**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:

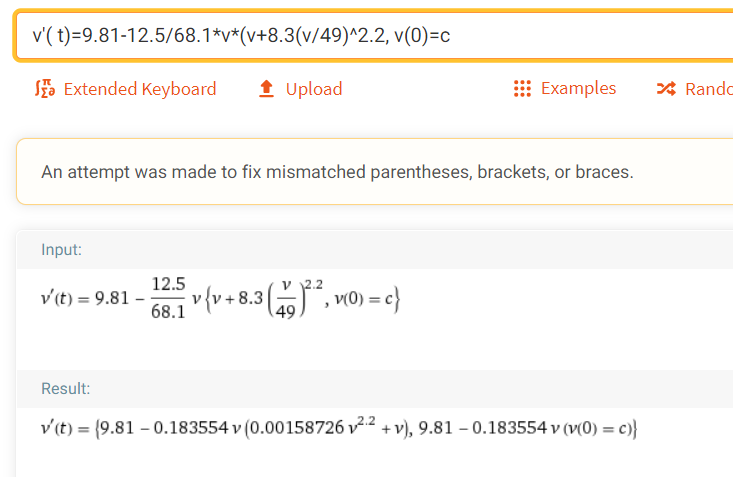
**import** java.util.ArrayList;  
  
**public class** ZmodyfikowanyEuler {  
  
 Licz **licz**= **new** Licz();  
  
  
 **public** ArrayList liczEuler(**double** przedzialL, **double** przedzialU, **double** krok){  
 ArrayList<Double> listXEuler = **new** ArrayList();  
 ArrayList<Double> listXTrue = **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;  
 }  
}

**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)**

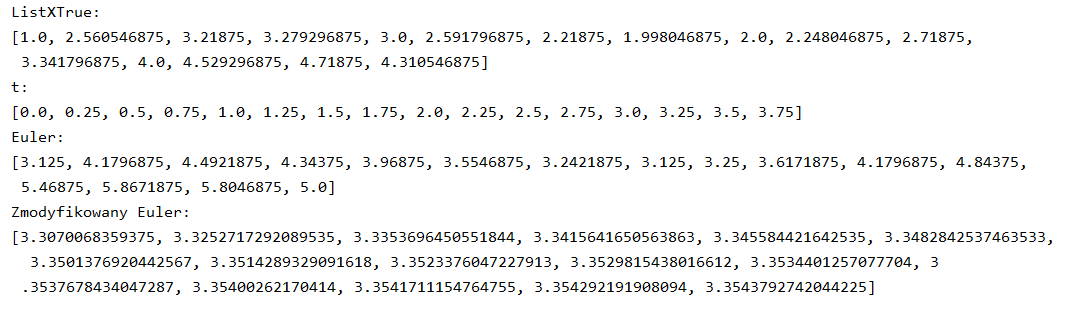
**public class** Licz **implements** Function {  
  
 @Override  
 **public double** xTrue (**double** x, **double** t){ *//calka, rozwiazanie  
 //double xTrue= -0.5\*Math.pow(x,4) + 4\*Math.pow(x,3)- 10\*Math.pow(x,2) + 8.5\*x + 1; //rownanie z listy 6  
 //double xTrue= (0-53.4449)\*Math.exp(-0.183554\*t)+53.4449; //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**

****

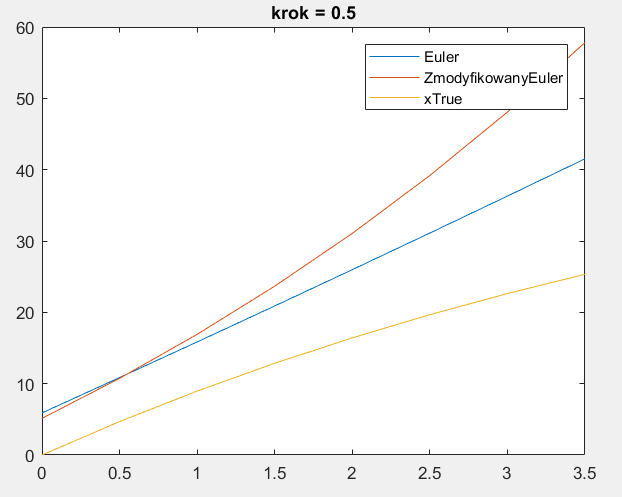
**Krok 0.25**

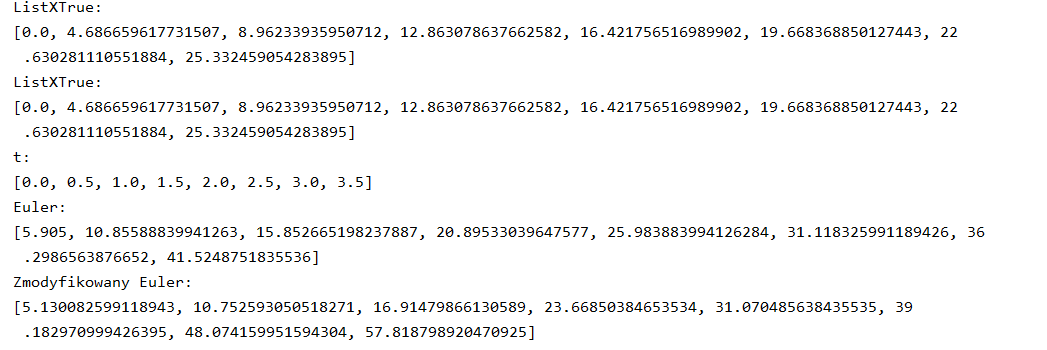
****

**4.2) skoczek**

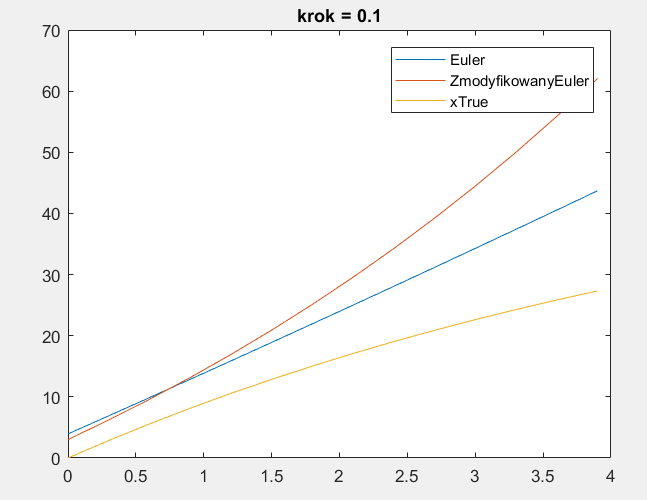
***I model***

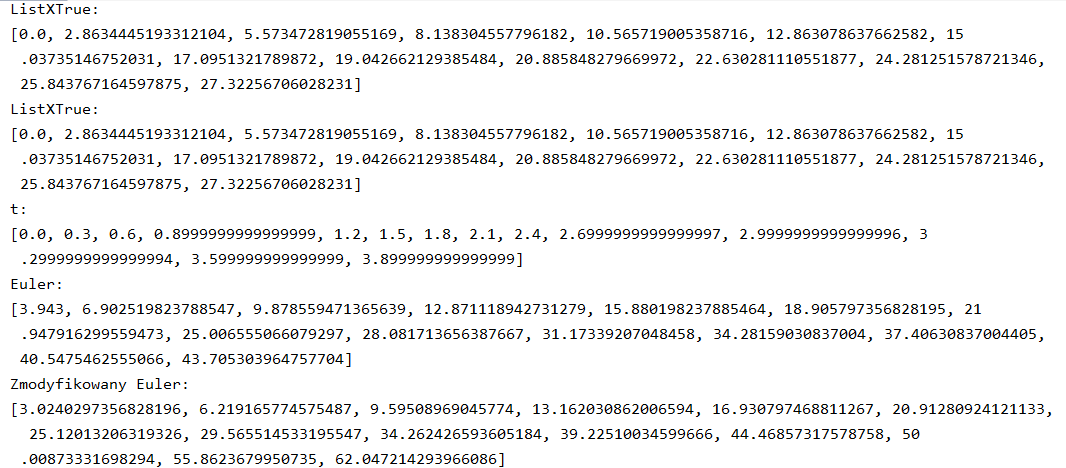
KROK 0.5:

****

****

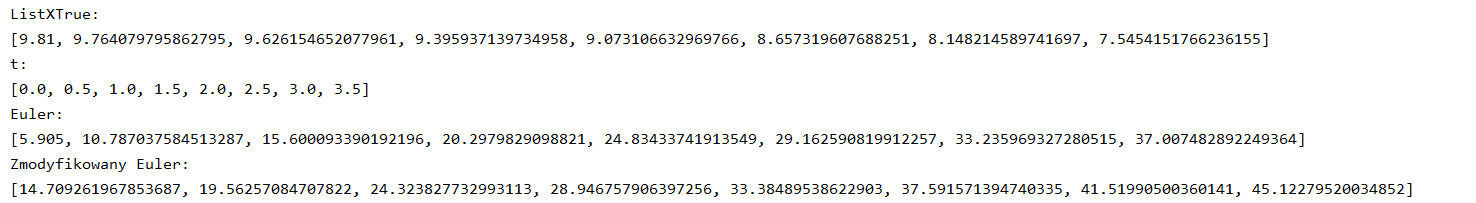
KROK 0.3:



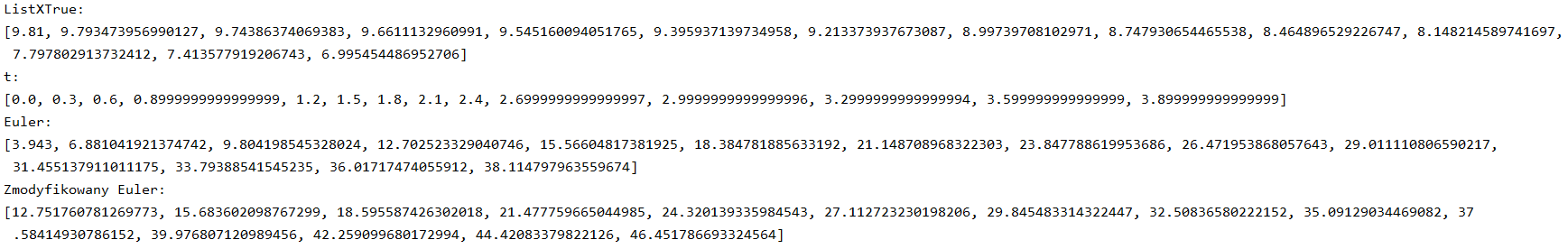


**II model:**

Krok 0.5

****

Krok 0.3

****

**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

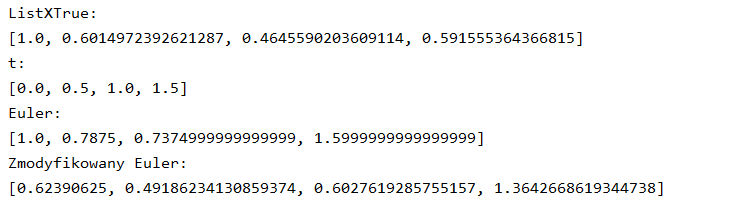
lnx = 1/3\*x^3-1,1x+C

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

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

**5.2b)**

*Krok 0.5:*



*Krok 0.25:*

