METODY NUMERYCZNE

LISTA 12

Wioletta Łupkowska

Czwartek, 9:15, 244831

**public interface** CalculateAcceleration {  
  
 **public double** a (**double** x);  
  
}

**public interface** ODEUpdate {  
   
 **void** update(**double** t, **double** x, **double** v);  
}

**public class** TennisToss **implements** CalculateAcceleration{  
  
 @Override  
 **public double** a(**double** x) {  
 **return** -9.81;  
 }  
}

**import** java.io.FileNotFoundException;  
**import** java.io.PrintWriter;  
**import** java.util.ArrayList;  
  
**public class** Analyzer **implements** ODEUpdate{  
  
 **private** ArrayList<Double> **tValues** = **new** ArrayList<>();  
 **private** ArrayList<Double> **xValues** = **new** ArrayList<>();  
 **private** ArrayList<Double> **vValues** = **new** ArrayList<>();  
  
 @Override  
 **public void** update(**double** t, **double** x, **double** v) {  
  
 **tValues**.add(t);  
 **xValues**.add(x);  
 **vValues**.add(v);  
 }  
  
 **public** ArrayList gettValues() {  
 **return tValues**;  
 }  
  
 **public** ArrayList getxValues() {  
 **return xValues**;  
 }  
  
 **public** ArrayList getvValues() {  
 **return vValues**;  
 }  
  
 **void** saveToFile(String fileName) **throws** FileNotFoundException{  
 PrintWriter printwriter = **new** PrintWriter(fileName);  
 printwriter.println((**"t \t x \t v"**));  
  
 **for**(**int** i = 0;i<**tValues**.size();i++)  
 printwriter.println(**tValues**.get(i).toString() + **'\t'** +  
 **xValues**.get(i).toString() + **'\t'** +  
 **vValues**.get(i).toString());  
  
 printwriter.close();  
  
 }  
  
}

**public class** EulerCromer {  
  
  
  
 **private double dt**;  
  
 **public** EulerCromer(**double** dt){  
 **this**.**dt** = dt;  
 }  
  
  
 **public void** integrate(CalculateAcceleration calculateAcceleration,  
 ODEUpdate odeUpdate, **double** tStart, **double** tStop,  
 **double** x0, **double** v0){  
  
 **int** nSteps = (**int**) ((tStop - tStart)/**dt**);  
  
 **double** t = tStart;  
 **double** x = x0;  
 **double** v = v0;  
 odeUpdate.update(t,x,v);  
  
  
  
 **for**(**int** i = 0; i<nSteps;i++) {  
  
 **double** a = calculateAcceleration.a(x);**double** vNew = v +**dt**\*a;  
 **double** xNew = x + **dt**\*vNew;  
  
 t+=**dt**;  
 x = xNew;  
 v = vNew;  
  
 *//zeby zapamietac wszystkie wartosci to:* odeUpdate.update(t, x, v);  
 }  
  
 *// 𝑣𝑛+1 = 𝑣𝑛 + 𝑎𝑛∆𝑡  
//// 𝑥𝑛+1 = 𝑥𝑛 + 𝑣𝑛+1∆𝑡.* }  
}

**public class** Car **implements** CalculateAcceleration{  
  
 **private double a**;  
 **private double m**;  
 **private double v**;  
 **private double c**;  
 **private double u**;  
  
 **public** Car( **double** m, **double** v, **double** c, **double** u) {  
  
 **this**.**m** = m;  
 **this**.**v** = v;  
 **this**.**c** = c;  
 **this**.**u** = u;  
 }  
  
 **public** Car() {  
 }  
  
 @Override  
 **public double** a(**double** v) {  
 **return** -**u**\*9.81-(**c**\*Math.*pow*(**this**.**v**,2))/**m**;  
 }  
  
  
}

**public class** VerletIntegrator {  
  
 **private double dt**;  
  
 **public** VerletIntegrator(**double** dt){  
 **this**.**dt** = dt;  
 }  
  
  
 **public void** integrate(CalculateAcceleration calculateAcceleration,  
 ODEUpdate odeUpdate, **double** tStart, **double** tStop,  
 **double** x0, **double** v0){  
  
 **int** nSteps = (**int**) ((tStop - tStart)/**dt**);  
  
 **double** t = tStart;  
 **double** x = x0;  
 **double** v = v0;  
 odeUpdate.update(t,x,v);  
  
 **double** a = calculateAcceleration.a(x);  
  
 **for**(**int** i = 0; i<nSteps;i++) {  
  
 **double** vHalf = v + **dt** \* a / 2;  
 **double** xNew = x + **dt** \* vHalf;  
 **double** aNew = calculateAcceleration.a(xNew);  
 **double** vNew = v + **dt** \* (a + aNew) / 2;  
  
 a = aNew;  
 t += **dt**;  
 x = xNew;  
 v = vNew;  
odeUpdate.update(t, x, v);  
  
}  
}

**public class** CarBezOporu **implements** CalculateAcceleration{  
  
 **private double a**;  
 **private double m**;  
 **private double v**;  
 **private double c**;  
  
 **public** CarBezOporu(**double** m, **double** v, **double** c) {  
  
 **this**.**m** = m;  
 **this**.**v** = v;  
 **this**.**c** = c;  
 }  
  
 **public** CarBezOporu() {  
 }  
  
 @Override  
 **public double** a(**double** v) {  
 **return** -(**c**\*Math.*pow*(**this**.**v**,2))/**m**;  
 }  
  
  
}

**import** java.io.FileNotFoundException;  
**import** java.util.ArrayList;  
  
**public class** Tester {  
  
 **public static void** main(String[] args) {  
  
 VerletIntegrator vi = **new** VerletIntegrator(0.01);  
 EulerCromer ec = **new** EulerCromer(0.01);  
 EulerCromer ecBez = **new** EulerCromer(0.01);  
 Analyzer analyzerVi = **new** Analyzer();  
 Analyzer analyzerEc = **new** Analyzer();  
 Analyzer analyzerEcCar = **new** Analyzer();  
 Analyzer analyzerEcCarBezOporu = **new** Analyzer();  
  
 TennisToss toss = **new** TennisToss();  
 Car car = **new** Car(1200,300,0.86,0.7);  
 CarBezOporu carBezOporu = **new** CarBezOporu(1200,300,0.86);  
  
 ArrayList<Double> xListVi = **new** ArrayList<>();  
 ArrayList<Double> xListEc = **new** ArrayList<>();  
 ArrayList<Double> xListEcCar = **new** ArrayList<>();  
 ArrayList<Double> vListEcCar = **new** ArrayList<>();  
 ArrayList<Double> tListEcCar= **new** ArrayList<>();  
 ArrayList<Double> xListEcCarBezOporu= **new** ArrayList<>();  
 ArrayList<Double> vListEcCarBezOporu= **new** ArrayList<>();  
 ArrayList<Double> tListEcCarBezOporu= **new** ArrayList<>();  
  
 vi.integrate(toss,analyzerVi,0,5,1.5,5.2);  
 ec.integrate(toss,analyzerEc,0,5,1.5,5.2);  
 ec.integrate(car,analyzerEcCar,0,5,1.5,300000./3600);  
 ecBez.integrate(carBezOporu,analyzerEcCarBezOporu,0,5,1.5,300000./3600);  
  
 **try**{  
 analyzerVi.saveToFile(**"testVerlet.txt"**);  
 analyzerEc.saveToFile(**"testEuler.txt"**);  
 analyzerEcCar.saveToFile(**"testEulerCar.txt"**);  
 analyzerEcCarBezOporu.saveToFile(**"testEulerCarBezOporu.txt"**);  
  
 }**catch** (FileNotFoundException e){  
 e.printStackTrace();  
 }  
  
 xListVi =analyzerVi.getxValues();  
 xListEc =analyzerEc.getxValues();  
 xListEcCar =analyzerEcCar.getxValues();  
 vListEcCar =analyzerEcCar.getvValues();  
 tListEcCar =analyzerEcCar.gettValues();  
 xListEcCarBezOporu =analyzerEcCarBezOporu.getxValues();  
 vListEcCarBezOporu =analyzerEcCarBezOporu.getvValues();  
 tListEcCarBezOporu =analyzerEcCarBezOporu.gettValues();  
 **double** xMaxVi=0;  
 **double** xMaxEc=0;  
 **double** tCarStop=0;  
 **double** xCarStop=0;  
 **double** tCarStopBezOporu=0;  
 **double** xCarStopBezOporu=0;  
  
 **for**(**int** i =0; i<xListVi.size()-1;i++) {  
 **if** (xListVi.get(i + 1) > xListVi.get(i)) {  
 xMaxVi = xListVi.get(i);  
 }  
 **if** (xListEc.get(i + 1) > xListEc.get(i)) {  
 xMaxEc = xListEc.get(i);  
 }  
 }  
  
  
 **for**(**int** i =0; i<xListEcCar.size()-1;i++){  
 **if**( vListEcCar.get(i)<0){  
 xCarStop=xListEcCar.get(i);  
 tCarStop=tListEcCar.get(i);  
 System.***out***.println(i);  
 **break**;  
 }  
 }  
  
 **for**(**int** i =0; i<xListEcCarBezOporu.size()-1;i++){  
 **if**( vListEcCarBezOporu.get(i)<0){  
 xCarStopBezOporu=xListEcCarBezOporu.get(i);  
 tCarStopBezOporu=tListEcCarBezOporu.get(i);  
 System.***out***.println(i);  
 **break**;  
 }  
 }  
  
 *//System.out.println(xListEcCar);* System.***out***.println(**"Verlet, maksymalna wysokość, na jaką wzniesie sie piłka: "**);  
 System.***out***.println(xMaxVi);  
 System.***out***.println(**"Euler, maksymalna wysokość, na jaką wzniesie sie piłka: "**);  
 System.***out***.println(xMaxEc);  
  
 System.***out***.println(**"Czas, po ktorym samochod sie zatrzyma"**);  
 System.***out***.println(tCarStop);  
 System.***out***.println(**"Droga, po której samochod sie zatrzyma"**);  
 System.***out***.println(xCarStop);  
  
 System.***out***.println(**"Czas, po ktorym samochod sie zatrzyma bez oporu"**);  
 System.***out***.println(tCarStopBezOporu);  
 System.***out***.println(**"Droga, po której samochod sie zatrzyma bez oporu "**);  
 System.***out***.println(xCarStopBezOporu);  
  
  
 }  
}

figure;

plot(t(1:117),x(1:117));

xlabel("czas");

ylabel("droga");

figure;

plot(t(1:117),v(1:117));

xlabel("czas");

ylabel("predkosc");

figure;

plot(t1(1:130),x1(1:130));

xlabel("czas");

ylabel("droga bez oporu");

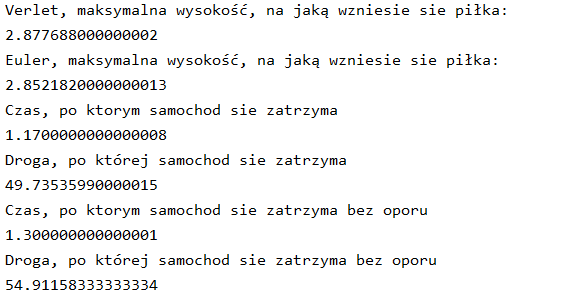
figure;

plot(t1(1:130),v1(1:130));

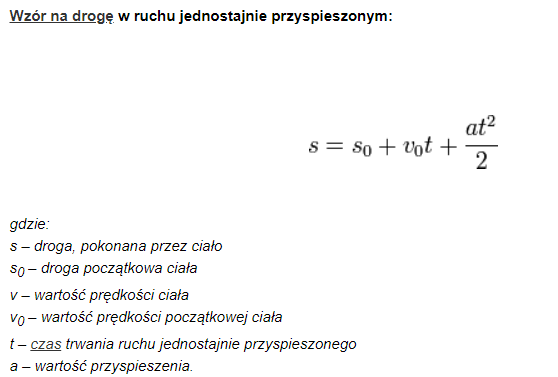
xlabel("czas");

ylabel("predkosc bez oporu");

* *WYNIKI:*

**

* *Rozwiązanie analityczne dla przykładu z piłką:*

**

S = 1.5 + 5.2\*1.2 -(9.81\*1.2\*1.2/2)

S = 0.6768

S\_końcowa = S\_0 + S = 2.1768

W czasie = 1.2 piłka jest już w trakcie spadania, w związku z czym wysokość jest trochę mniejsza, niż ta wyliczona za pomocą programu.

* *WYKRESY:*

**

*Rys1: Wykres drogi od czasu z oporem.*

**

*Rys.2: Wykres prędkości od czasu z oporem.*



*Rys3: Wykres drogi od czasu bez oporu.*



*Rys3: Wykres prędkości od czasu bez oporu.*

Oś Y została ograniczona od dołu zerem, ze względu na brak możliwości spadnięcia piłki poniżej poziomu ziemi. Dla rozwiązania, w którym nie zostały uwzględnione opory, droga zatrzymania jest dłuższa, a co za tym idzie, również czas.