        //double xTrue= -0.5*Math.pow(x,4) + 4*Math.pow(x,3)- 10*Math.pow(x,2) +
8.5*x + 1;  //rownanie z listy 6
        //double xTrue= (0-53.4449)*Math.exp(-0.183554*t)+53.4449; //skoczek
//zerowe warunki poczaktkowe?
         //double xTrue= (9.81-0.183554*x*(0.00158726*Math.pow(x,2.2)+x));
        double xTrue=Math.exp(1./3.*(Math.pow(t,3)) - 1.1*x); // zad.5.//ok
        return xTrue;
    }


    @Override
    public double ft(double x, double t){  //pierwotna, to przyblizam
        ///double ft= -2*Math.pow(x,3)+ 12*Math.pow(x,2) -20*x+ 8.5;
        //double ft= 9.81+12.5/68.1*x;
       // double ft=9.81-
12.5/68.1*(Math.pow(t,2)+8.3/(Math.pow(46,2.2))*Math.pow(t,3.2));
        double ft=x*Math.pow(t,2)-1.1*x;
        return ft;
    }
}
```

3.2)

Krok 0.1



```
ListXTrue:
[1.0, 1.7539500000000001, 2.3312, 2.75395, 3.0432, 3.21875, 3.2992, 3.3019499999999997, 3.2432000000000007,
 3.137950000000001, 2.999999999999999, 2.8419500000000006, 2.6752000000000002, 2.509949999999998, 2.355199999999998,
  2.2187499999999982, 2.107200000000004, 2.0259500000000017, 1.9792000000000058, 1.9699500000000008, 2
 .0000000000000036, 2.069949999999995, 2.179199999999998, 2.325949999999999, 2.5071999999999974, 2.71875, 2
 .955199999999998, 3.2099500000000134, 3.475200000000001, 3.741949999999992, 3.9999999999999964, 4.237950000000001,
 4.443199999999987, 4.6019499999999915, 4.699200000000012, 4.718749999999986, 4.643199999999986, 4.453949999999978,
 4.1311999999999856, 3.6539499999999805]
t:
[0.0, 0.1, 0.2, 0.30000000000000004, 0.4, 0.5, 0.6, 0.7, 0.7999999999999999, 0.8999999999999999, 0.9999999999999999,
  1.0999999999999999, 1.2, 1.3, 1.4000000000000001, 1.5000000000000002, 1.6000000000000003, 1.7000000000000004,
 1.8000000000000005, 1.9000000000000006, 2.0000000000000004, 2.1000000000000005, 2.2000000000000006, 2
 .3000000000000007, 2.400000000000001, 2.500000000000001, 2.600000000000001, 2.700000000000001, 2.800000000000001,
 2.9000000000000012, 3.0000000000000013, 3.1000000000000014, 3.2000000000000015, 3.3000000000000016, 3
 .4000000000000017, 3.5000000000000018, 3.600000000000002, 3.700000000000002, 3.800000000000002, 3.900000000000002]
Euler:
[1.85, 2.5118, 3.0082, 3.3608000000000002, 3.5900000000000003, 3.7150000000000003, 3.7538000000000005, 3
 .7232000000000003, 3.6388000000000003, 3.515, 3.365, 3.2008, 3.0332000000000003, 2.8718000000000004, 2
 .7250000000000005, 2.600000000000001, 2.502800000000001, 2.4382000000000015, 2.4098000000000015, 2
 .4200000000000013, 2.470000000000002, 2.5598000000000014, 2.6882000000000006, 2.852800000000002, 3.050000000000003,
  3.2750000000000044, 3.5218000000000047, 3.783200000000004, 4.050800000000003, 4.315000000000003, 4
 .565000000000005, 4.788800000000005, 4.973200000000004, 5.103800000000002, 5.165000000000002, 5.140000000000001,
 5.010799999999998, 4.758199999999995, 4.3617999999999935, 3.799999999999991]
Zmodyfikowany Euler:
[3.215234375, 3.343479922490754, 3.408779842413823, 3.4409954171751567, 3.456961641044619, 3.464934310122422,
 3.468935435148072, 3.4709490917216033, 3.4719640259542204, 3.472475973933426, 3.4727343099980303, 3
 .4728646960693967, 3.472930510538342, 3.47296373315363, 3.4729805040971202, 3.4729889702645185, 3.4729932441125375,
  3.472995401622169, 3.472996490770783, 3.4729970405925306, 3.472997318152524, 3.4729974582698775, 3
 .4729975290036808, 3.4729975647114046, 3.472997582737317, 3.4729975918371268, 3.4729975964308806, 3
 .472997598749892, 3.472997599920572, 3.4729976005115506, 3.472997600809889, 3.472997600960493, 3.472997601036522,
 3.472997601074903, 3.4729976010942782, 3.4729976011040598, 3.4729976011089967, 3.472997601111488, 3
 .4729976011127457, 3.472997601113378]
```

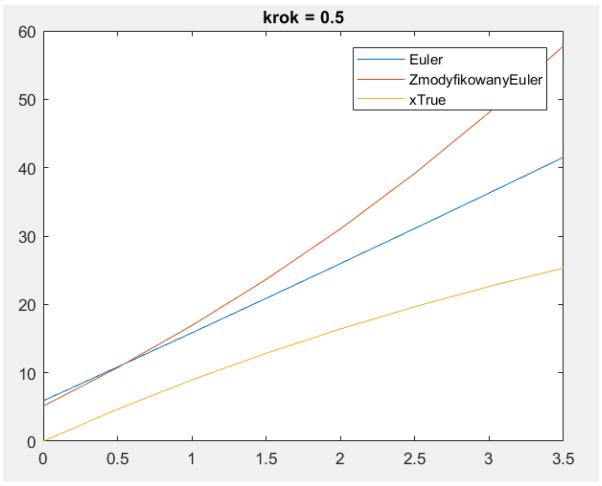
Krok 0.25

```
ListXTrue:
[1.0, 2.560546875, 3.21875, 3.279296875, 3.0, 2.591796875, 2.21875, 1.998046875, 2.0, 2.248046875, 2.71875,
 3.341796875, 4.0, 4.529296875, 4.71875, 4.310546875]
t:
[0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0, 3.25, 3.5, 3.75]
Euler:
[3.125, 4.1796875, 4.4921875, 4.34375, 3.96875, 3.5546875, 3.2421875, 3.125, 3.25, 3.6171875, 4.1796875, 4.84375,
 5.46875, 5.8671875, 5.8046875, 5.0]
Zmodyfikowany Euler:
[3.3070068359375, 3.3252717292089535, 3.3353696450551844, 3.3415641650563863, 3.345584421642535, 3.3482842537463533,
  3.3501376920442567, 3.3514289329091618, 3.3523376047227913, 3.3529815438016612, 3.3534401257077704, 3
 .3537678434047287, 3.35400262170414, 3.3541711154764755, 3.354292191908094, 3.3543792742044225]
```

4.2) skoczek

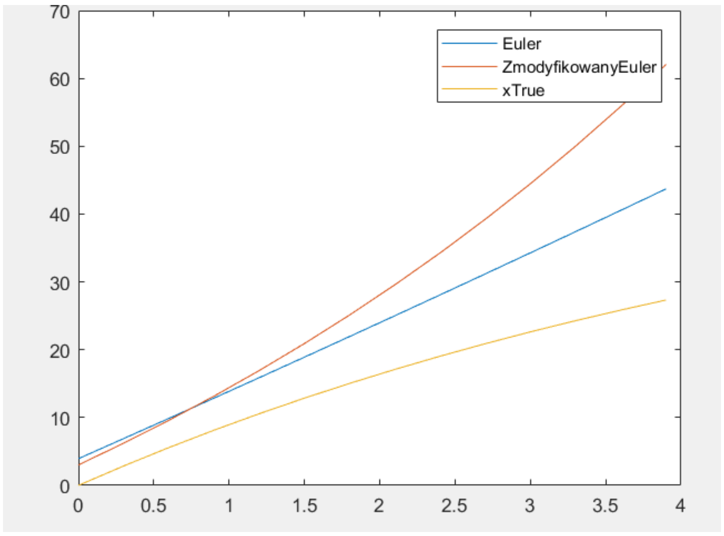
I model

KROK 0.5:



```
ListXTrue:
[0.0, 4.686659617731507, 8.96233935950712, 12.863078637662582, 16.421756516989902, 19.668368850127443, 22
 .630281110551884, 25.332459054283895]
ListXTrue:
[0.0, 4.686659617731507, 8.96233935950712, 12.863078637662582, 16.421756516989902, 19.668368850127443, 22
 .630281110551884, 25.332459054283895]
t:
[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]
Euler:
[5.905, 10.85588839941263, 15.852665198237887, 20.89533039647577, 25.983883994126284, 31.118325991189426, 36
 .2986563876652, 41.5248751835536]
Zmodyfikowany Euler:
[5.130082599118943, 10.752593050518271, 16.91479866130589, 23.66850384653534, 31.070485638435535, 39
 .182970999426395, 48.074159951594304, 57.818798920470925]
```

KROK 0.3:



```
ListXTrue:
[0.0, 2.8634445193312104, 5.573472819055169, 8.138304557796182, 10.565719005358716, 12.863078637662582, 15
 .03735146752031, 17.0951321789872, 19.042662129385484, 20.885848279669972, 22.630281110551877, 24.281251578721346,
 25.843767164597875, 27.32256706028231]
ListXTrue:
[0.0, 2.8634445193312104, 5.573472819055169, 8.138304557796182, 10.565719005358716, 12.863078637662582, 15
 .03735146752031, 17.0951321789872, 19.042662129385484, 20.885848279669972, 22.630281110551877, 24.281251578721346,
 25.843767164597875, 27.32256706028231]
t:
[0.0, 0.3, 0.6, 0.8999999999999999, 1.2, 1.5, 1.8, 2.1, 2.4, 2.6999999999999997, 2.9999999999999996, 3
 .2999999999999994, 3.599999999999999, 3.899999999999999]
Euler:
[3.943, 6.902519823788547, 9.878559471365639, 12.871118942731279, 15.880198237885464, 18.905797356828195, 21
 .947916299559473, 25.006555066079297, 28.081713656387667, 31.17339207048458, 34.28159030837004, 37.40630837004405,
 40.5475462555066, 43.705303964757704]
Zmodyfikowany Euler:
[3.0240297356828196, 6.219165774575487, 9.59508969045774, 13.162030862006594, 16.930797468811267, 20.91280924121133,
  25.12013206319326, 29.565514533195547, 34.262426593605184, 39.22510034599666, 44.46857317578758, 50
 .00873331698294, 55.8623679950735, 62.047214293966086]
```

II model:

Krok 0.5

```
ListXTrue:
[9.81, 9.764079795862795, 9.626154652077961, 9.395937139734958, 9.073106632969766, 8.657319607688251, 8.148214589741697, 7.5454151766236155]
t:
[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]
Euler:
[5.905, 10.787037584513287, 15.600093390192196, 20.2979829098821, 24.83433741913549, 29.162590819912257, 33.235969327280515, 37.007482892249364]
Zmodyfikowany Euler:
[14.709261967853687, 19.56257084707822, 24.323827732993113, 28.946757906397256, 33.38489538622903, 37.591571394740335, 41.51990500360141, 45.12279520034852]
```

Krok 0.3

```
ListXTrue:
[9.81, 9.793473956990127, 9.74386374069383, 9.6611132960991, 9.545160094051765, 9.395937139734958, 9.213373937673087, 8.99739708102971, 8.747930654465538, 8.464896529226747, 8.148214589741697,
 7.797802913732412, 7.413577919206743, 6.995454486952706]
t:
[0.0, 0.3, 0.6, 0.8999999999999999, 1.2, 1.5, 1.8, 2.1, 2.4, 2.6999999999999997, 2.9999999999999996, 3.2999999999999994, 3.599999999999999, 3.899999999999999]
Euler:
[3.943, 6.881041921374742, 9.804198545328024, 12.702523329040746, 15.56604817381925, 18.384781885633192, 21.148708968322303, 23.847788619953686, 26.471953868057643, 29.011110806590217,
 31.455137911011175, 33.79388541545235, 36.01717474055912, 38.114797963559674]
Zmodyfikowany Euler:
[12.751760781269773, 15.683602098767299, 18.595587426302018, 21.477759665044985, 24.320139335984543, 27.112723230198206, 29.845483314322447, 32.50836580222152, 35.09129034469082, 37
 .58414930786152, 39.976807120989456, 42.259099680172994, 44.42083379822126, 46.451786693324564]
```

5.2a)

dx/dt = x*t^2-1,1x

dx/dt = x(t^2-1,1)

dx/x = (t^2-1,1)dt

integral(1/x)dx = integral(t^2-1,1)dt

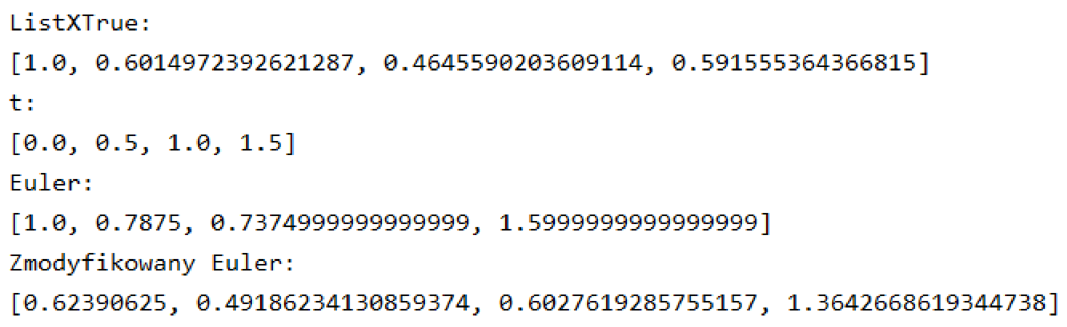
lnx = 1/3*x^3-1,1x+C

x = e^(1/3*x^3-1,1x+C)

x = e^(1/3*x^3-1,1x) + C

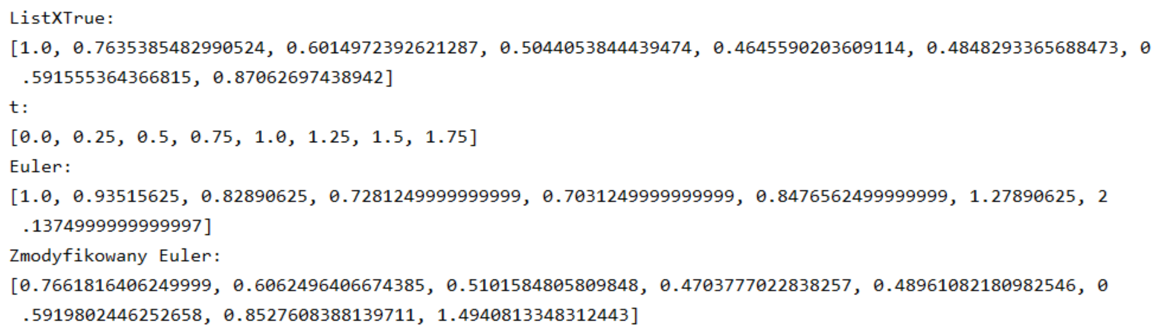
5.2b)

Krok 0.5:



```
ListXTrue:
[1.0, 0.6014972392621287, 0.4645590203609114, 0.591555364366815]
t:
[0.0, 0.5, 1.0, 1.5]
Euler:
[1.0, 0.7875, 0.7374999999999999, 1.5999999999999999]
Zmodyfikowany Euler:
[0.62390625, 0.49186234130859374, 0.6027619285755157, 1.3642668619344738]
```

Krok 0.25:



```
ListXTrue:
[1.0, 0.7635385482990524, 0.6014972392621287, 0.5044053844439474, 0.4645590203609114, 0.4848293365688473, 0
 .591555364366815, 0.87062697438942]
t:
[0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75]
Euler:
[1.0, 0.93515625, 0.82890625, 0.7281249999999999, 0.7031249999999999, 0.8476562499999999, 1.27890625, 2
 .1374999999999997]
Zmodyfikowany Euler:
[0.7661816406249999, 0.6062496406674385, 0.5101584805809848, 0.4703777022838257, 0.48961082180982546, 0
 .5919802446252658, 0.8527608388139711, 1.4940813348312443]
```