yМИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Поволжский государственный университет телекоммуникаций и информатики»

Факультет информатики и вычислительной техники

Кафедра ПрИ

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №8**

**Дисциплина: Численные методы**

Выполнил: студент  
ПрИ-21 Плеханов М.А.  
Проверил(а):  
Осанов В.А.

Самара 2024

**Цель работы:** изучить методы решения задачи Коши, такие как метод Эйлера, метод Эйлера-Коши, метод Рунге-Кутта 4-го порядка, реализовать программными средствами.

Вариант 13

Таблица 1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***F1(x,y1,y2)*** | ***F2(x,y1,y2)*** | ***y1(a)*** | ***y2(a)*** | ***a*** | ***b*** |
|  |  | 1 | -1 | 2 | 4 |

**Метод Эйлера:** это численный метод, который используется для приближенного решения обыкновенных дифференциальных уравнений. Основная идея метода заключается в том, чтобы на каждом шаге вычислять новое значение функции, используя текущее значение и производную (или правую часть уравнения).

|  |
| --- |
| **Метод Эйлера:**  public class Euler {  public static void solve(double a, double b, int n, Function2[] TFMas, double[] y) {  double h = (b - a) / n;  double x = a;  double[][] TFunZnach = new double[n + 1][2];  TFunZnach[0][0] = y[0];  TFunZnach[0][1] = y[1];  for (int i = 0; i < n; i++) {  double xNext = x + h;  double[] yNext = new double[2];  yNext[0] = y[0] + h \* TFMas[0].apply(x, y[0], y[1]);  yNext[1] = y[1] + h \* TFMas[1].apply(x, y[0], y[1]);  TFunZnach[i + 1][0] = yNext[0];  TFunZnach[i + 1][1] = yNext[1];  x = xNext;  y = yNext;  }  for (int i = 0; i <= n; i++) {  double xi = a + i \* h;  System.out.printf("x = %.2f, y1 = %.6f, y2 = %.6f%n", xi, TFunZnach[i][0], TFunZnach[i][1]);  }  }  } |

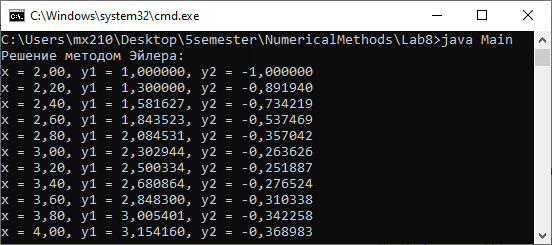


Рис.1.

**Метод Эйлера-Коши:** это улучшенная версия метода Эйлера, которая использует среднее значение производной на шаге интегрирования для вычисления следующего значения функции. Это позволяет улучшить точность приближения.

|  |
| --- |
| **Метод Эйлера-Коши:**  public class EulerCauchy {  public static void solve(double a, double b, int n, Function2[] TFMas, double[] y) {  double h = (b - a) / n;  double x = a;  double[][] TFunZnach = new double[n + 1][2];  TFunZnach[0][0] = y[0];  TFunZnach[0][1] = y[1];  for (int i = 0; i < n; i++) {  double xNext = x + h;  double[] yNext = new double[2];  yNext[0] = y[0] + h \* TFMas[0].apply(x + h / 2, y[0] + h / 2 \* TFMas[0].apply(x, y[0], y[1]), y[1]);  yNext[1] = y[1] + h \* TFMas[1].apply(x + h / 2, y[0] + h / 2 \* TFMas[0].apply(x, y[0], y[1]), y[1]);  TFunZnach[i + 1][0] = yNext[0];  TFunZnach[i + 1][1] = yNext[1];  x = xNext;  y = yNext;  }  for (int i = 0; i <= n; i++) {  double xi = a + i \* h;  System.out.printf("x = %.2f, y1 = %.6f, y2 = %.6f%n", xi, TFunZnach[i][0], TFunZnach[i][1]);  }  }  } |

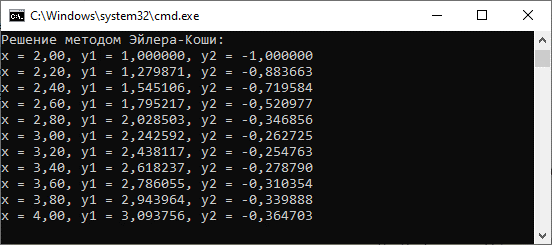


Рис.2.

**Метод Рунге-Кутта:** это более сложный численный метод, который использует несколько промежуточных вычислений для получения более точного результата. Он обладает более высокой точностью, чем методы Эйлера и Эйлера-Коши. Этот метод обладает высокой точностью (четвертый порядок аппроксимации), что делает его предпочтительным для большинства задач, требующих точных решений.

|  |
| --- |
| **Метод Рунге-Кутта:**  public class RungeKutta {  public static void solve(double a, double b, int n, Function2[] TFMas, double[] y) {  double h = (b - a) / n;  double x = a;  double[][] TFunZnach = new double[n + 1][2];  TFunZnach[0][0] = y[0];  TFunZnach[0][1] = y[1];  // Вычисление методом Рунге-Кутта 4-го порядка  for (int i = 0; i < n; i++) {  double xNext = x + h;  // Вычисление коэффициентов k1, k2, k3, k4  double k1\_0 = h \* TFMas[0].apply(x, y[0], y[1]);  double k1\_1 = h \* TFMas[1].apply(x, y[0], y[1]);  double k2\_0 = h \* TFMas[0].apply(x + h / 2, y[0] + k1\_0 / 2, y[1] + k1\_1 / 2);  double k2\_1 = h \* TFMas[1].apply(x + h / 2, y[0] + k1\_0 / 2, y[1] + k1\_1 / 2);  double k3\_0 = h \* TFMas[0].apply(x + h / 2, y[0] + k2\_0 / 2, y[1] + k2\_1 / 2);  double k3\_1 = h \* TFMas[1].apply(x + h / 2, y[0] + k2\_0 / 2, y[1] + k2\_1 / 2);  double k4\_0 = h \* TFMas[0].apply(x + h, y[0] + k3\_0, y[1] + k3\_1);  double k4\_1 = h \* TFMas[1].apply(x + h, y[0] + k3\_0, y[1] + k3\_1);  double[] yNext = new double[2];  yNext[0] = y[0] + (k1\_0 + 2 \* k2\_0 + 2 \* k3\_0 + k4\_0) / 6;  yNext[1] = y[1] + (k1\_1 + 2 \* k2\_1 + 2 \* k3\_1 + k4\_1) / 6;  TFunZnach[i + 1][0] = yNext[0];  TFunZnach[i + 1][1] = yNext[1];  x = xNext;  y = yNext;  }  for (int i = 0; i <= n; i++) {  double xi = a + i \* h;  System.out.printf("x = %.2f, y1 = %.6f, y2 = %.6f%n", xi, TFunZnach[i][0], TFunZnach[i][1]);  }  }  } |

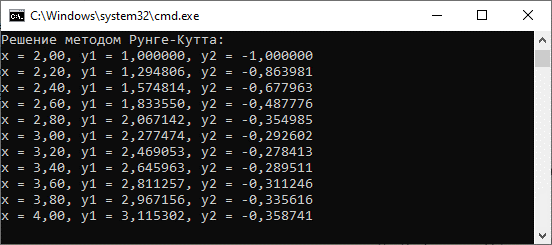


Рис. 3.

Пример решения в Mathcad

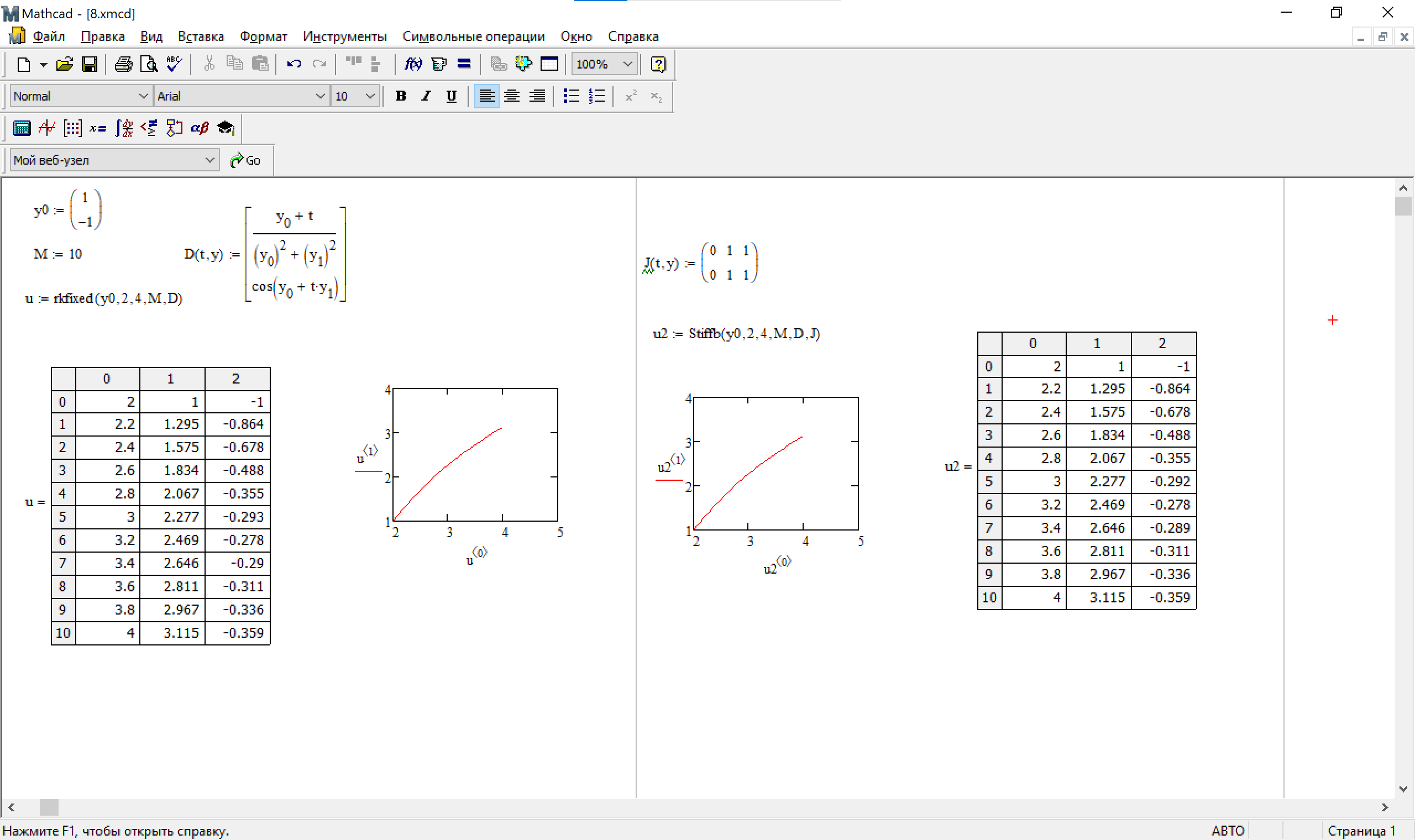


Рис.4.