Санкт-Петербургский политехнический университет Петра Великого

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе 3**

**Дисциплина**: Вычислительная математика

Вариант 1

Выполнил студент гр. 3530901/90003 В.С. Андрианов

(подпись)

Преподаватель В.Н. Цыган

(подпись)

“ ” 2021 г.

Санкт-Петербург

2021

**Содержание**

1. **Техническое задание**
2. **Цель работы**
3. **Ход выполнения работы**
   1. **Используемая среда разработки и язык программирования**
   2. **Объяснение подхода к решению поставленной задачи**
   3. **Описание входных и выходных данных для работы исследуемых процедур**
   4. **Листинг программы**
   5. **Результат выполнения программы**
4. **Вывод**

**1. Техническое задание**

Привести дифференциальное уравнение: к системе двух дифференциальных уравнений первого порядка.

Начальные условия:

Точное решение:

Решить на интервале 1≤t ≤2

1. используя программу RKF45 с шагом печати = 0.1 и выбранной

погрешностью EPS в диапазоне 0.001 – 0.00001;

1. Используя метод Рунге-Кутты 3-й степени точности с шагом

интегрирования = 0.1

Исследовать влияние величины шага интегрирования на величины локальной и глобальной погрешностей решения заданного уравнения, для этого взять шаг вычисления

**2. Цель работы**

Использовать в рабочих целях реализацию базовых подпрограмм, таких как RKF45, а также рассмотреть метод Рунге-Кутты третьего порядка.

**3. Ход выполнения работы**

**3.1 Используемая среда разработки и язык программирования**

Для решения поставленной задачи в качестве языка программирования был выбран язык С++. Разработка велась в Clion (среда разработки).

**3.2 Объяснение подхода к решению поставленной задачи**

Метод Рунге-Кутты 3ей степени точности:

Приведение дифференциального уравнения: к системе двух дифференциальных уравнений первого порядка**:**

Пусть , тогда . После замены получим:

После получения системы двух дифференциальных уравнений первого порядка необходимо составить процедуру, которая рассчитывает значения . Составленной процедурой является процедура fun(t, y, dy). Процедура fun получает на вход значение интервала t, значение y и дифференциала dy.

С помощью полученной процедуры fun(), можно использовать подпрограмму RKF45(fun, n, x, T, TOUT, RE, AE, IFLAG, WORK, IWORK). Для получения всех значений на промежутке [1, 2] с шагом интегрирования h = 0.1 и погрешностью 0.00001 была составлена процедура rkf45(y, dy, y0, dy0, h), которая на вход принимает два массива y и dy, в которые будут записаны результаты, значения y0, dy0 и шаг интегрирования h.

Для вычисления методом Рунге-Кутты 3ей степени точности была составлена процедура RungeKutta3(t, Xn, Xnp0, Xnp1, h, i). На вход процедура RungeKutta3() получает: точку вычисления t, вектор значений Xn в начале промежутка, два массива Xnp0 и Xnp1, в которые будут записаны результаты работы, шаг интегрирования h и номер точки вычисления i. Для вычисления всех точек промежутка была составлена процедура RungeKutta(Xn, Xnp0, Xnp1, h), которая на вход получает вектор значений Xn в начале промежутка, два массива Xnp0 и Xnp1, в которые будут записаны результаты работы, шаг интегрирования h.

Для вычисления погрешности результатов были составлены несколько процедур. Процедура funSolver(y, dy, h) получает на вход массивы y и dy (в них будет записан результат) и шаг вычисления h. Процедура funSolver() вычисляет точное значение системы. Процедура errorRKF(Y0rkf, Y1rkf, F0, F1) получает на вход результаты работы процедур rkf45() и funSolver(), и вычисляет погрешность результатов процедуры rkf45(). Процедура errorRungeKutta(Y0rkf, Y1rkf, F0, F1) получает на вход результаты работы процедур RungeKutta() и funSolver(), и вычисляет погрешность результатов процедуры RungeKutta().

**3.3 Описание входных и выходных данных для работы исследуемых процедур**

**RKF45 (F, N, X, T, TOUT, RE, AE, IFLAG, WORK, IWORK)**

**F** – имя процедуры, написанной пользователем для вычисления правых частей системы

**N** – количество интегрируемых уравнений

**X** – вектор решения размерностью **N** в точке **Т** на входе в программу и в точке **TOUT** при выходе из нее

**Т** – начальное значение независимой переменной на входе в программу (при нормальном выходе TOUT)

**TOUT** – точка выхода по независимой переменной

**RE** – границы относительной погрешности

**АE** – границы абсолютной погрешности

**WORK** – рабочий вещественный массив размерности 6N + 3

**IWORK** – рабочий целый массив размерности не менее 5

**IFLAG** – указатель режима интегрирования.

При первом обращении на входе **IFLAG = 1**, а при последующих обращениях на входе **IFLAG = 2**. Нормальное выходное значение **IFLAG = 2**. Другие выходные значения указывают на возникшие отклонения от нормального процесса интегрирования:

**IFLAG = 3** – заданное значение RE оказалось слишком малым и требуется его увеличить;

**IFLAG = 4** – потребовалось более 3000 вычислений f(t, x)

**IFLAG = 5** – решение обратилось в нуль, а АЕ равно нулю. Требуется задать ненулевое значение АЕ

**IFLAG = 6** – требуемая точность не достигнута даже при наименьшей допустимой величине шага и требуется увеличить АЕ и RE

**IFLAG = 7** – слишком большое число требуемых выходных точек препятствует выбору естественной величины шага. Нужно или увеличить TOUT–T или задать значение IFLAG=2 и продолжить работу программы

**IFLAG = 8** – неправильное задание параметров процедуры (например**, N < 0, AE < 0, RE < 0**).

**3.4 Листинг программы**

// количество вычислений  
int n = 11;  
  
// система первого порядка  
void fun(double t, double \*y, double \*dy) {  
 dy[0] = y[1];  
 dy[1] = ((t + 1) \* y[1] + 2 \* (t - 1) \* y[0]) / t;  
}  
  
// вычисление точного значения системы  
void funSolver(double \*y, double \*dy, double h) {  
 double T = 1.0;  
 printf("\nfun solutions");  
 for (int i = 0; i < n; ++i) {  
 y[i] = exp(2 \* T);  
 dy[i] = 2 \* exp(2 \* T);  
 printf("\nY0[0] = %.10lf Y0[1] = %.10lf", y[i], dy[i]);  
 T += h;  
 }  
}  
  
// применение процедуры rkf45 на всех точка в промежутке [1, 2]  
void rkf45(double \*y, double \*dy, double y0, double dy0, double h) {  
 double Y0[] = { y0, dy0 };  
 double T = 1.0, TOUT = 1.0;  
 int iflag = 1;  
 int iwork[30];  
 double work[15];  
 double RE = 1e-5, AE = 1e-5;  
 printf("RKF45");  
 for (int i = 0; i < n; ++i) {  
 RKF45(fun, 2, Y0, &T, &TOUT, &RE, &AE, &iflag, work, iwork);  
 printf("\nY[0] = %.10lf Y0[1] = %.10lf iflag = %d", Y0[0], Y0[1], iflag);  
 y[i] = Y0[0];  
 dy[i] = Y0[1];  
 TOUT += h;  
 }  
}  
  
// применение метода Рунге-Кутты 3ей степени точности  
void RungeKutta3(double t, double \*Xn, double \*Xnp0, double \*Xnp1, double h, int i) {  
  
 double k[2], k2[2], k1[2], k3[2];  
  
 fun(t, Xn, k1);  
  
 k1[0] = h \* k1[0];  
 k1[1] = h \* k1[1];  
  
 k[0] = Xn[0] + k1[0] / 2;  
 k[1] = Xn[1] + k1[1] / 2;  
  
 fun(t + h / 2, k, k2);  
  
 k2[0] = h \* k2[0];  
 k2[1] = h \* k2[1];  
  
 k[0] = Xn[0] - k1[0] + 2 \* k2[0];  
 k[1] = Xn[1] - k1[1] + 2 \* k2[1];  
  
 fun(t + h, k, k3);  
  
 k3[0] = h \* k3[0];  
 k3[1] = h \* k3[1];  
  
 Xnp0[i] = Xn[0] + (k1[0] + 4 \* k2[0] + k3[0]) / 6;  
 Xnp1[i] = Xn[1] + (k1[1] + 4 \* k2[1] + k3[1]) / 6;  
}  
  
// применение метода Рунге-Кутты 3ей степени точности на всех точка в промежутке [1, 2]  
void RungeKutta(double \*Xn, double \*Xnp0, double \*Xnp1, double h) {  
 double T = 1.0;  
 printf("\nRungeKutta3\n");  
 for (int i = 1; i < 12; ++i) {  
 RungeKutta3(T, Xn, Xnp0, Xnp1, h, i);  
 printf("Y[0] = %.10lf Y0[1] = %.10lf\n", Xn[0], Xn[1]);  
 Xn[0] = Xnp0[i];  
 Xn[1] = Xnp1[i];  
 T += h;  
 }  
}  
  
// вычисление погрешности работы процедуры rkf45  
void errorRKF(double \*Y0rkf, double \*Y1rkf, double \*F0, double \*F1) {  
 printf("\nRKF45 ERROR:\n");  
 for (int i = 0; i < n; ++i) {  
 printf("Y0rkf = %.10lf Y1rkf = %.10lf\n", abs(Y0rkf[i] - F0[i]), abs(Y1rkf[i] - F1[i]));  
 }  
}  
  
// вычисление погрешности работы процедуры RungeKutta  
void errorRungeKutta(double \*Y0rk3, double \*Y1rk3, double \*F0, double \*F1) {  
 printf("\nRunge Kutta 3 ERROR:\n");  
 for (int i = 0; i < n; ++i) {  
 printf("Y0rk3 = %.10lf Y1rk3 = %.10lf\n", abs(Y0rk3[i] - F0[i]), abs(Y1rk3[i] - F1[i]));  
 }  
}  
  
int main() {  
  
 double h = 0.1;  
  
 double Y0rkf[11], Y1rkf[11];  
 double Y0[2] = { exp(2), 2 \* exp(2) };  
 rkf45(Y0rkf, Y1rkf, Y0[0], Y0[1], h);  
  
 double Xn1[]= { exp(2), 2 \* exp(2) }, Xnp0[12] = { exp(2) }, Xnp1[12] { 2 \* exp(2) };  
 RungeKutta(Xn1, Xnp0, Xnp1, h);  
  
 double F0[11], F1[11];  
 funSolver(F0, F1, h);  
  
 errorRKF(Y0rkf, Y1rkf, F0, F1);  
 errorRungeKutta(Xnp0, Xnp1, F0, F1);  
  
 return 0;  
}

**3.5 Результат выполнения программы**

RKF45 – результат работы процедуры rkf45

RungeKutta3 – результат работы процедуры RungeKutta

Fun solutions – точное значение системы

RKF45 Error – погрешность результатов процедуры rkf45

RungeKutta3 Error – погрешность результатов процедуры RungeKutta

Шаг интегрирования h = 0.1

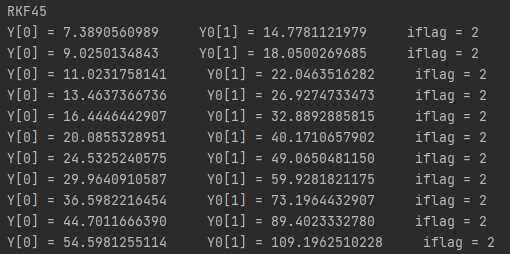


Рис. 1 Результат работы процедуры rkf45

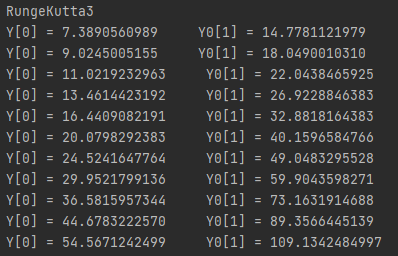


Рис. 2 Результат работы процедуры RungeKutta

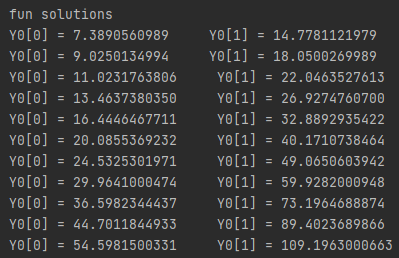


Рис. 3 Точное значение системы

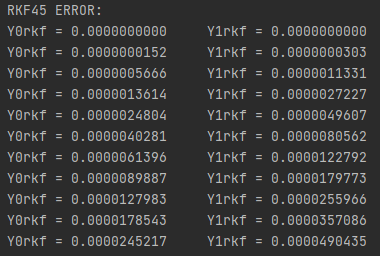


Рис. 4 Погрешность результатов процедуры rkf45

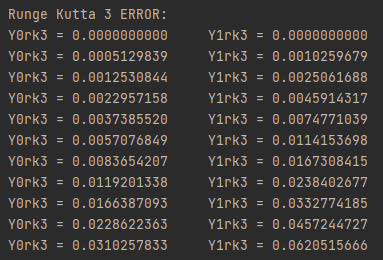


Рис. 5 Погрешность результатов процедуры RungeKutta

Шаг интегрирования h = 0.05

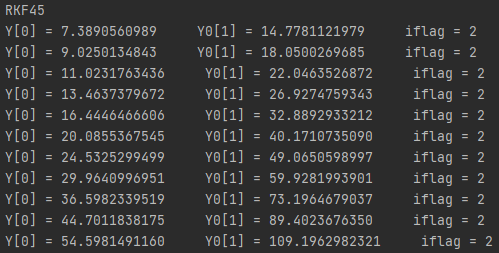


Рис. 6 Результат работы процедуры rkf45

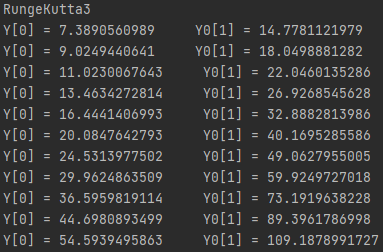


Рис. 7 Результат работы процедуры RungeKutta

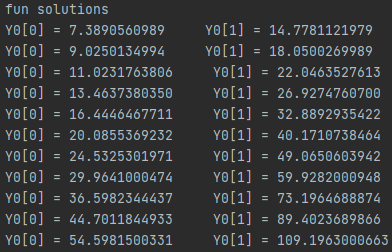


Рис. 8 Точное значение системы

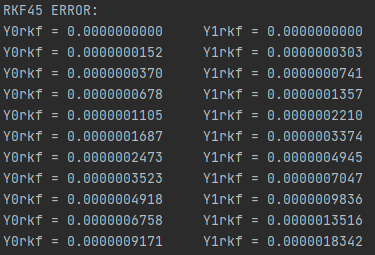


Рис. 9 Погрешность результатов процедуры rkf45

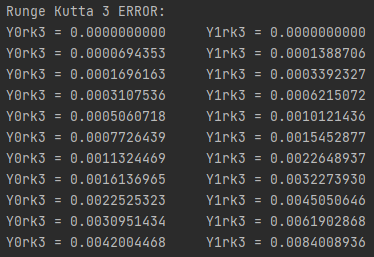


Рис. 10 Погрешность результатов процедуры RungeKutta

Шаг интегрирования h = 0.0125

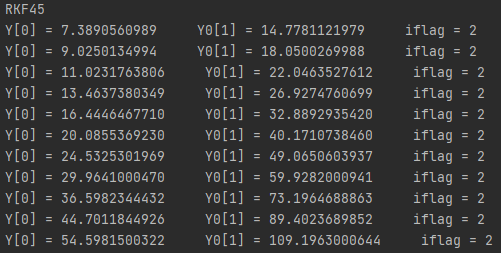


Рис. 11 Результат работы процедуры rkf45

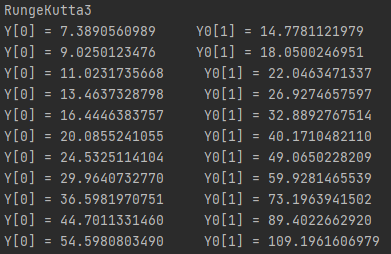


Рис. 12 Результат работы процедуры RungeKutta

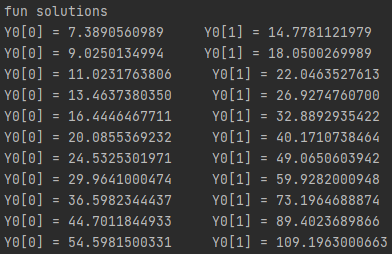


Рис. 13 Точное значение системы

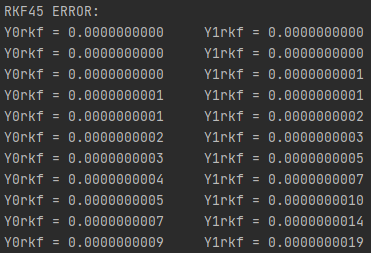


Рис. 14 Погрешность результатов процедуры rkf45

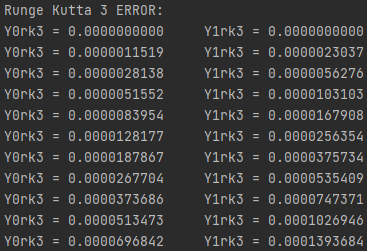


Рис. 15 Погрешность результатов процедуры RungeKutta

Исходя из полученных результатов работы процедур и значений, полученных из точного решения, можно сказать о корректности данных, полученных после выполнения составленной программы.

Из результатов погрешностей можно отметить, что погрешность растет на каждом шаге. Также стоит отметить, что на первых шагах результат работы процедуры RungeKutta() сопоставим с результатом работы процедуры rkf45(), но с каждым следующим шагом погрешность сильно возрастает.

Также из исследования поведения погрешностей при уменьшении шага интегрирования можно сделать вывод, что при уменьшении шага интегрирования растет количество вычислений, но сильно сокращается погрешность.

**4. Вывод**

В ходе выполнения работы было проведено исследование работы процедуры RKF45(), а также исследован метод Рунге-Кутты 3ей степени точности.

С помощью процедур RKF45() и RungeKutta() были вычислены значение системы дифференциальных уравнений первого порядка на интервале [1, 2]. Также было вычислено реальное значение функции на интервале.

С помощью полученных данных были найдены погрешности процедур RKF45() и RungeKutta(). Из результатов вычисления погрешностей видно, что погрешность растет на каждом шаге.

Также было проведено исследование влияния шага интегрирования на погрешность. Из результатов исследования можно сделать вывод, что при уменьшении шага интегрирования уменьшается и погрешность.