METODY NUMERYCZNE

LISTA 8

WIOLETTA ŁUPKOWSKA, 244831

CZWARTEK, 9:15

1.1

```
public interface FirstOrderODE {

    double f (double t, double x);

}
```

```
public interface ODESingleStep {

    double singleStep(FirstOrderODE ode, double t, double x, double h);
}
```

```
public interface StepHandler {

    void handler (double t, double x);

}
```

```
public class FirstOrderODESolver {



    private ODESingleStep odeSingleStep;
    private StepHandler stepHandler;

    public FirstOrderODESolver(ODESingleStep odeSingleStep) {
this.odeSingleStep = odeSingleStep;
    }


    public void addStepHandler(StepHandler stepHandler){
        this.stepHandler = stepHandler;

    }

    public double integrate(FirstOrderODE ode, double tStart, double xStart,
double tStop, int n){
        double h = (tStop - tStart)/n;
        double x= xStart;
        double t = tStart;

        for(int i = 0; i<n; i++){                           //i<liczba krokow
            if(stepHandler != null)
                stepHandler.handler(t,x);
            //wyswietla t, x
            x = odeSingleStep.singleStep(ode, t, x, h);                 t += h;

        }
        if(stepHandler != null)
```

```
            stepHandler.handler(t,x);
        return x;
    }


}
```

```
import java.util.ArrayList;
import java.util.List;

public class SaveAllStepHandler implements StepHandler {

    private List<Double> tList = new ArrayList<>();
    private List<Double> xList = new ArrayList<>();


    @Override
    public void handler(double t, double x) {
        tList.add(t);
        xList.add(x);

    }

    public void clear(){
        tList.clear();
        xList.clear();
    }

    public List<Double> getT(){

        List<Double> export = new ArrayList<>();
        for (Double d : tList) export.add(d);
        return export;
    }

    public List<Double> getX(){

        List<Double> export = new ArrayList<>();
        for (Double d : xList) export.add(d);
        return export;
    }
}
```

```
public class EulerSingleStep implements ODESingleStep {
    @Override
    public double singleStep(FirstOrderODE ode, double t, double x, double h) {
        return x+ ode.f(t,x)*h;
    }
}
```

```
public class ModifiedEulerSingleStep implements ODESingleStep{


    @Override
    public double singleStep(FirstOrderODE ode, double t, double x, double h) {
        double temp_x = x+ode.f(t,x)*h/2;
        return x = x+ ode.f(t+h/2,temp_x)*h;
    }
}
```

```
public class RK4SingleStep implements ODESingleStep {
    double k1;
```

```
    double k2;
    double k3;
    double k4;
    double xEnd;

    @Override
    public double singleStep(FirstOrderODE ode, double t, double x, double h) {

        k1 = ode.f(t,x);
        k2 = ode.f(t+h/2, x+k1*(h/2));
        k3 = ode.f(t+h/2, x+k2*(h/2));
        k4= ode.f(t+h,x+k3*h);
        double nachylenie = (1./6.)* (k1+2*k2+2*k3+k4);
        xEnd = x + h*nachylenie;
        return xEnd;
    }
}
```

```
import java.util.ArrayList;
import java.util.List;

public class Main {


    public static double trueValue= 75.33896;
    public static int n=400;



    public static ArrayList bledy (double xEndEuler, double xEndModifiedEuler,
double xEndRK4){

        double bladEuler = Math.abs((xEndEuler-trueValue)/trueValue*100);
        double bladModifiedEuler = Math.abs((xEndModifiedEuler-
trueValue)/trueValue*100);
        double bladRK4 = Math.abs((xEndRK4-trueValue)/trueValue*100);
        ArrayList bledy = new ArrayList();
        bledy.add(bladEuler);
        bledy.add(bladModifiedEuler);
        bledy.add(bladRK4);

        return bledy;
    }
```

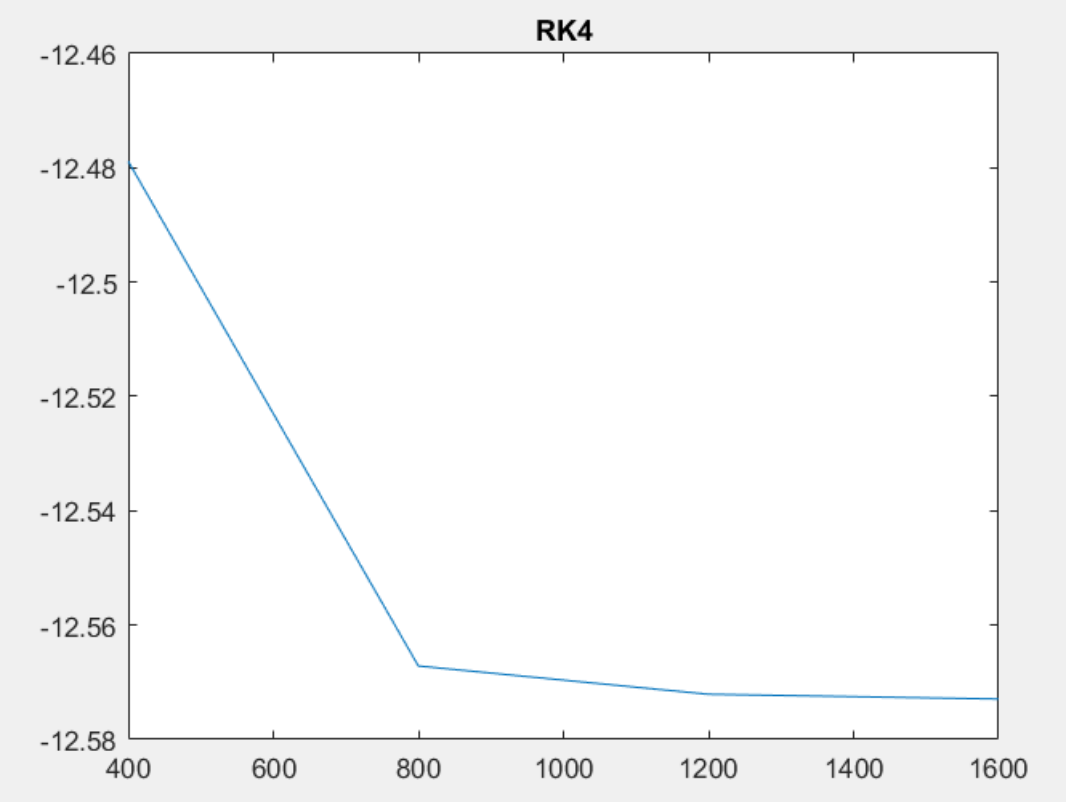
```
    public static void main(String[] args) {

        ArrayList bledy = new ArrayList();
        ArrayList bledyLn = new ArrayList();


       FirstOrderODESolver solverEuler = new FirstOrderODESolver((new
EulerSingleStep()));
       FirstOrderODESolver solverModifiedEuler = new FirstOrderODESolver((new
ModifiedEulerSingleStep()));
       FirstOrderODESolver solverRK4 = new FirstOrderODESolver((new
RK4SingleStep()));
```

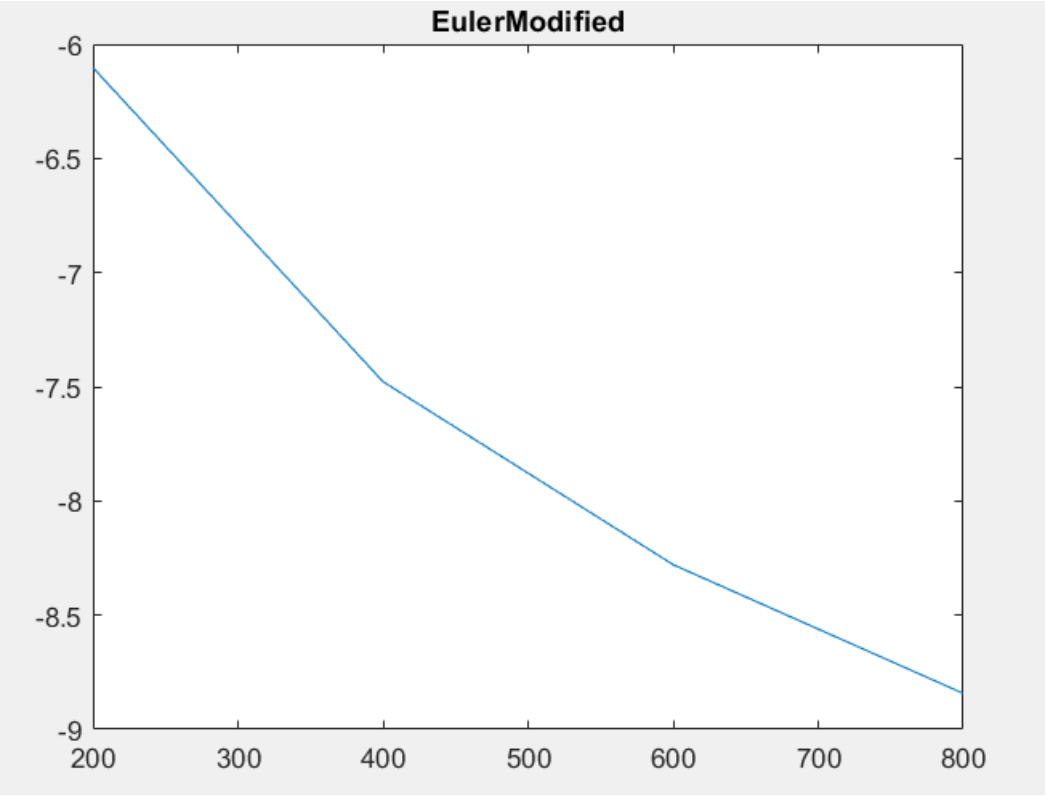
```
        SaveAllStepHandler saveAllStepHandlerEuler = new SaveAllStepHandler();
        SaveAllStepHandler saveAllStepHandlerModifiedEuler = new
SaveAllStepHandler();
        SaveAllStepHandler saveAllStepHandlerRK4 = new SaveAllStepHandler();
        solverEuler.addStepHandler(saveAllStepHandlerEuler);
        solverModifiedEuler.addStepHandler(saveAllStepHandlerModifiedEuler);
        solverRK4.addStepHandler(saveAllStepHandlerRK4);


       double xEndEuler = solverEuler.integrate(  (t,x) -> 4*Math.exp(0.8*t)-0.5*x
, 0,2,4,n);
       double xEndModifiedEuler = solverModifiedEuler.integrate(  (t,x) ->
4*Math.exp(0.8*t)-0.5*x , 0,2,4,n);
       double xEndRK4 = solverRK4.integrate(  (t,x) -> 4*Math.exp(0.8*t)-0.5*x ,
0,2,4,n);


        System.out.println("solution Euler:");
        System.out.println(xEndEuler);

        System.out.println("solution modified Euler:");
        System.out.println(xEndModifiedEuler);

        System.out.println("solution Rungeg-Kutt:");
        System.out.println(xEndRK4);

        bledy= bledy(xEndEuler, xEndModifiedEuler, xEndRK4);
        System.out.println(bledy.get(1));
        for (int i=0; i<3;i++) {
            bledyLn.add(Math.log((double)bledy.get(i)));
        }

        System.out.println("błędy każdej z metod: ");
        System.out.println(bledy);
        System.out.println("logarytmy błędów: ");
        System.out.println(bledyLn);


    }

}
```

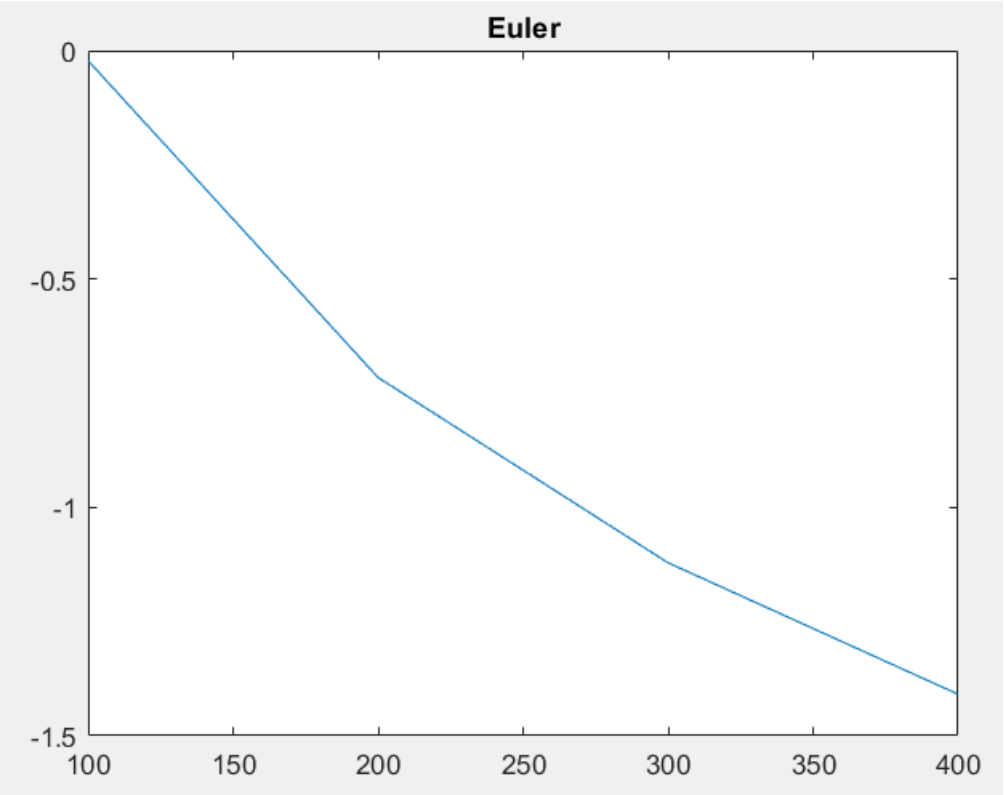
```
%n=[100,200,300,400]

bledy=[-0.022164330754621743, -6.102754049304768, -
12.478868104406045;-0.7157550732455098, -
7.477690583438678, -12.567145014945977;-
1.1213735086210366, -8.278797479284247, -
12.572089827430023;-1.4091345986776485, -
8.842523432525018, -12.572924365793591];
Euler=bledy(:,1);
ModifiedEuler=bledy(:,2);
RK4=bledy(:,3);
```

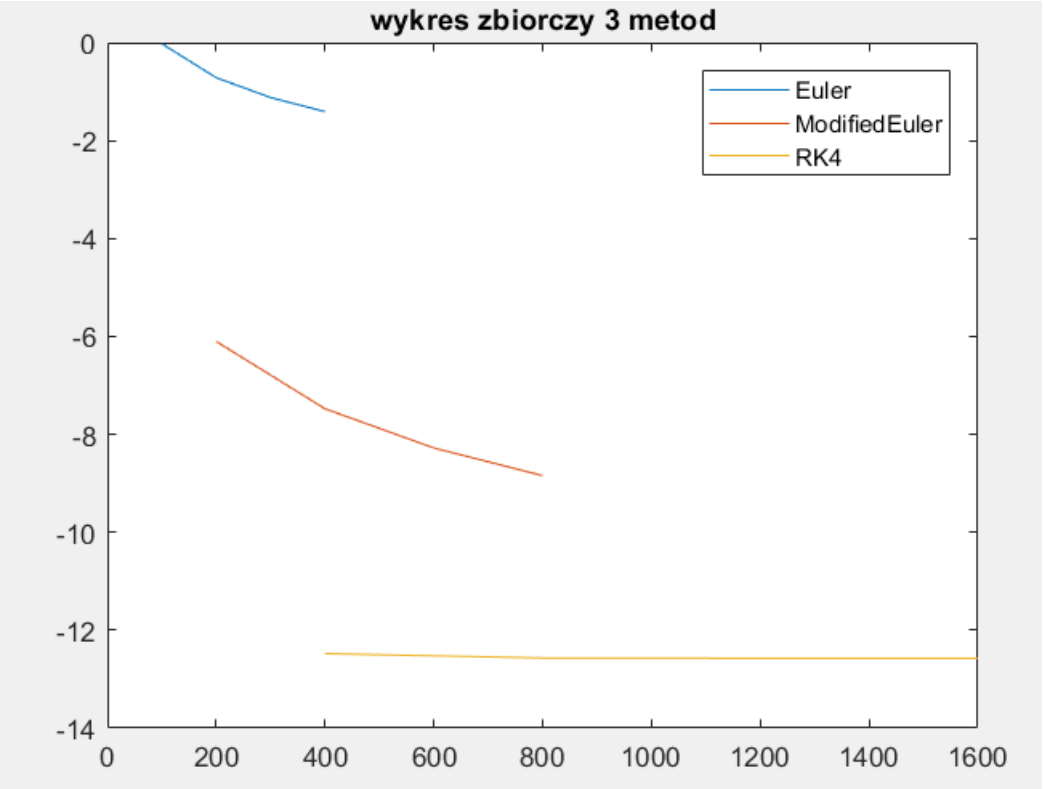
```
EulerKroki=[100;200;300;400];
ModifiedEulerKroki=EulerKroki.*2;
RK4Kroki=EulerKroki.*4;

figure;
plot(EulerKroki, Euler,
ModifiedEulerKroki,ModifiedEuler,RK4Kroki,RK4);
legend("Euler", "ModifiedEuler", "RK4");
title("wykres zbiorczy 3 metod");
figure;
plot(EulerKroki, Euler);
title("Euler");
figure;
plot(ModifiedEulerKroki,ModifiedEuler);
title("EulerModified");
figure;
plot(RK4Kroki,RK4);
title("RK4")
```

1.2




EulerModified
-6
-6.5
-7
-7.5
-8
-8.5
-9
200
300
400
500
600
700
800


Euler
0
-0.5
-1
-1.5
100
150
200
250
300
350
400


wykres zbiorczy 3 metod
Euler
ModifiedEuler
RK4
0
-2
-4
-6
-8
-10
-12
-14
0
200
400
600
800
1000
1200
1400
1600