METODY NUMERYCZNE

LISTA 7

WIOLETTA ŁUPKOWSKA, 244831

CZWARTEK, 9:15

1.1)

```
double xEulerPrevious=1;
double xEulerNext=0;
```

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;
        double xEulerPrevious=1;
        double xEulerNext=0;
        do{
            listT.add(t);
            xTrue=licz.xTrue(t);
            listXTrue.add(xTrue);
            xEulerNext= Math.abs(xEulerPrevious + licz.ft(xEulerNext,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 wartoscSrodkowa;
        double t=przedzialL;
        double xEulerNext=0;
        double xTrue;
```

```
        double xEulerPrevious=1; //==0
        do{
            tSrodkowy= (t+przedzialU)/2;
            listT.add(t);
            xTrue=licz.xTrue(t);
            listXTrue.add(xTrue);
            xEulerNext = Math.abs(xEulerPrevious + licz.ft(xEulerNext,
tSrodkowy)*krok); //x(0.5)
            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)

```
 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 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);
    }
}
```

Indefinite integral:

$$\int \left(9.81 - \frac{12.5\left(v^2 + \frac{8.3\,v^{3.2}}{46^{2.2}}\right)}{68.1}\right) dv =$$

$$-0.0000797121\,v^{21/5} - 0.0611845\,v^3 + 9.81\,v + \text{constant}$$

4.1)

```
double xEulerPrevious=licz.xTrue(0);


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;
    }
}
```

5.1a)

```
(double xEulerNext=1;
double xEulerPrevious=0;)
```

```
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(x,2)+8.3/(Math.pow(46,2.2))*Math.pow(x,3.2));
        double ft=x*Math.pow(t,2)-1.1*x;
        return ft;
    }
}
```

3.2)

Krok 0.5

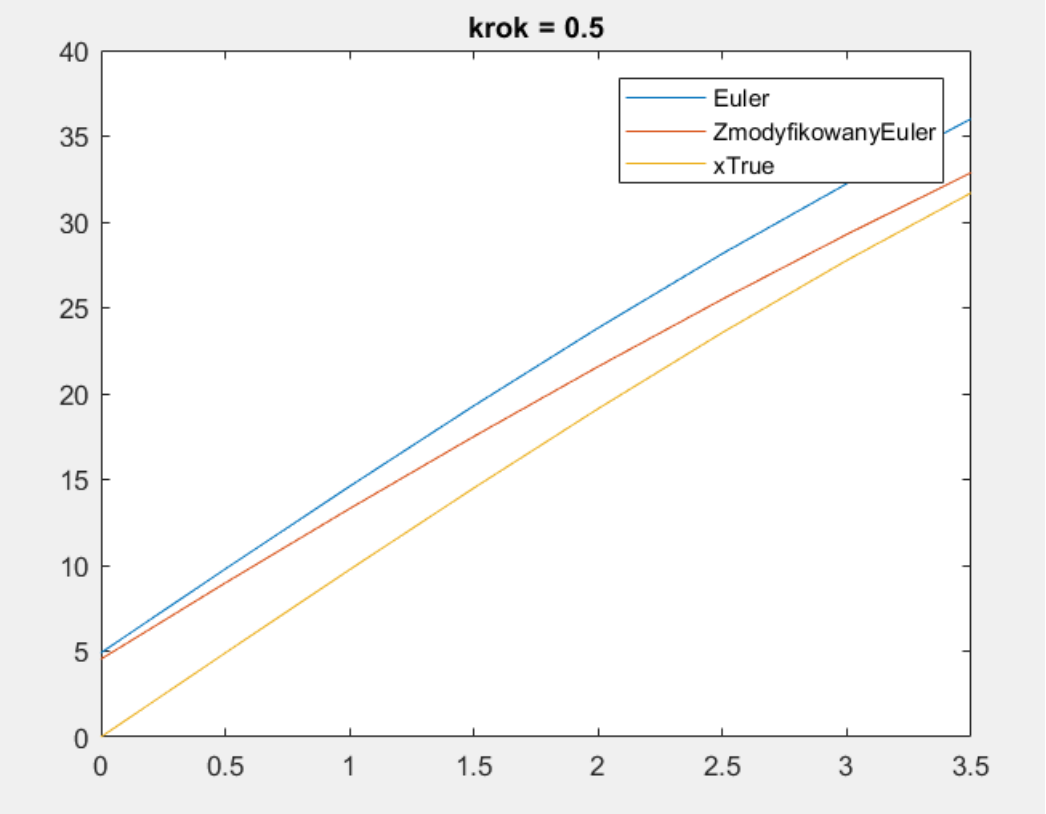
```
ListXTrue:
[1.0, 3.21875, 3.0, 2.21875, 2.0, 2.71875, 4.0, 4.71875]
t:
[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]
Euler:
[5.25, 5.875, 5.125, 4.5, 4.75, 5.875, 7.125, 7.0]
Zmodyfikowany Euler:
[1.25, 1.984375, 3.109375, 4.4375, 5.6875, 6.484375, 6.359375, 4.75]
```

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:
[1.125, 1.3740234375, 1.7412109375, 2.21484375, 2.77734375, 3.4052734375, 4.0693359375, 4.734375, 5.359375, 5.8974609375,
 6.2958984375, 6.49609375, 6.43359375, 6.0380859375, 5.2333984375, 3.9375]
```

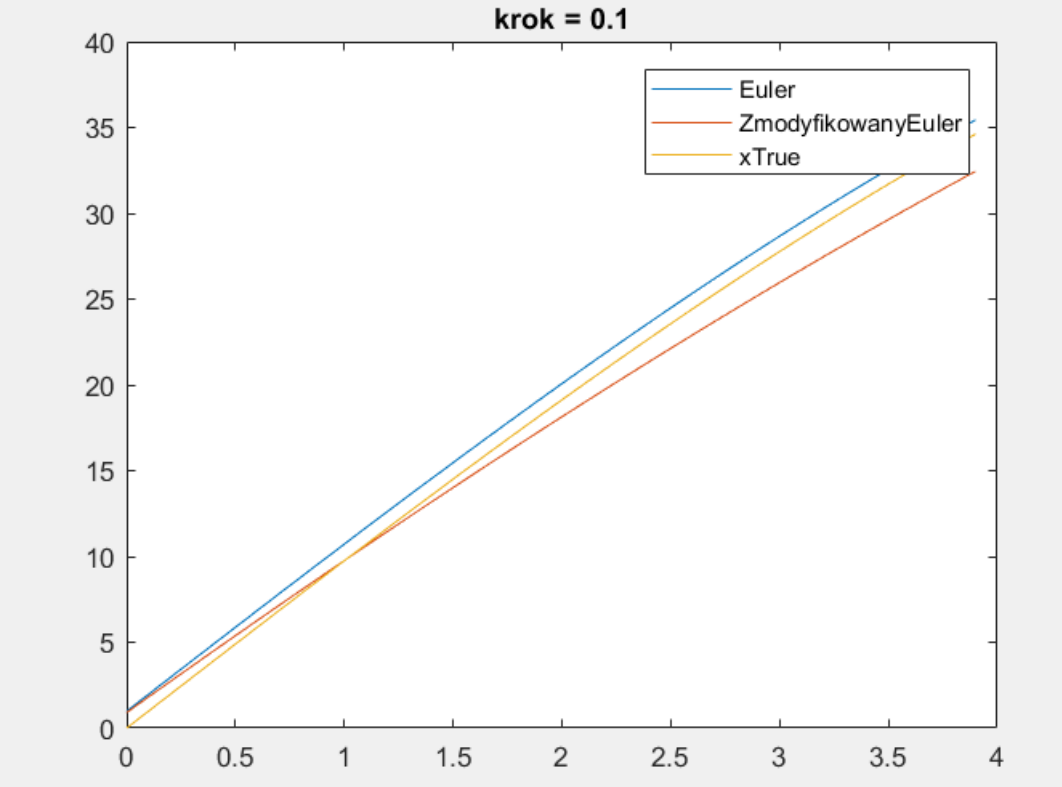
4.2) skoczek

KROK 0.5:




```
ListXTrue:
[0.0, 4.8973469554937505, 9.748735787900001, 14.50809876999375, 19.1292486064, 23.565878433593753, 27.7715618199, 31
 .69975276549375]
t:
[0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5]
Euler:
[4.905, 9.787037584513287, 14.600093390192196, 19.2979829098821, 23.83433741913549, 28.162590819912257, 32.235969327280515,
 36.007482892249364]
Zmodyfikowany Euler:
[4.536354509253389, 8.97449198908517, 13.302745389861936, 17.509421398373238, 21.5827999057415, 25.511133514602577,
 29.282647079571426, 32.88553727631854]
```

KROK 0.1:



skoczek krok 0.1:

```
ListXTrue:
[0.0, 0.9809388075287901, 1.9615103964606402, 2.9413473728319905, 3.92008215137024, 4.8973469554937505, 5.87277381731184, 6.84599457762479, 7.81664088592384, 8.784344200391189, 9.7487357879,
 10.709446724014388, 11.66610789298944, 12.618349987771191, 13.565803509966641, 14.508098769993753, 15.444865886781443, 16.375734788069593, 17.300335210259046, 18.218296698441595, 19
 .129248606400004, 20.032820096607995, 20.928640140230247, 21.8163375171224, 22.69554081583105, 23.56587843359376, 24.42697857633905, 25.2784692586864, 26.11997830394625, 26.95113334412,
 27.77156181990001, 28.580890980669604, 29.378747884503056, 30.164759398165607, 30.938552197113456, 31.69975276549377, 32.447987396144654, 33.1828821905952, 33.90406305906546, 34.61115572046641]
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
 .5000000000000001, 2.6000000000000001, 2.7000000000000001, 2.8000000000000001, 2.9000000000000012, 3.0000000000000013, 3.1000000000000014, 3.2000000000000015, 3.3000000000000016, 3
 .4000000000000017, 3.5000000000000018, 3.6000000000000002, 3.7000000000000002, 3.8000000000000002, 3.9000000000000002]
Euler:
[0.9810000000000001, 1.9618164252784662, 2.9420820167681025, 3.9214293238930162, 4.8994906824502, 5.875898199352857, 6.850283740670618, 7.822278921683044, 8.79151509830948, 9.757623359547054,
 10.720234520682837, 11.678979117121331, 12.633487398714166, 13.583389324508271, 14.528314557848766, 15.467892461786747, 16.40175209475227, 17.329522206460325, 18.250831234023362, 19
 .165307298248216, 20.072578200098892, 20.972271417309354, 21.86401410113267, 22.747433073214818, 23.622154822582804, 24.48780550273816, 25.344010928847815, 26.19039657502534, 27.02658757169622,
  27.852208703041573, 28.666884404515223, 29.470238760429552, 30.261895501605988, 31.04147800308638, 31.808609281901784, 32.56291199489556, 33.30400843659781, 34.03152053714868, 34
 .74506986026782, 35.444277601268006]
Zmodyfikowany Euler:
[0.9072709018506778, 1.8107995469948803, 2.7104927642053416, 3.6062573413441568, 4.4980000251674745, 5.385627521133831, 6.269046493215978, 7.148163563716066, 8.022885313084052, 8
 .893118279739195, 9.758768959894548, 10.619743807384314, 11.475949233493969, 12.327291606793063, 13.173677252970588, 14.01501245467285, 14.85120345134373, 15.682156439067299, 16
 .507777570412653, 17.327972954280966, 18.14264865575462, 18.95171069594841, 19.75506505186274, 20.552617656238716, 21.344274397415152, 22.129941119187368, 22.909523620667763, 23
 .682927656148106, 24.45005893496351, 25.210823121358025, 25.965125834351795, 26.712872647609796, 27.45396908931205, 28.188320642025314, 28.915832742576182, 29.636410781925605, 30
 .349960105044747, 31.056386010792178, 31.755593751792365, 32.44748853431541]
```

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, 0.5769498103804866, 0.33287108369807955, 0.19204990862075408]
ListXTrue:
[1.0, 0.5769498103804866, 0.33287108369807955, 0.19204990862075408]
t:
[0.0, 0.5, 1.0, 1.5]
Euler:
[0.55, 0.31625000000000003, 0.3004375, 0.47318906250000003]
Zmodyfikowany Euler:
[0.050000000000000044, 0.06156250000000005, 0.09696093750000007, 0.19210385742187513]
```

Krok 0.25:

```
ListXTrue:
[1.0, 0.7595721232249685, 0.5769498103804866, 0.4382349924649492, 0.33287108369807955, 0.25283959580474646,
 0.19204990862075408, 0.14587575685622733]
ListXTrue:
[1.0, 0.7595721232249685, 0.5769498103804866, 0.4382349924649492, 0.33287108369807955, 0.25283959580474646,
 0.19204990862075408, 0.14587575685622733]
t:
[0.0, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5, 1.75]
Euler:
[0.275, 0.203671875, 0.16039160156249999, 0.13883898010253903, 0.13536800559997555, 0.15101993124747273, 0
 .19443816148112114, 0.2898343844577962]
Zmodyfikowany Euler:
[0.025000000000000022, 0.0260351562500002, 0.029045471191406273, 0.03478649010658267, 0.04478760601222519,
 0.06203783239037129, 0.0924751439068972, 0.14832146128192186]
```