МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО
ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГУ»)

Лабораторная работа №2
Тема: «Решение задачи Коши с заданной точностью методом Рунге-Кутта»

Выполнил студент 3 курса 8 группы: Кураков Н.П.
Преподаватель: Шабунина З.А.

ВОРОНЕЖ 2019

1. Постановка задачи

Назначение:

Интегрирование обыкновенного дифференциального уравнения вида $y'=f(x,y), x\in[A,B]$ с начальным условием $y(x_0)=y_0$, где точка x_0 совпадает либо с началом, либо с концом отрезка интегрирования.

Вид метода Рунге-Кутта (метод второго порядка):

$y_1=y_0+K2$

$K1=h*f(x_0, y_0)$

$K2=h*f(x_0+\frac{h}{2}, y_0+\frac{K1}{2})$

Входные данные (входной файл):

A, B – отрезок интегрирования

x_0, y_0 – начальная точка

h_{min} – минимально допустимый шаг

eps_{min} – наиболее допустимое значение абсолютной точности

Выходные данные (выходной файл):

Вторая и последующие строки содержат: X, Y, EPS - локальная погрешность, h - шаг с которым были получены данные значения.

Последние строки содержат: общее количество вычисленных точек, количество точек в которых не достигнута точность.

2. Метод решения

1. Вывод погрешности на шаге

Пусть $f(x,y)=f(x)$, тогда

$$u(x_0+h)-y_0-K2=y_0+h*f_0+\frac{h^2}{2!}*f_0'+\frac{h^3}{3!}*f_0''+\frac{h^4}{4!}*f_0'''+...-y_0-h*f_0-\frac{h^2}{2}*f_0'-\frac{h^3}{4}*\frac{1}{2}*f_0''-...$$

=

$$\frac{1}{24}*h^3*f''+O(h^4)$$

2. Оценка локальной погрешности по правилу Рунге

Суть метода заключается: по одной и той же выбранной вычислительной формуле считаются два приближения к решению в одной очке, но с разными шагами. Обозначим:

$y_h=y(x+h)$ - значение функции в точке $(x+h)$

$y_{\frac{h}{2}}=y(x+\frac{h}{2})$ - значение функции в точке $(x+\frac{h}{2})$

Переприсвоим начальный x: $x_{\frac{h}{2}}=x+\frac{h}{2}$, $y_h=y_{\frac{h}{2}+\frac{h}{2}}=y(x_{\frac{h}{2}}+\frac{h}{2})$

Главная часть погрешности метода на шаге равна $\frac{y_{\frac{h}{2}}-y_h}{(\frac{1}{2})^s-1}$, где s=2

3. Автоматический выбор шага интегрирования задачи Коши. Метод давления и удвоения шага на два

Обозначим:

eps_{n+1} - локальная погрешность метода на шаге x_h+h

y_{n+1}^{h} - приближённое значение, вычисленное с данным шагом

Пусть наибольшая допустимая погрешность eps > 0

Если $|eps_n + 1| > eps$, то приближённое значение y_{n+1}^{h} считается неудовлетворительным по точности и выбирается новое значение шага $h^{(1)} = \frac{h}{2}$

С новым шагом по той же формуле Рунге-Кутта вычисляется новое значение

Так происходит до тех пор, пока локальная погрешность не станет меньше или равна eps, либо не достигнем минимального шага (в таком случае отмечаем, что в этой точке не была достигнута необходимая точность, и шаг выставляется $h = h_{min}$)

Если $|eps_{n+1}| < \frac{eps}{2^s}$, то шаг интегрирования удваивается $h_{n+1} = 2 * h_n$, иначе остаётся таким же. Для оптимизации удвоение шага за один раз ограничено значением 5

Код Алгоритма

```
runge(h * direct());     // делаем оценку погрешности с заданным шагом
if (eps <= epsmax)       // если точность слишком высокая  (eps <= epsmax/8)
    while (eps <= epsmax)          //(eps <= epsmax/8)
        if (h * 2 <= hmax) { // наибольший шаг взят условно, можно отключить
            if (lastdiv) { // не допускаем умножения после деления
                lastdiv = false;
                break;
            }
            if (multiplyLimit <= 5)
                multiplyLimit++;
            else  // если достигли предела для умножения
            {
                multiplyLimit = 0;
                break;
            }
            h *= 2;
            runge(h * direct());
            // если при умножении вышли за макс погрешность, то возвращаемся назад
            if (eps > epsmax) {
                h /= 2;
                break;
            }
        }
        else
            break;
else
if (eps > epsmax)   // если точность слишком низкая
    while (eps > epsmax)
        if (h / 2 >= hmin) {
            lastdiv = true;
            h /= 2;
            runge(h * direct());
        }
        else { // если не удалось достичь точности
            h = hmin; // устанавливаем мин шаг
            break;
        }
```

4. Проверка на конец интервала

За два шага вперед проверяется точка конца интервала интегрирования с тем, чтобы исправить при необходимости величину шага, чтобы достигнуть конца отрезка интегрирования без слишком резких изменений в величине шага.

Для каждого вычисленного шага h_n делается проверка на конец интервала. Пусть интегрирование происходит слева направо, тогда проверяется выполнение неравенства $B-(x_n+h_n) < h_{min}$. Если оно не удовлетворяется, то следующей точкой назначается x_n+h_n. Если неравенство справедливо, то для достижения конца отрезка необходимо сделать один или два шага, что регламентируется следующим правилом:

1. Если $B-x_n \geq 2h_{min}$, то делается два шага. $x_{n+1}=b-h_{min}, x_{n+2}=B$
2. Если $B-x_n \leq 1.5h_{min}$, то выполняется один шаг. $x_{n+1}=B$
3. Если $1.5h_{min} < B-x_n < 2h_{min}$, то делается два шага

$x_{n+1}=x_n+(B-x_n)/2, x_{n+2}=B$

```
if ((direction ? x-h-A : B-x-h) < hmin || x + h == x) // Если
после текущего шага B-x будет < hmin
        break;
    step();
    pointsCount++;
}
// обрабатываем состояние, когда находимся у конца отрезка
// выбор направления - direction ? [справа налево] : [слева направо]
if ((direction ? x-A : B-x) >= 2*hmin) { // если больше чем за 2 мин шага от
края
    h = direction ? x - hmin - A : B - hmin - x;
    step();
    lastStep();
    pointsCount++;
}
else
if ((direction ? x-A : B-x ) <= 1.5*hmin) // если меньше чем за 1.5 мин шага
    lastStep();
else { // если 1.5*hmin < остаток(B-x) < 2*hmin
    h = direction ? (x-A)/2.0 : (B-x)/2.0;
    step();
    lastStep();
    pointsCount++;
}
```

3. Основные процедуры

Задача реализована на языке Java. RungeKutta — класс, реализующий решение задачи Коши с заданной точностью методом Рунге-Кутта, с автоматическим выбором шага интегрирования.

1) rungeRule(double yh1, double yh2) - оценка погрешности по правилу Рунге, параметрами являются вычисленные значения, интегрированные с шагом h и с шагом h/2. Функция возвращает оценку погрешности на шаге h.

2) runge(double h) - Вычисление Y_{i+1} значения + оценка погрешности по правилу рунге (использую функцию из п.1). Возвращает Y_{i+1} и сохраняет оценку погрешности.

3) step() - объединяет функцию из п.2, увеличение X = X + h, и вывод вычисленных значений в файл.

4) start() - основная функция, осуществляет полный цикл действий (автовыбор шага, интегрирование, проверка на конец интервала и т. д.) с использованием функций 1)-3).

4. Тестирование

1. $y = x,\ y' = 1,\ y'' = 0$ Начальные условия: [A, B] = [0, 1], x=0, y=0, h=0.001, R=0
Результат:

X	Y	EPS	h
0,000000000000000	0,000000000000000	0.0	
0,100000000000000	0,100000000000000	0.0	0,100000000000000
0,200000000000000	0,200000000000000	0.0	0,100000000000000
0,300000000000000	0,300000000000000	0.0	0,100000000000000
0,400000000000000	0,400000000000000	0.0	0,100000000000000
0,500000000000000	0,500000000000000	0.0	0,100000000000000
0,600000000000000	0,600000000000000	0.0	0,100000000000000
0,700000000000000	0,700000000000000	0.0	0,100000000000000
0,800000000000000	0,800000000000000	0.0	0,100000000000000
0,900000000000000	0,900000000000000	0.0	0,100000000000000
0,999000000000000	0,999000000000000	0.0	0,099000000000000
1,000000000000000	1,000000000000000	0.0	0,001000000000000
Points count: 12			
Accuracy bad in: 0			

2. $y = 12 * x^{2},\ y' = 24 * x,\ y'' = 24$ Начальные условия: [A, B] = [0, 1], x=0, y=0, h=0.001, R=0
Результат:

X	Y	EPS	h
0,000000000000000	0,000000000000000	0.0	
0,100000000000000	0,120000000000000	0.0	0,100000000000000
0,200000000000000	0,480000000000000	0.0	0,100000000000000
0,300000000000000	1,080000000000000	0.0	0,100000000000000
0,400000000000000	1,920000000000000	0.0	0,100000000000000
0,500000000000000	3,000000000000000	0.0	0,100000000000000
0,600000000000000	4,320000000000000	0.0	0,100000000000000
0,700000000000000	5,880000000000001	0.0	0,100000000000000
0,800000000000000	7,680000000000001	0.0	0,100000000000000
0,900000000000000	9,720000000000000	0.0	0,100000000000000
0,999000000000000	11,976012000000003	0.0	0,099000000000000
1,000000000000000	12,000000000000002	0.0	0,001000000000000
Points count: 12			
Accuracy bad in: 0			

3. $y = 12 * x^{2},\ y' = 24 * x,\ y'' = 24$ Начальные условия: [A, B] = [1, 2], x=1, y=12, h=0.001, R=0
Результат:

X	Y	EPS	h
1,000000000000000	12,000000000000000	0.0	
1,100000000000000	14,520000000000000	0.0	0,100000000000000
1,200000000000000	17,280000000000000	0.0	0,100000000000000
1,300000000000000	20,280000000000000	0.0	0,100000000000000
1,400000000000000	23,520000000000003	0.0	0,100000000000000
1,500000000000000	27,000000000000004	0.0	0,100000000000000
1,600000000000001	30,720000000000006	0.0	0,100000000000000
1,700000000000001	34,680000000000010	0.0	0,100000000000000
1,800000000000001	38,880000000000010	0.0	0,100000000000000
1,900000000000001	43,320000000000014	0.0	0,100000000000000
1,999000000000000	47,952011999999980	0.0	0,098999999999999
2,000000000000000	47,999999999999980	0.0	0,001000000000000
Points count: 12			
Accuracy bad in: 0			

4. $y = 12 * x^2 ; y' = 24 * x ; y'' = 24$ Начальные условия: [A, B] = [0, 1], x=0, y=0, h=0.001

```
X                     Y                     EPS                         h
0,000000000000000     0,000000000000000     0.0
0,999000000000000     11,976011999999999    0.0       0,999000000000000
1,000000000000000     11,999999999999998    0.0       0,001000000000000
Points count:     3
Accuracy bad in:   0
```

5. $y = 4 * x^3 ; y' = 12 * x^2 ; y'' = 24 * x ; y''' = 24$ Начальные условия: [A, B] = [0, 1], x=0, y=0, h=0.001, eps=4.768372e-7

```
X                 Y                 EPS                 h
0,000000000000000     0,000000000000000     0.0
0,007812500000000     0,000001430511475     4.76837158203125E-7       0,007812500000000
0,015625000000000     0,000014305114746     4.76837158203125E-7       0,007812500000000
0,023437500000000     0,000050067901611     4.76837158203125E-7       0,007812500000000
0,031250000000000     0,000120162963867     4.76837158203125E-7       0,007812500000000
0,039062500000000     0,000236034393311     4.76837158203125E-7       0,007812500000000
0,046875000000000     0,000409126281738     4.76837158203125E-7       0,007812500000000
0,054687500000000     0,000650882720947     4.76837158203125E-7       0,007812500000000
0,062500000000000     0,000972747802734     4.76837158203125E-7       0,007812500000000
0,070312500000000     0,001386165618896     4.76837158203125E-7       0,007812500000000
0,078125000000000     0,001902580261230     4.76837158203125E-7       0,007812500000000
0,085937500000000     0,002533435821533     4.76837158203125E-7       0,007812500000000
0,093750000000000     0,003290176391602     4.76837158203125E-7       0,007812500000000
0,101562500000000     0,004184246063232     4.76837158203125E-7       0,007812500000000
. . . . .
0,953125000000000     3,463397026062012     4.76837158203125E-7       0,007812500000000
0,960937500000000     3,549263477325440     4.76837158203125E-7       0,007812500000000
0,968750000000000     3,636537551879883     4.76837158203125E-7       0,007812500000000
0,976562500000000     3,725230693817139     4.76837158203125E-7       0,007812500000000
0,984375000000000     3,815354347229004     4.76837158203125E-7       0,007812500000000
0,992187500000000     3,906919956207275     4.76837158203125E-7       0,007812500000000
0,999000000000000     3,987951121511719     3.161691897920112E-7      0,006812500000000
1,000000000000000     3,999939124511719     1.000000082740371E-9      0,001000000000000
Points count:     130
Accuracy bad in:   0
```