**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**имени М.В.Ломоносова**

**Факультет вычислительной математики и кибернетики**

**Компьютерный практикум по курсу**

**«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»**

**ЗАДАНИЕ № 2.**

**Часть 1**

**Численные методы решения дифференциальных уравнений**

**ОТЧЕТ**

**о выполненном задании**

студента 206 учебной группы факультета ВМК МГУ

Мукина Николая Михайловича

Москва, 2014

**Цель работы:**

Освоить 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:**

Решить задачу Коши для ОДУ 1-го порядка.

y` = f (x, y), x > xo (1)

y(xo) = yo (2)

Предполагается, что правая часть уравнения (1) функция f = f (x, y) такова, что гарантирует существование и единственность задачи Коши (1)-(2).

Решить задачу Коши для СОДУ 1-го порядка.

y`1 = f1 (x, y­1, y2),

y`2 = f2 (x, y1, y2), x > x0 (1)

y1(xo) = y1(0)

y2(xo) = y2(0)(2)

Так же предполагается, что правые части уравнений (3) заданы так, что гарантирует существование и единственность задачи Коши (3)-(4).

**Описание метода:**

**Метод Рунге-Кутта второго порядка.**

Рассмотрим [задачу Коши](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%9A%D0%BE%D1%88%D0%B8):

y` = f (x, y), x > xo y(xo) = yo

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

yn+1 = yn + (h / 2)(k1 + k2)

Вычисление нового значения проходит в четыре стадии:\

k1 = f(xn, yn)

k2 = f(xn + h, yn + h \* k1)

где h — величина шага сетки по x.

Этот метод имеет второй порядок точности, то есть суммарная ошибка на конечном интервале интегрирования имеет порядок O(h2)

**Метод Рунге-Кутта четвёртого порядка.**

Рассмотрим [задачу Коши](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%9A%D0%BE%D1%88%D0%B8):

y` = f (x, y), x > xo y(xo) = yo

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

yn+1 = yn + (h / 2)(k1 + 2k2 +2k3 + k4)

Вычисление нового значения проходит в четыре стадии:

k1 = f(xn, yn)

k2 = f(xn + h/2, yn + h/2 \* k1)

k3 = f(xn + h/2, yn + h/2 \* k2)

k4 = f(xn + h, yn + h \* k3)

где h — величина шага сетки по x.

Этот метод имеет четвёртый порядок точности, то есть суммарная ошибка на конечном интервале интегрирования имеет порядок O(h­4)

Метод Рунге-Кутта для СОДУ аналогичен методу для ОДУ.

**Тестирование:**

ОДУ 1-го порядка.

Вариант 1.

y` = (3 - x - y)

y(0) = 0

h = 0.2 n = 5

|  |  |  |  |
| --- | --- | --- | --- |
| x | y(2-ой порядок) | y(4-ый порядок) | Точное значение |
| 0 | 0 | 0 | 0 |
| 0.2 | 0.52 | 0.52506667 | 0.52507699 |
| 0.4 | 0.9104 | 0.91870292 | 0.91871982 |
| 0.6 | 1.194528 | 1.2047327 | 1.2047535 |
| 0.8 | 1.391513 | 1.4026615 | 1.4026841 |
| 1 | 1.5170406 | 1.528459 | 1.5284822 |

h = 0.1 n = 10

|  |  |  |  |
| --- | --- | --- | --- |
| x | y(2-ой порядок) | y(4-ый порядок) | Точное значение |
| 0 | 0 | 0 | 0 |
| 0.1 | 0.28 | 0.28065 | 0.28065033 |
| 0.2 | 0.5239 | 0.52507639 | 0.52507699 |
| 0.3 | 0.7351295 | 0.73672631 | 0.73672712 |
| 0.4 | 0.9167922 | 1.4026615 | 1.4026841 |
| 0.5 | 1.0716969 | 1.0738763 | 1.0738774 |
| 0.6 | 1.2023857 | 1.2047523 | 1.2047535 |
| 0.7 | 1.3111591 | 1.3136575 | 1.3136588 |
| 0.8 | 1.400099 | 1.4026828 | 1.4026841 |
| 0.9 | 1.4710896 | 1.47372 | 1.4737214 |
| 1 | 1.5258361 | 1.5284809 | 1.5284822 |

h = 0.05 n = 10

|  |  |  |  |
| --- | --- | --- | --- |
| x | y(2-ой порядок) | y(4-ый порядок) | Точное значение |
| 0 | 0 | 0 | 0 |
| 0.05 | 0.145 | 0.14508229 | 0.1450823 |
| 0.1 | 0.28049375 | 0.28065031 | 0.28065033 |
| 0.15 | 0.40694468 | 0.40716807 | 0.40716809 |
| 0.2 | 0.52479363 | 0.52507695 | 0.52507699 |
| 0.25 | 0.63445994 | 0.63479683 | 0.63479687 |
| 0.3 | 0.73634252 | 0.73672707 | 0.73672712 |
| 0.35 | 0.83082082 | 0.83124759 | 0.83124764 |
| 0.4 | 0.9182558 | 0.91871976 | 0.91871982 |
| 0.45 | 0.99899083 | 0.99948733 | 0.99948739 |
| 0.5 | 1.0733525 | 1.0738773 | 1.0738774 |

СОДУ 1-го порядка.

Вариант 9.

y`1 = 2.1y2 - y1­2

y`2 = (exp( -y1 )) + x + 2.1y2

y1(xo) = 1

y2(xo) = 0.25

h = 0.2 n = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 2-ой порядок | | 4-ый порядок | |
| y1 | y2 | y1 | y2 |
| 0 | 1 | 0.35 | 1 | 0.25 |
| 0.2 | 0.9031475 | 0.48006055 | 0.90342531 | 0.46392052 |
| 0.4 | 0.94262018 | 0.87933014 | 0.93591103 | 0.84165636 |
| 0.6 | 1.1346758 | 1.5213255 | 1.116163 | 1.4560839 |
| 0.8 | 1.4951235 | 2.5194348 | 1.4654937 | 2.4207093 |
| 1 | 2.0144634 | 4.0515919 | 1.9876239 | 3.9156658 |

h = 0.1 n = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 2-ой порядок | | 4-ый порядок | |
| y1 | y2 | y1 | y2 |
| 0 | 1 | 0.35 | 1 | 0.25 |
| 0.1 | 0.95214969 | 0.35209201 | 0.9521856 | 0.34754359 |
| 0.2 | 0.93525253 | 0.49026289 | 0.9343193 | 0.48026744 |
| 0.3 | 0.95090371 | 0.67155979 | 0.94803202 | 0.6550809 |
| 0.4 | 1.0018829 | 0.9045873 | 0.99613492 | 0.88042601 |
| 0.5 | 1.0914991 | 1.1999734 | 1.0820125 | 1.1667365 |
| 0.6 | 1.2229226 | 1.5709884 | 1.2090038 | 1.5270511 |
| 0.7 | 1.3985311 | 2.0343226 | 1.3797927 | 1.9777875 |
| 0.8 | 1.6193949 | 2.6110054 | 1.5959194 | 2.5396651 |
| 0.9 | 1.8850818 | 3.3274478 | 1.8575786 | 3.2387562 |
| 1 | 2.1939184 | 4.2166144 | 2.1638335 | 4.1076686 |

h = 0.05 n = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| x | 2-ой порядок | | 4-ый порядок | |
| y1 | y2 | y1 | y2 |
| 0 | 1 | 0.35 | 1 | 0.25 |
| 0.05 | 0.97617652 | 0.29783622 | 0.97618107 | 0.2966408 |
| 0.1 | 0.95973616 | 0.35397579 | 0.95961626 | 0.3514622 |
| 0.15 | 0.95080686 | 0.41919263 | 0.95043379 | 0.41522785 |
| 0.2 | 0.94961448 | 0.49433758 | 0.94885863 | 0.48877736 |
| 0.25 | 0.9564646 | 0.58034837 | 0.95519494 | 0.57303608 |
| 0.3 | 0.97172524 | 0.67826144 | 0.96980948 | 0.66902671 |
| 0.35 | 0.99580974 | 0.78922598 | 0.99311506 | 0.77788311 |
| 0.4 | 1.0291591 | 0.91452016 | 1.0255536 | 0.90086629 |
| 0.45 | 1.0722238 | 1.05557 | 1.0675785 | 1.039383 |
| 0.5 | 1.125445 | 1.2139705 | 1.1196365 | 1.1950065 |

**Вывод:**

Метод Рунге-Кутта позволяет численно решить ОДУ 1-го порядка с заданной наперед точностью. Метод 4-го порядка дает по точности результат лучше, чем метод 2-го порядка, но производит больше вычислений.

**Исходный код:**

Для ОДУ:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

double f(double x, double y){

return (3 - x - y);

}

double df(double x){

return (4 - x - 4 \* exp(-x));

}

int main(){

int i, n;

double h;

printf("Enter h & n\n");

scanf("%lg", &h);

scanf("%d", &n);

++n;

double \*x = malloc (n \* sizeof(double));

x[0] = 0.0;

for (i = 1; i < n; ++i){

x[i] = x[i - 1] + h;

}

double \*y2 = malloc (n \* sizeof(double));

double \*y4 = malloc (n \* sizeof(double));

y2[0] = y4[0] = 0;

for (i = 0; i < n - 1; ++i){

double k1, k2, k3, k4;

k1 = f(x[i], y2[i]);

k2 = f(x[i] + h, y2[i] + h \* k1);

y2[i + 1] = y2[i] + (h / 2) \* (k1 + k2);

k1 = f(x[i], y4[i]);

k2 = f(x[i] + h / 2, y4[i] + (h / 2) \* k1);

k3 = f(x[i] + h / 2, y4[i] + (h / 2) \* k2);

k4 = f(x[i] + h, y4[i]+ h \* k3);

y4[i + 1] = y4[i] + (h / 6)\*(k1 + 2 \* k2 + 2 \* k3 + k4);

}

for (i = 0; i < n; ++i){

printf (" %10.8lg | %10.8lg | %10.8lg | %10.8g \n", x[i], y2[i], y4[i], df(x[i]));

}

return 0;

}

Для ОДУ:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

double f1(double x, double u, double v){

return (2.1 \* v - u \* u);

}

double f2(double x, double u, double v){

return (exp( -u )) + x + 2.1 \* v;

}

int main(){

int i, n;

double h;

printf("Enter h & n\n");

scanf("%lg", &h);

scanf("%d", &n);

++n;

double \*x = malloc (n \* sizeof(double));

x[0] = 0.0;

for (i = 1; i < n; ++i){

x[i] = x[i - 1] + h;

}

double \*\*y2 = malloc (2 \* sizeof(double \*));

double \*\*y4 = malloc (2 \* sizeof(double \*));

y2[0] = malloc (n \* sizeof(double));

y4[0] = malloc (n \* sizeof(double));

y2[1] = malloc (n \* sizeof(double));

y4[1] = malloc (n \* sizeof(double));

y2[0][0] = y4[0][0] = 1;

y2[1][0] = y4[1][0] = 0.25;

for (i = 0; i < n - 1; ++i){

double k1, k2, k3, k4;

k1 = f1(x[i], y2[0][i], y2[1][i]);

k2 = f1(x[i] + h, y2[0][i] + h\* k1, y2[1][i] + h \* k1);

y2[0][i + 1] = y2[0][i] + (h / 2) \* (k1 + k2);

k1 = f2(x[i], y2[0][i], y2[1][i]);

k2 = f2(x[i] + h, y2[0][i] + h\* k1, y2[1][i] + h \* k1);

y2[1][i + 1] = y2[1][i] + (h / 2) \* (k1 + k2);

k1 = f1(x[i], y4[0][i], y4[1][i]);

k2 = f1(x[i], y4[0][i] + (h / 2) \* k1, y4[1][i] + (h / 2) \* k1);

k3 = f1(x[i], y4[0][i] + (h / 2) \* k2, y4[1][i] + (h / 2) \* k2);

k4 = f1(x[i], y4[0][i] + h \* k3, y4[1][i] + h \* k3);

y4[0][i + 1] = y4[0][i] + (h / 6)\*(k1 + 2 \* k2 + 2 \* k3 + k4);

k1 = f2(x[i], y4[0][i], y4[1][i]);

k2 = f2(x[i], y4[0][i] + (h / 2) \* k1, y4[1][i] + (h / 2) \* k1);

k3 = f2(x[i], y4[0][i] + (h / 2) \* k2, y4[1][i] + (h / 2) \* k2);

k4 = f2(x[i], y4[0][i] + h \* k3, y4[1][i] + h \* k3);

y4[1][i + 1] = y4[1][i] + (h / 6)\*(k1 + 2 \* k2 + 2 \* k3 + k4);

}

for (i = 0; i < n; ++i){

printf (" %10.8lg || %10.8lg | %10.8lg || %10.8g | %10.8lg \n",

x[i], y2[0][i], y2[1][i], y4[0][i], y4[1][i]);

}

return 0;

}