Московский Государственный Университет
имени М. В. Ломоносова

Компьютерный практикум по учебному курсу
«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

ЗАДАНИЕ № 2
Подвариант №1

«РЕШЕНИЕ ЗАДАЧИ КОШИ ДЛЯ ДИФФЕРЕНЦИАЛЬНОГО УРАВНЕНИЯ ПЕРВОГО ПОРЯДКА ИЛИ СИСТЕМЫ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА»

ОТЧЕТ
о выполненном задании

студента 206 учебной группы факультета ВМК МГУ
Оганисяна Эдгара Гагиковича

Москва, 2019 г.

Цель работы

Изучить методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка.

Постановка задачи

1) Рассматривается обыкновенное дифференциальное уравнение первого порядка, разрешенное относительно производной и имеющее вид:

$$\frac{dy}{dx} = f(x, y), \quad x_0 < x$$

с дополнительным начальным условием, заданным в точке $x = x_0$: $y(x_0) = y_0$
Условиями задачи гарантируется существование и единственность решения.

2) Рассматривается система обыкновенных дифференциальных уравнений первого порядка, разрешенных относительно производных неизвестных функций:

$$\begin{cases} \frac{dy_1}{dx} = f_1(x, y_1, y_2), \\ \frac{dy_2}{dx} = f_2(x, y_1, y_2), \quad x > x_0 \end{cases}$$

Дополнительные начальные условия задаются в точке $x = x_0$:

$$y_1(x_0) = y_1^{(0)}, \qquad y_2(x_0) = y_2^{(0)}$$

Условиями задачи гарантируется существование и единственность решения.

Цели и задачи практической работы

1) Решить задачи Коши методами Рунге-Кутта второго и четвертого порядка точности, аппроксимировав дифференциальную задачу соответствующей разностной схемой (на равномерной сетке); полученное конечно-разностное уравнение, представляющее некоторую рекуррентную формулу, просчитать численно.

2) Найти численной решение задачи и построить его график.

3) Найденное численное решение сравнить с точным решением (например с источника wolframalpha.com)

Описание метода решения

- **Метод Рунге-Кутта 2 порядка точности**

  Данный метод с использованием схемы вычислений типа «предиктор-корректор» является усовершенствованием метода Эйлера. Для получения результата используется следующая рекуррентная формула:

$$y_{i+1} = y_i + \frac{h}{2}\left\{ f\left(x_i, y_i\right) + f\left(x_i + h, y_i + hf\left(x_i, y_i\right)\right)\right\}$$

  Сначала делается шаг *h* и по схеме Эйлера вычисляется значение:

$$\tilde{y}_{i+1} = y_i + f\left(x_i, y_i\right)h$$

  Затем находится значение функции *f* в точке $\left(x_{i+1}, \tilde{y}_{i+1}\right)$, составляется полусумма: $\frac{f\left(x_i, y_i\right) + f\left(x_{i+1}, \tilde{y}_{i+1}\right)}{2}$ и окончательно:

$$y_{i+1} = y_i + \frac{f\left(x_i, y_i\right) + f\left(x_{i+1}, \tilde{y}_{i+1}\right)}{2}h$$

- **Метод Рунге-Кутта 4 порядка точности**

  Данный метод усовершенствование уже метода Рунге-Кутта. А именно, если в схеме второго приходилось на каждом шаге функцию *f(x, y)* приходилось вычислять 2 раза, то здесь — 4 раза. Однако усложнение схемы расчета окупается высокой точностью. Сама схема расчета:

$$y_{i+1} = y_i + \frac{1}{6}h\left(k_1 + 2k_2 + 2k_3 + k_4\right), \text{ где}$$

$$k_1 = f\left(x_i, y_i\right),$$
$$k_2 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_1\right),$$
$$k_3 = f\left(x_i + \frac{h}{2}, y_i + \frac{h}{2}k_2\right),$$
$$k_4 = f\left(x_i + h, y_i + hk_3\right).$$

Описание и листинг программы

Основной частью программы являются 4 функции реализующие методы Рунге-Кутты с различной точность как для одного уравнения, так и для системы. Пояснения будут далее.

Т.к программа достаточна велика, здесь приведем пояснения ко всем функциям. Текст программы будет доступен в приложении.

Сначала блок из тестовых ф-ций (из условия задания):

```
double f1(double x, double y);
double f1_exac(double x, double y);
double test2(double x, double y);
double test2_exac(double x, double y);
double test3(double x, double y);
double test3_exac(double x, double y);
double f1_sys(double x, double y, double z);
double f2_sys(double x, double y, double z);
```

Далее 4 функции реализующие методы Рунге-Кутта. Они в качестве параметров принимают одну (или две) функции *f, (g)*, указатели на массив переменных *x, y, z* для ф-ций, их размер *n*, шаг алгоритма *h*, а также файл *out* для записи результатаю.

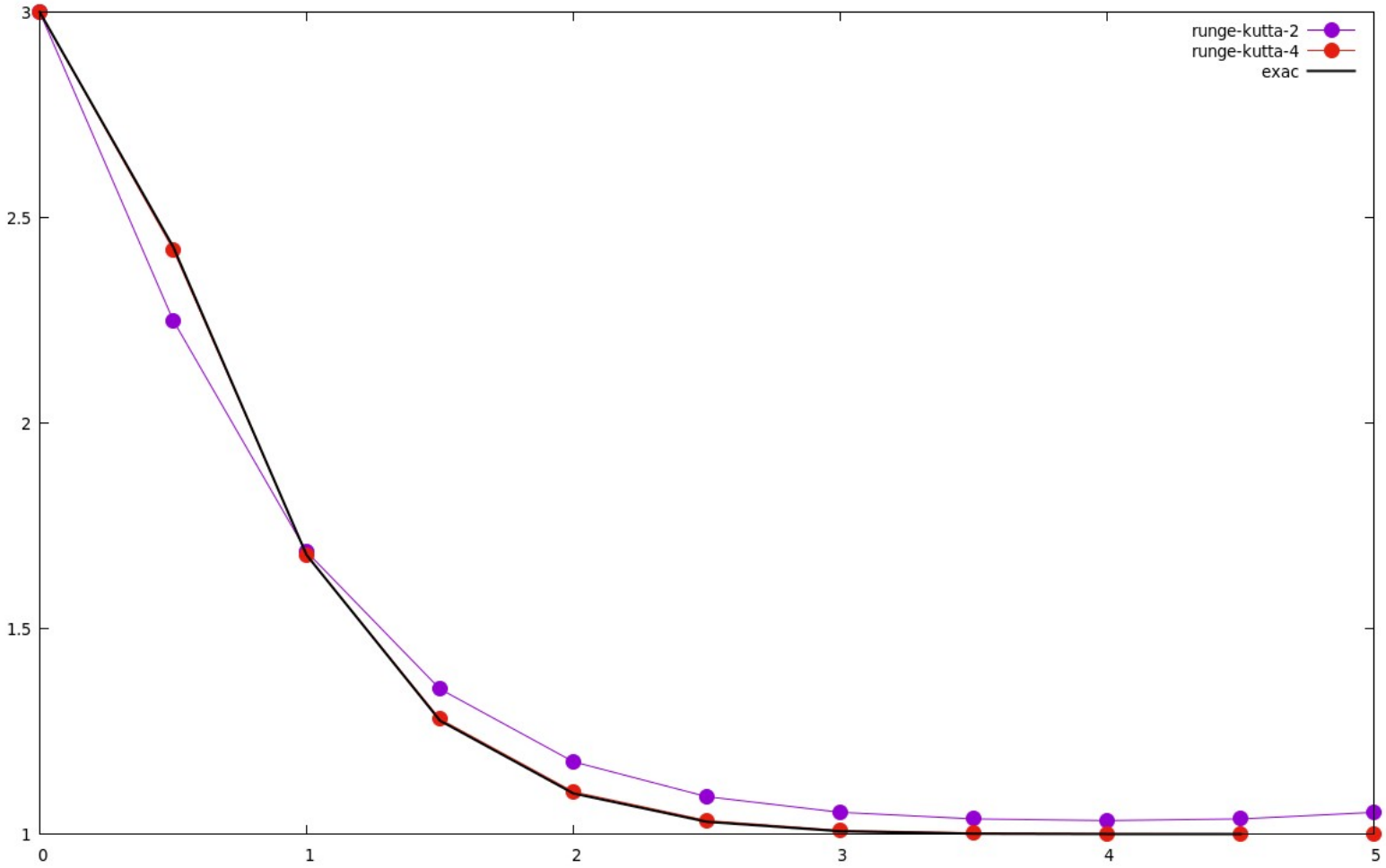
```
void runge_kutta_2(double (*f)(double, double), double
*x, double *y, double h, int n, FILE *out);

void runge_kutta_4(double (*f)(double, double), double
*x, double *y, double h, int n, FILE *out);

void runge_kutta_sys_2(double (*f)(double, double,
double), double (*g)(double, double, double), double *x,
double *y, double *z, double h, int n, FILE *out);

void runge_kutta_sys_4(double (*f)(double, double,
double), double (*g)(double, double, double), double *x,
double *y, double *z, double h, int n, FILE *out);
```

Тесты

Результаты тестов будут представлены в виде графиков с приближенными и точными решениями

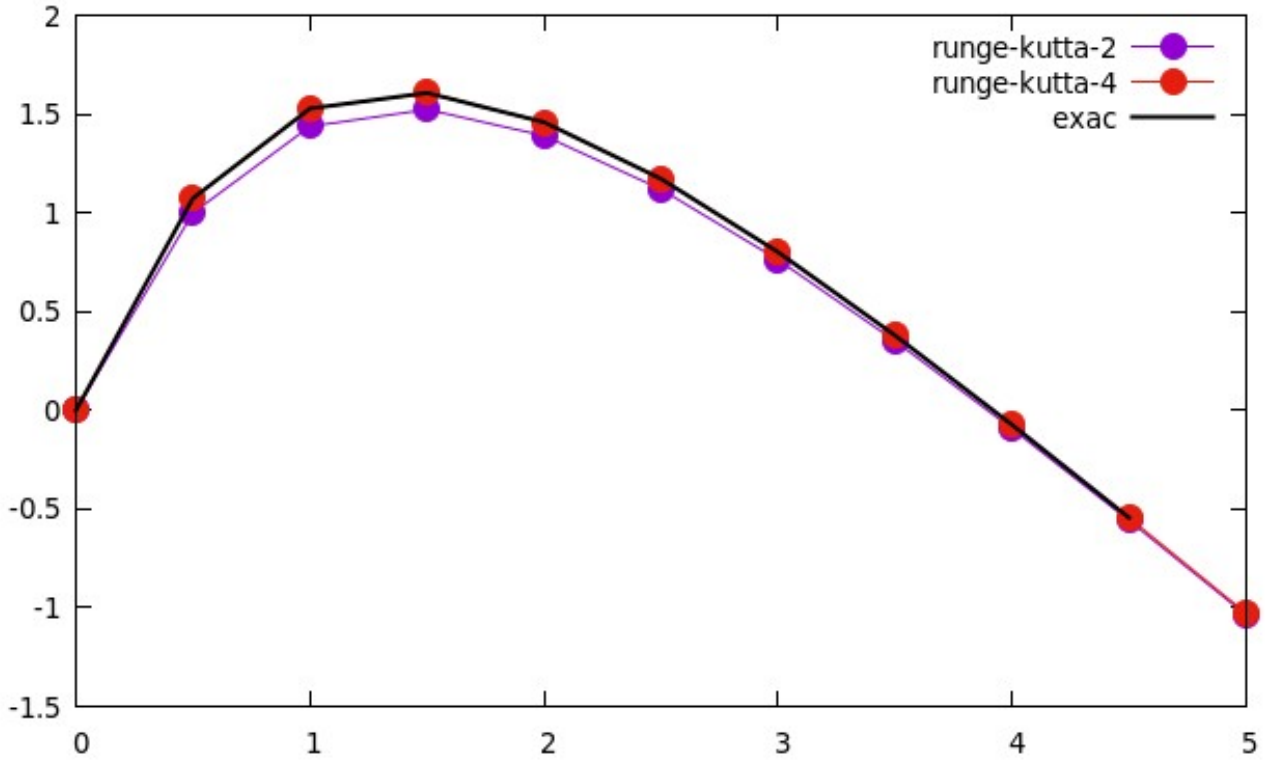
- **Тест №1**

$$f(x,y)=(y-y^2)\cdot x,\ \ y(0)=1 \quad \text{Точное решение:} \quad y(x)=\frac{e^{\frac{x^2}{2}}}{e^{\frac{x^2}{2}}-\frac{2}{3}}$$



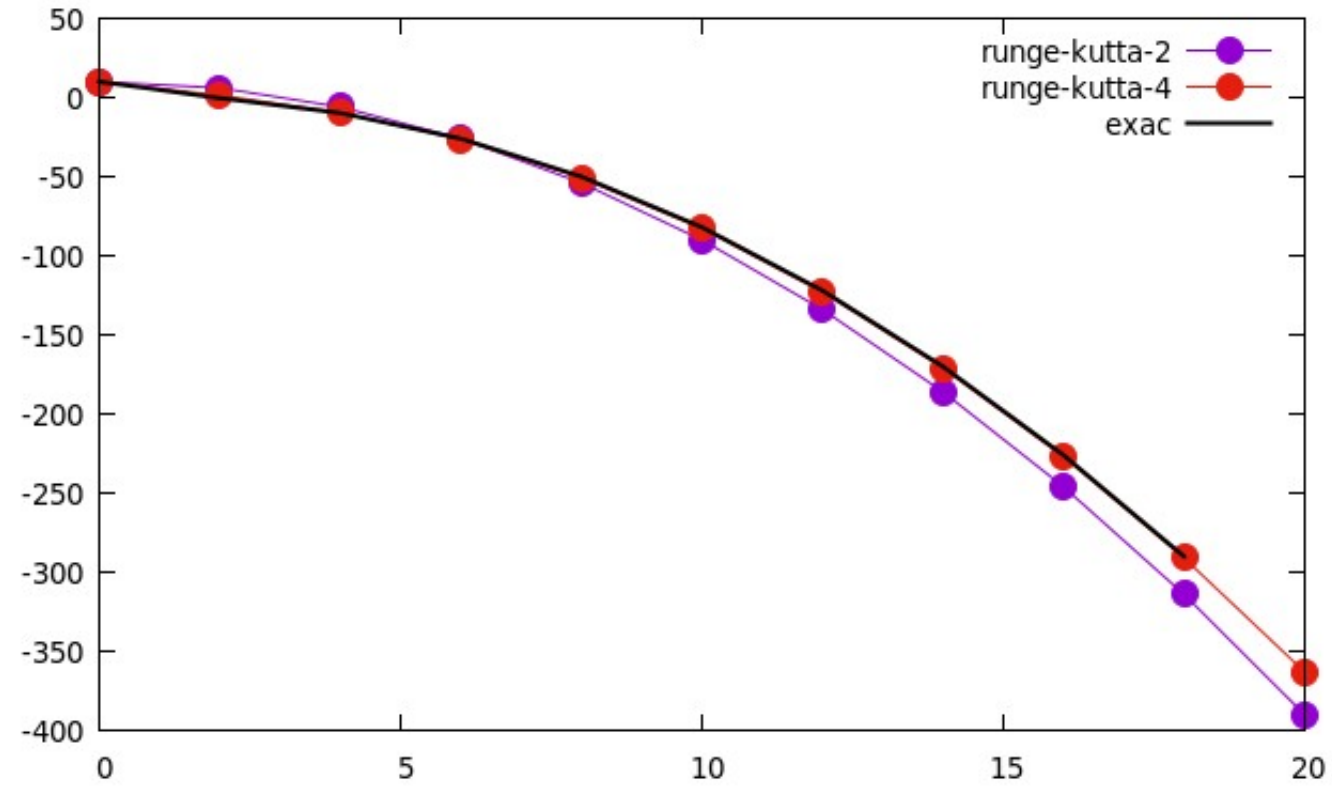
- **Тест №2**

  $f(x,y) = 3 - y - x,\ y(0) = 0$ Точное решение: $y(x) = 4 - x - 4e^{-x}$



- **Тест №3**

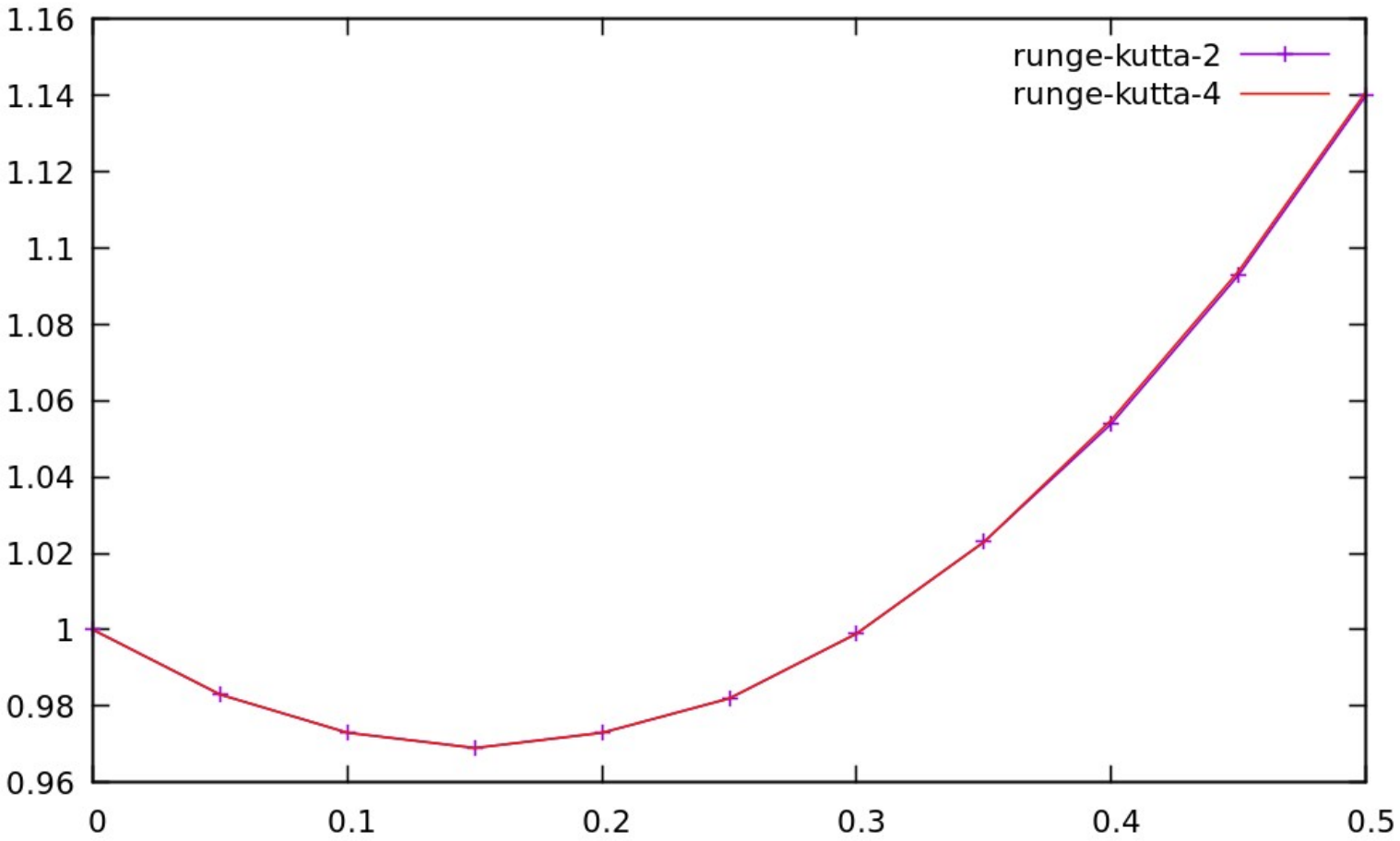
  $f(x,y) = -y - x^2,\ y(0) = 10$ Точное решение: $y(x) = -x^2 + 2x - 2 + 12e^{-x}$



- **Тест 4**

  Система дифференциальных уравнений:

$$\begin{cases} \frac{dy_1}{dx} = 2.4 \cdot v - u, & y_1(0) = 1 \\ \frac{dy_2}{dx} = e^{-u} - x + 2.2 \cdot v, & y_2(0) = 0.25 \end{cases}$$



Выводы

Были изучены методы Рунге-Кутта второго и четвертого порядка точности, применяемые для численного решения задачи Коши для дифференциального уравнения (или системы дифференциальных уравнений) первого порядка. Экспериментально было показано, что второй дает большую точность вычислений.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

enum
{
    NOSYS = 0,
    SYS = 1,
};

// начальные аргументы
const double e = 2.7182818284;
double nosys_args[2] = {0., 3.};
double sys_args[3] = {0., 1., 0.25,};

double f1(double x, double y);
double f1_exac(double x, double y);
double test2(double x, double y);
double test2_exac(double x, double y);
double test3(double x, double y);
double test3_exac(double x, double y);

double f1_sys(double x, double y, double z);
double f2_sys(double x, double y, double z);


void runge_kutta_2(double (*f)(double, double), double *x, double *y, double h, int n, FILE *out);
void runge_kutta_4(double (*f)(double, double), double *x, double *y, double h, int n, FILE *out);
void runge_kutta_sys_2(double (*f)(double, double, double), double (*g)(double, double, double), double *x, double *y, double *z, double h, int n, FILE *out);
void runge_kutta_sys_4(double (*f)(double, double, double), double (*g)(double, double, double), double *x, double *y, double *z, double h, int n, FILE *out);

int main(int argc, char **argv)
{
    int mode, accur;
    sscanf(argv[1], "%d", &mode);
    sscanf(argv[2], "%d", &accur);

    printf("Enter [a, b] and n:\n");
    double a, b; int n;
    scanf("%lf%lf%d", &a, &b, &n);
    double h = (b - a) / n;

    char name[128];
    sprintf(name, "table_%d_%d.txt", mode, accur);
    // файл для записи сетки и построения графика
    FILE *out = fopen(name, "w");

    // аргументы сетки сохраняются
    double *x = calloc(n + 1, sizeof(double));
    double *y = calloc(n + 1, sizeof(double));
    double *z = calloc(n + 1, sizeof(double));

    if (mode == NOSYS) {
        x[0] = nosys_args[0];
        y[0] = nosys_args[1];

        if (accur == 2)
            runge_kutta_2(&f1, x, y, h, n, out);
        if (accur == 4)
            runge_kutta_4(&f1, x, y, h, n, out);
    }

    if (mode == SYS) {
```

```
64          x[0] = sys_args[0];
65          y[0] = sys_args[1];
66          z[0] = sys_args[2];
67  printf("here\n");
68          if (accur == 2)
69              runge_kutta_sys_2(&f1_sys, &f2_sys, x, y, z, h, n, out);
70          if (accur == 4)
71              runge_kutta_sys_4(&f1_sys, &f2_sys, x, y, z, h, n, out);
72      }
73      return 0;
74  }
75
76  void runge_kutta_2(double (*f)(double, double), double *x, double *y, double
    h, int n, FILE *out)
77  {
78      for(int i = 0; i < n; i++) {
79          fprintf(out, "%9.3lf %9.3lf ", x[i], y[i]);
80          fprintf(out, "%9.3lf\n", f1_exac(x[i], y[i]));
81          double k1 = (*f)(x[i], y[i]);
82          double k2 = (*f)(x[i] + h, y[i] + h * k1); // пересчет tilda <y>
83          y[i + 1] = y[i] + h * (k1 + k2) / 2;  // добавка с полусуммой
84          x[i + 1] = x[i] + h;
85      }
86      fprintf(out, "%9.3lf %9.3lf\n", x[n], y[n]);
87      return;
88  }
89
90  void runge_kutta_4(double (*f)(double, double), double *x, double *y, double
    h, int n, FILE *out)
91  {
92      for(int i = 0; i < n; i++) {
93          fprintf(out, "%9.3lf %9.3lf ", x[i], y[i]);
94          fprintf(out, "%9.3lf\n", f1_exac(x[i], y[i]));
95          double k1 = (*f)(x[i], y[i]);
96          double k2 = (*f)(x[i] + h / 2, y[i] + h * k1 / 2);
97          double k3 = (*f)(x[i] + h / 2, y[i] + h * k2 / 2);
98          double k4 = (*f)(x[i] + h, y[i] + h * k3);
99          y[i + 1] = y[i] + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
100         x[i + 1] = x[i] + h;
101     }
102     fprintf(out, "%9.3lf %9.3lf\n", x[n], y[n]);
103     return;
104 }
105
106 void runge_kutta_sys_2(double (*f)(double, double, double), double (*g)
    (double, double, double), double *x, double *y, double *z, double h, int n,
    FILE *out)
107 {
108     for(int i = 0; i < n; i++) {
109         fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x[i], y[i], z[i]);
110         double k1 = (*f)(x[i], y[i], z[i]);
111         double m1 = (*g)(x[i], y[i], z[i]);
112         double k2 = (*f)(x[i] + h, y[i] + h * k1, z[i] + h * m1);
113         double m2 = (*g)(x[i] + h, y[i] + h * k1, z[i] + h * m1);
114         y[i + 1] = y[i] + h * (k1 + k2) / 2;
115         z[i + 1] = z[i] + h * (m1 + m2) / 2;
116         x[i + 1] = x[i] + h;
117     }
118     fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x[n], y[n], z[n]);
119     return;
120 }
121
122 void runge_kutta_sys_4(double (*f)(double, double, double), double (*g)
    (double, double, double), double *x, double *y, double *z, double h, int n,
    FILE *out)
123 {
124     for(int i = 0; i < n; i++) {
125         fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x[i], y[i], z[i]);
126         double k1 = (*f)(x[i], y[i], z[i]);
```

```
127             double m1 = (*g)(x[i], y[i], z[i]);
128             double k2 = (*f)(x[i] + h / 2, y[i] + h * k1 / 2, z[i] + h * m1 / 2);
129             double m2 = (*g)(x[i] + h / 2, y[i] + h * k1 / 2, z[i] + h * m1 / 2);
130             double k3 = (*f)(x[i] + h / 2, y[i] + h * k2 / 2, z[i] + h * m2 / 2);
131             double m3 = (*g)(x[i] + h / 2, y[i] + h * k2 / 2, z[i] + h * m2 / 2);
132             double k4 = (*f)(x[i] + h, y[i] + h * k3, z[i] + h * m3);
133             double m4 = (*g)(x[i] + h, y[i] + h * k3, z[i] + h * m3);
134             y[i + 1] = y[i] + (h / 6) * (k1 + 2 * k2 + 2 * k3 + k4);
135             z[i + 1] = z[i] + (h / 6) * (m1 + 2 * m2 + 2 * m3 + m4);
136             x[i + 1] = x[i] + h;
137         }
138         fprintf(out, "%9.3lf %9.3lf %9.3lf\n", x[n], y[n], z[n]);
139         return;
140     }
141
142     double f1(double x, double y) { return (y - y*y) * x; } //(0, 3)
143
144     double f1_exac(double x, double y)
145     {
146         double proc = x*x / 2;
147         return pow(e, proc) / (pow(e, proc) - 2./3);
148     }
149
150     double f1_sys(double x, double y, double z) { return 2.4 * z - y; }
151
152     double f2_sys(double x, double y, double z) { return pow(e, -1. * y) - x +
        2.2 * z; }
153
154     double test2(double x, double y) { return 3 - y - x; } // (0, 0)
155
156     double test2_exac(double x, double y) { return 4 - x - 4*pow(e, -1. * x); }
157
158     double test3(double x, double y) { return -1.*y - x*x; } // (0, 10)
159
160     double test3_exac(double x, double y) {return -1.*x*x +2*x - 2 + 12*pow(e,
        -1.*x);}
```