УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Дисциплина: вычислительная математика

Лабораторная работа №5

“Решение ОДУ методом Милна”

Выполнил: Тарасов Александр

Группа: P3212

Преподаватель: Перл О.В.

Вариант: метод Милна

Санкт-Петербург, 2020 г.

Описание метода

Многошаговые методы основаны на вычислении значения y_{i+1} помощью результатов k предыдущих шагов, то есть с помощью значений
y_{i-k+1}, y_{i-k+2}, …y_i. Такой метод называется k-шаговым.

Многошаговые методы для нахождения очередного значения используется следующая схема:

исходное уравнение записывается в виде $dY(x) = f(x, Y)dx$

Далее интегрируются обе части уравнения на отрезке [x_i, x_{i+1}]. Для левой части интегрирование очевидно, а для интегрирования правой части строится интерполяционный многочлен по значениям
$f(x_{i-k+1}, y_{i-k+1}), f(x_{i-k+2}, y_{i-k+2}), \ldots, f(x_i, y_i)$

То есть $y_{i+1} = y_i + \int\limits_{x_i}^{x_{i+1}} P_{k-1}(x)dx$ - формула для вычисления значения сеточной функции в узле x_{i+1}.

На основе этой формулы строятся различные многошаговые методы.

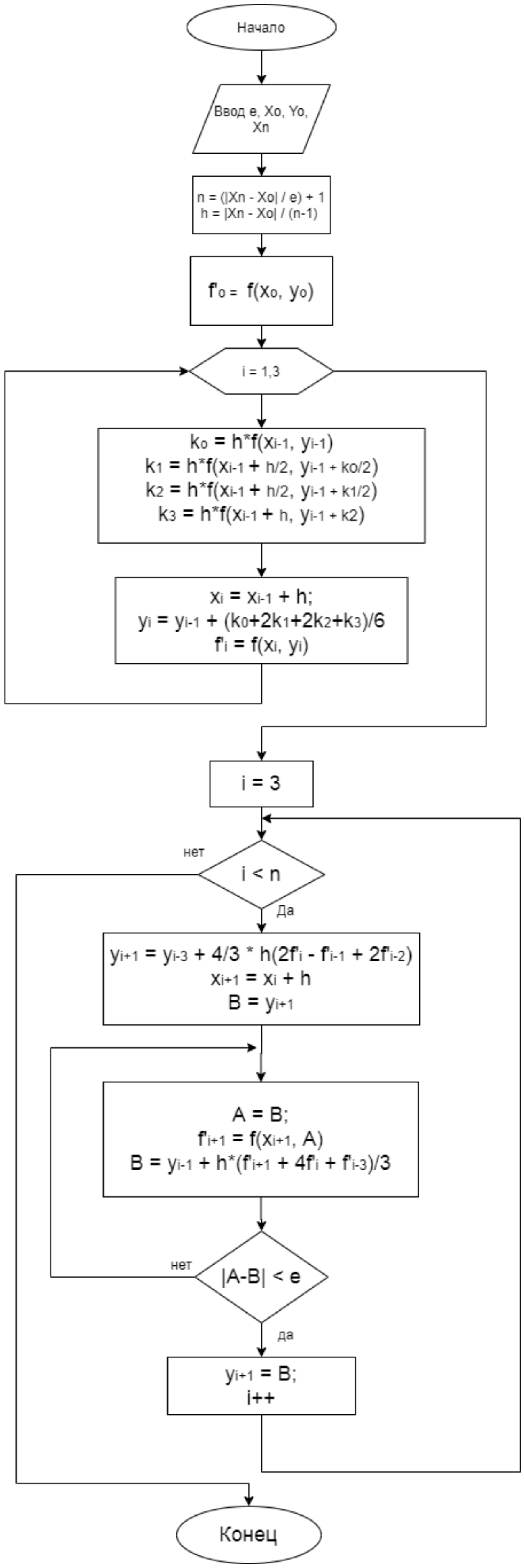
Одним из таких методов является метод Милна. Метод Милна – это метод семейства методов прогноза и коррекции. В таких методах применяются две формулы – формула прогноза и формула коррекции.

Так как данный метод использует ранее полученные значения, необходимо получить начальные k значений. Обычно это делается с помощью одношагового метода. В моем случае я использую метод Рунге-Кутты.
Далее по формуле прогноза находится значение сеточной функции, которое затем корректируется формулой коррекции до тех пор, пока не будет достигнута заданная точность.
В методе Милна используются следующие формулы:

$y_{i+1} = y_{i-3} + \frac{4}{3}h \cdot (2y'_i - y'_{i-1} + 2y'_{i-2}) + O(h^5)$ - для прогноза

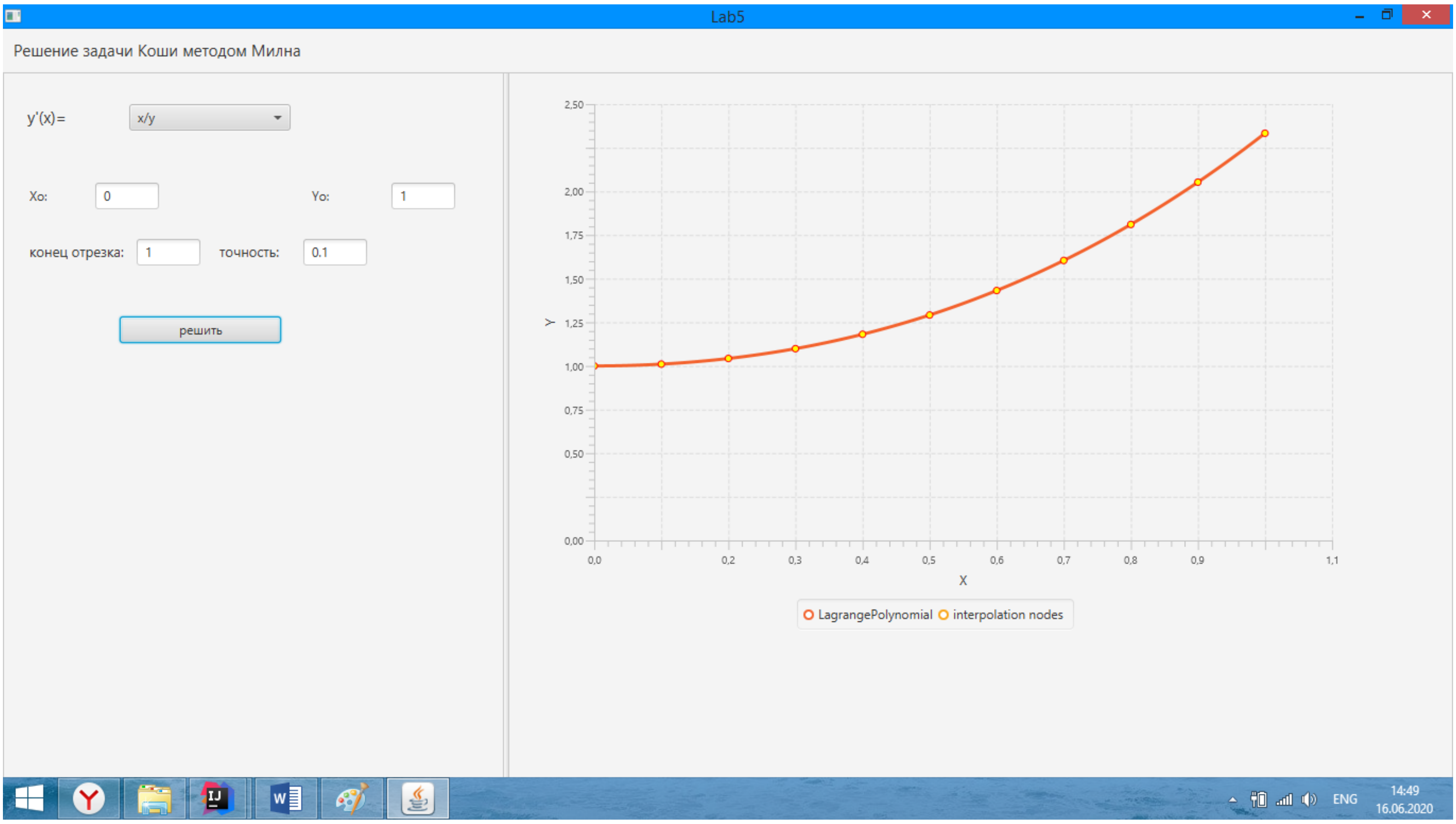
и формула Симпсона
$y_{i+1} = y_{i-1} + \frac{1}{3}h \cdot (y'_{i+1} + 4y'_i + y'_{i-1}) + O(h^5)$ - для коррекции значения.

Блок-схема численного метода

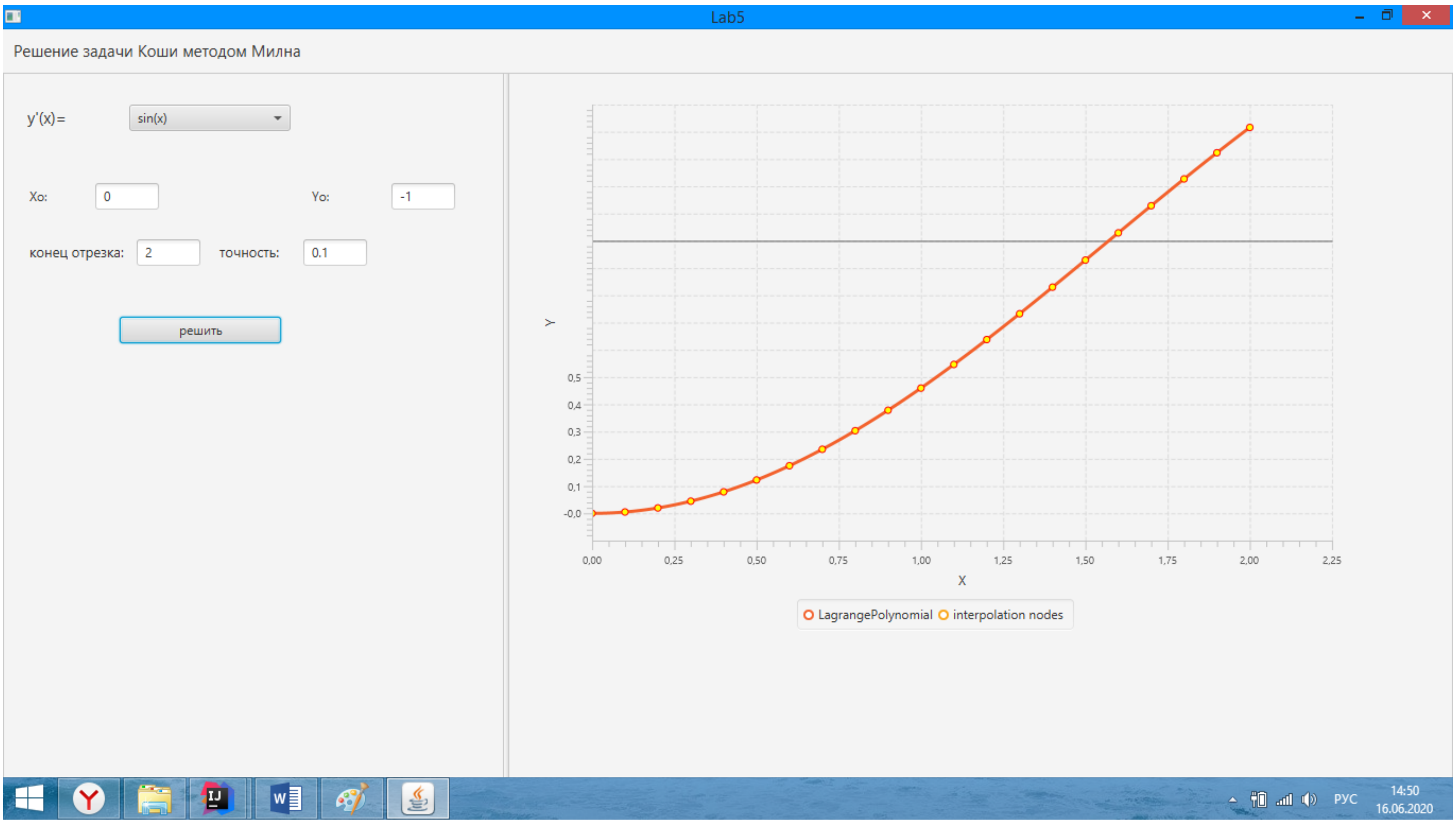


Примеры

Пример 1



Пример 2



Листинг численного метода

MilnaMethod.java

<…>

```java
    public static void initParam(Functions func, double xBgn, double yBegin, double xnd, double
accuracy){

        function = func;
        xBegin = xBgn;
        xEnd = xnd;
        E = accuracy;

        dotsAmnt = (int) Math.ceil(  (xEnd - xBegin)/E) + 1;
        h =  (xEnd - xBegin)/ (dotsAmnt - 1);

        xValues = new Double[dotsAmnt];
        yValues = new Double[dotsAmnt];
        derValues = new double[dotsAmnt];
        xValues[0] = xBegin;
        yValues[0] = yBegin;
        derValues[0] = function.solve(xValues[0], yValues[0]);

        solve();
    }


    public static void solve(){
        for (int i = 1; i <= 3; i++){
            k0 = h*function.solve(xValues[i-1], yValues[i-1]);
            k1 = h*function.solve(xValues[i-1] + h/2,yValues[i-1] + k0/2);
            k2 = h*function.solve(xValues[i-1] + h/2,yValues[i-1] + k1/2);
            k3 = h*function.solve(xValues[i-1] + h,yValues[i-1] + k2);
            xValues[i] =  xValues[i-1] + h;
            yValues[i] =  yValues[i-1] + (k0 + 2*k1 + 2*k2 + k3)/6;
            derValues[i] = function.solve(xValues[i], yValues[i]);
        }

       int i=3;

        while (i < dotsAmnt - 1){
            yValues[i+1]  = yValues[i-3] +  (4/3)*h*(2*derValues[i] - derValues[i-1] +
2*derValues[i-2]);
            xValues[i+1] = xValues[i] + h;

            B = yValues[i+1];
          do{
                A = B ;
                derValues[i+1] = function.solve(xValues[i+1], A);
                B = yValues[i-1] + h*(derValues[i+1] + 4*derValues[i] + derValues[i-1])/3;
            } while(Math.abs(A-B) > E);
            yValues[i+1] = B;
            i++;
        }
    }
```

<…>

AppController.java

```java
<…>
@FXML
void solveBtnPressed(ActionEvent event) {
    xBegin = Double.parseDouble(xBeginInput.getText());
    yBegin = Double.parseDouble(yBeginInput.getText());
    xEnd = Double.parseDouble(xEndInput.getText());
    accuraccy = Double.parseDouble(accuraccyInput.getText());
    MilnaMethod.initParam(currentFunction, xBegin, yBegin, xEnd, accuraccy);


    LagrangePolynomial.initParam(MilnaMethod.getxValues(), MilnaMethod.getyValues(),
MilnaMethod.getDotsAmnt());
    xVals = MilnaMethod.getxValues();
    drawGraph();
    drawPoints();


}

private void drawGraph(){
    double step = (xVals[1] - xVals[0])/40;
    XYChart.Series<Double, Double> series = new XYChart.Series<>();
    series.setName("LagrangePolynomial");
    for (double x = xBegin; x < xVals[MilnaMethod.getDotsAmnt()-1] ; x = x + step) {
        series.getData().add(new XYChart.Data<>(x,
LagrangePolynomial.interpolite(x)));
    }
    chart.getData().setAll(series);
}

private void drawPoints() {
    XYChart.Series<Double, Double> series = new XYChart.Series<>();
    for (int i = 0; i < MilnaMethod.getDotsAmnt(); i++){
        XYChart.Data point = new XYChart.Data(LagrangePolynomial.getxValues().get(i),
LagrangePolynomial.getyValues().get(i));
        point.setNode(new Circle(3.0, Color.YELLOW));
        series.getData().add(point);
    }
    series.setName("interpolation nodes");
    chart.getData().add(series);
}
 <…>
```

Вывод

Численные методы решения ОДУ выгодно применять, когда нет возможности получить решение аналитически.
Для этой цели существуют различные методы – одношаговые, многошаговые.
Преимуществом метода Милна можно считать высокую точность (4 порядок точности).
Однако метод Милна использует одношаговый метод для формирования начальных данных, то есть зависим от него. Это можно отнести к недостатку.

Метод Эйлера прост в реализации, однако с увеличением числа узлов копится погрешность результата.

Метод Рунге-Кутты точнее метода Эйлера, однако требует бОльшее количество вычислений.