**ИНСТИТУТ ТРАНСПОРТА И СВЯЗИ**



ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК И ТЕЛЕКОММУНИКАЦИЙ

Лабораторная работа №4

по предмету

«Численные методы»

«Решение дифференциальных уравнений первого порядка»

Выполнили: Беляев Марк

Михаил Красильников

Группа: 4601BV

Проверил: Александр Граковский

Рига, 2020

Содержание

[1. Задание. 3](#_Toc29836809)

[2. Аппроксимация методом МНК 3](#_Toc29836810)

[3. Индивидуальный метод – заданных точек 5](#_Toc29836811)

[4. Графическое сравнение методов 6](#_Toc29836812)

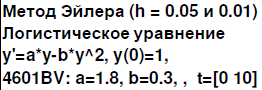
[5. Выводы 7](#_Toc29836813)

1. Задание.

Для выполнения лабораторной работы необходимо

1. реализовать метод Эйлера
2. реализовать индивидуальный метод – Метод Гира 2-го порядка ,
3. графически сравнить два реализованных метода, т.е. на один график нанести одну функцию, которая будет обработана двумя различными методами приближения.

Задачи для решения:



1. Метод Эйлера  
     
   public (List<float>, List<float>) Solve(float startXPoint, float endXPoint, float initialСondition, float partition)

{

int overalSteps = (int)(1 / partition);

var step = (endXPoint - startXPoint) \* partition;

step = (float)Math.Round(step \* 100f) / 100f; // Round

X = startXPoint;

Y = initialСondition;

XList.Add(X);

YList.Add(Y);

for (int i = 0; i < overalSteps; i++)

{

Y += step \* F(X, Y);

YList.Add(Y);

X += (float)Math.Round(step \* 100f) / 100f; // Round

XList.Add((float)Math.Round(X \* 100f) / 100f); // Round

}

return (XList, YList);

}

private float F(float x, float y)

{

return (1.8f \* y) + (0.3f \* (float)Math.Pow(x,2));

}

**Вызов для h = 0.01**

var result = euler.Solve(0, 10, 1, 0.01f);

x y=f(x)

0 1

0.1 1.18

0.2 1.3927

0.3 1.644586

0.4 1.94331148

0.5 2.2979075464

0.6 2.719030904752

0.7 3.21925646760736

0.8 3.8134226317766848

0.9 4.51903870549648806

1 5.35676567248585592

1.1 6.35098349353330998

1.2 7.53046052236930578

1.3 8.92914341639578082

1.4 10.5870892313470214

1.5 12.5515652929894852

1.6 14.8783470457275926

1.7 17.6332495139585592

1.8 20.8939344264710999

1.9 24.7520426232358978

2 29.3157102954183595

2.1 34.7125381485936642

2.2 41.0930950153405237

2.3 48.635052118101818

2.4 57.5480614993601452

2.5 68.0795125692449713

2.6 80.5213248317090662

2.7 95.2179633014166981

2.8 112.575896695671704

2.9 133.07475810089261

3 157.28051455905328

3.1 185.861007179682871

3.2 219.604288472025788

3.3 259.440260396990429

3.4 306.466207268448707

3.5 361.976924576769474

3.6 427.500271000587979

3.7 504.839119780693815

3.8 596.120861341218702

3.9 703.855816382638068

4 831.00616333151292

4.1 981.067272731185246

4.2 1,158.16368182279859

4.3 1,367.16234455090234

4.4 1,613.80626657006476

4.5 1,904.87219455267641

4.6 2,248.35668957215817

4.7 2,653.69569369514664

4.8 3,132.02361856027303

4.9 3,696.47906990112218

5 4,362.56560248332417

5.1 5,148.57741093032252

5.2 6,076.10164489778058

5.3 7,170.6111409793811

5.4 8,462.1638463556697

5.5 9,986.2281386996902

5.6 11,784.6567036656345

5.7 13,906.8357103254487

5.8 16,411.0408381840294

5.9 19,366.0373890571547

6 22,852.9684190874426

6.1 26,967.5827345231822

6.2 31,822.863926737355

6.3 37,552.1326335500789

6.4 44,312.7072075890931

6.5 52,290.22330495513

6.6 61,703.730999847053

6.7 72,811.709379819523

6.8 85,919.163768187037

6.9 101,386.000446460704

7 119,636.90882682363

7.1 141,173.022415651884

7.2 166,585.678750469223

7.3 196,572.656125553683

7.4 231,957.332928153346

7.5 273,711.295655220948

7.6 322,981.016373160719

7.7 381,119.332120329648

7.8 449,722.590601988985

7.9 530,674.482110347

8 626,197.76119020946

8.1 738,915.27820444717

8.2 871,921.99658124766

8.3 1,028,869.97316587223

8.4 1,214,068.63503572924

8.5 1,432,603.1061421605

8.6 1,690,473.83274774939

8.7 1,994,761.34144234428

8.8 2,353,820.65360196625

8.9 2,777,510.69445032017

9 3,277,464.9957513778

9.1 3,867,411.12498662581

9.2 4,563,547.61178421845

9.3 5,384,988.72110537777

9.4 6,354,289.2856043458

9.5 7,498,064.007813128

9.6 8,847,718.2367194911

9.7 10,440,310.2841289994

9.8 12,319,568.9579722193

9.9 14,537,094.2516072188

10 17,153,774.1571965182  
**Вызов для h = 0.05**  
var result = euler.Solve(0, 10, 1, 0.05f);

x y=f(x)

0 1

0.5 1.9

1 3.6475

1.5 7.08025

2 13.789975

2.5 26.8009525

3 51.85930975

3.5 99.882688525

4 191.6146081975

4.5 366.46775557525

5 699.326235592975

5.5 1,332.4698476266525

6 2,536.23021049063975

6.5 4,824.23739993221553

7 9,172.3885598712095

7.5 17,434.8882637552981

8 33,134.7252011350663

8.5 62,965.5778821566259

9 119,645.435476097589

9.5 227,338.47740458542

10 431,956.644568712297

1. Индивидуальный метод – Метод Гира 3-го порядка

public class Main {

private static final float LOW\_PRECISION = 0.05f;

private static final float HIGH\_PRECISION = 0.01f;

public static void main(String[] args) {

float end = 1f;

float y0 = 1.0f;

float step = Main.HIGH\_PRECISION;

int rank = 2;

int n = (int)(end / step);

float[] listOfY = new float[n];

listOfY[0] = y0;

for (int i = 1; i <= rank; i++) {

float y = listOfY[i - 1];

listOfY[i] = y + step \* f( step \* i, y);

}

for (int i = rank; i < n; i++) {

float y1 = listOfY[i - 1];

float y2 = listOfY[i - 2];

float predict = f(step \* i, y1 + step \* f(step \* i, y1));

listOfY[i] = ((4.f / 3.f) \* y1) - ((1.f/3.f) \* y2) + (step \* (2.f/3.f) \* predict);

}

for (int i = 0; i < listOfY.length; i++) {

float X = (i \* step);

float real = \_f(X, listOfY[i]);

System.out.println("X = " + X + " Y = " + listOfY[i] + " err: " + (Math.abs(listOfY[i] - real)));

}

}

public static float f(float x, float y) {

return (1.8f \* y) - (0.3f \* (float)Math.pow(y,2));

}

public static float \_f(float x, float y) {

float tmp = (float)Math.exp(1.8f \* x);

return 1.8f \* y \* tmp / (1.8f + 0.3f \* y \* (tmp - 1));

}

}

1. Графическое сравнение методов
2. Выводы

В данной лабораторной работе были использованы два метода решения дифференциальных уравнений явный и неявный

Метод Гира дал более точное решение. Его недостаток состоит в затрате времени на каждую итерацию. Это связано с попыткой совместить свойства явного метода и встроить в тело алгоритма цифровой сглаживающий фильтр 2-го порядка

Если же сравнивать полученные графики, видно, что эталонное и численное решение выданные явным методом совпадают. Это и есть доказательство его неустойчивости. При очень большом шаге кривые пойдут вразнос. Для борьбы с неустойчивостью используют схему "прогноз - коррекция". Аналитическое решение метода Гира 2-го порядка также сходится с графиком кривой, полученном предыдущим методом, тем не менее кривая для численного решения резко отличается и после тройки выгнута в противоположную сторону.

Увеличивая точность, мы получаем более реальных решения, но платим за это ресурсами процессора вычислительной машины и своим временем

Конкретно, после указания шага 0.001 машина зависла и заставила ожидать решение около 1 минуты. Это может серьезно повлиять на крупные технологические процессы. Не всегда выгодно рисковать временем вычислений, особенно если их много, поэтому точность должна соответствовать желаемому результату, однако не превосходить его