Федеральное агентство связи
Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

Кафедра ПМиК

Курсовая работа
по дисциплине «Вычислительная математика»
по теме «Решение краевой задачи методом Рунге-Кутта IV порядка»

Выполнил: студент группы ИА-831
Зарубин Максим Евгеньевич
Проверил: ассистент кафедры ПМиК
Петухова Яна Владимировна

Новосибирск

Содержание

Постановка задачи

Решить краевую задачу методом Рунге-Кутта IV порядка:

$$y'' = \frac{y + e^x}{2}$$

$y(0) = 1;$

$y(1) = 2.7182828$

Теоретические сведения

ДУ высших порядков часто бывает необходимо решить не задачу Коши, а так называемую краевую задачу, т.е. начальные условия, которые заданы в разных точках.

Рассмотрим простейшую краевую задачу для ДУ 2го порядка:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = y_0 \\ y(b) = y_1 \end{cases} \quad (1),$$

А мы умеем решать:

$$\begin{cases} y'' = f(x, y, y') \\ y(a) = y_0 \\ y'(a) = y_0' = k \end{cases} \quad (2),$$

В (2) нам известно $y'(a)$, поэтому для решения задачи (1) мы будем подбирать $y'(a)$ в (2), с тем, чтобы y(b) = y_1

- Метод стрельб

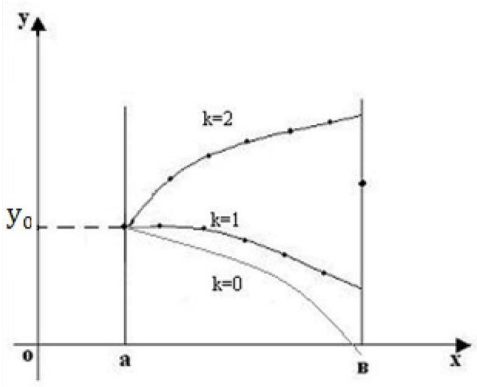

Рис 1. Метод стрельб

После пристрелки и определения интервала [a,b], где идёт смена знака, запускаем МПД или МХ.

На практике это выглядит так, как будто решаем уравнение $q(k) = y_1$, где $q(k)$ возвращает решение задачи Коши (2) в точке b при заданном k.

- Метод Рунге-Кутта 4 порядка

Наиболее применяемым методом решения ДУ и СДУ является метод Рунге-Кутта 4-го порядка.

Формулы метода Рунге-Кутта 4-го порядка:

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

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

В векторной форме данной формулы, величины y, f, k заменяют на Y, F, K.

- Оценка погрешности решения ДУ и СДУ методом двойного пересчета

Используя такую же идею, как и в численном интегрировании, находим решение ДУ на [a,b] дважды с шагом h и с шагом h/2.

Сравниваем попарно, если расхождение между $|y_{h(k)} - y_{h/2(k)}| < 3\varepsilon$ для метода 2-го порядка, $|y_{h(k)} - y_{h/2(k)}| < 15\varepsilon$ для метода 4-го порядка, то в качестве точного решения берём $y_{h/2}$. Если же точность не достигнута, то шаг h уменьшаем вдвое и т.д., пока она не будет достигнута.

Метод двойного пересчёта при решении ДУ и СДУ практически единственный имеет возможность для оценки погрешностей, так как иные формулы очень сложны и требуют оценок различных производных.

Как и при ЧИ, при решении ДУ и СДУ после 2-го пересчёта в качестве точного решения выгодно брать не $Y_{h/2}$, а $Y_{кор}$.

$Y_{кор} = Y_{h/2} + \frac{1}{3}(Y_{h/2} + Y_h)$ - для второго порядка

Метод двойного пересчёта применим не только лишь при ЧИ, при решении ДУ и СДУ, но и при решении других численных методов.

Программная реализация

Для реализации работы программы были разработаны следующие функции:

double f(double, double, double) – возвращает значение второй производной;

double g(double, double, double) –возвращает значение первой производной;

double RK4(double, int, double[], double) – реализует решение дифференциального уравнения с помощью метода Рунге-Кутта 4 порядка. Возвращает значение y.

double RK4DP(double) – реализация двойного пересчета для метода Рунге-Кутта. Принимает значение первой производной. Возвращает значение y.

void shooting( ) – реализация метода стрельбы.

Результаты работы программы

```
      X                Y                Y'
H = 0.2
0,0000000       1,0000000       1,0000018
0,2000000       1,2214015       1,2214041
0,4000000       1,4918217       1,4918254
0,6000000       1,8221135       1,8221185
0,8000000       2,2255326       2,2255392
1,0000000       2,7182694       2,7182783
h = 0.1
0,0000000       1,0000000       1,0000018
0,1000000       1,1051710       1,1051727
0,2000000       1,2214030       1,2214045
0,3000000       1,3498592       1,3498606
0,4000000       1,4918252       1,4918264
0,5000000       1,6487218       1,6487230
0,6000000       1,8221195       1,8221206
0,7000000       2,0137535       2,0137545
0,8000000       2,2255418       2,2255427
0,9000000       2,4596040       2,4596049
1,0000000       2,7182828       2,7182837

y подсчитан, shot: 2,7182828, m3 = 1,0000018
```

Листинг

```
#include <iostream>

using namespace std;

double x0 = 0, xn = 1, yo = 1, z0 = 0.6, yk = 2.7182828;
double m1 = z0, m2 = 1.3, m3, H = 0.2, h = H/2;
double *yy, *YY, shot1, shot2, shot, eps = 0.001;


double f(double x, double y, double z);
double g(double x, double y, double z);
double RK4(double h, int range, double yy[], double z0);
double RK4DP(double m);
void shooting();

int main() {
	setlocale(LC_ALL, "Russian");
	shooting();
	system("pause");
	return 0;
}

double f(double x, double y, double z) {
	double result = (exp(x) + y) / 2;
	return result;
}

double g(double x, double y, double z) {
	return (z);
}

void shooting() {
	shot1 = RK4DP(m1);
	shot2 = RK4DP(m2);
	if (abs(shot1 - yk) < eps) printf("y подсчитан, shot1: %0.7f", shot1);
	else if (abs(shot2 - yk) < eps) printf("y подсчитан, shot2: %0.7f",
shot2);
	else {
		m3 = m2 + (((m2 - m1) * (yk - shot2)) / ((shot2 - shot1)));
		shot = RK4DP(m3);
	}
	while (abs(shot - yk) >= eps) {
		m1 = m2;
		m2 = m3;
		shot1 = shot2;
		shot2 = shot;
		m3 = m2 + (((m2 - m1) * (yk - shot2)) / ((shot2 - shot1)));
		z0 = m3;
		shot = RK4DP(m3);
	}
	printf("y подсчитан, shot: %0.7f, m3 = %0.7f", shot, m3);
```

```
}

double RK4DP(double m) {
	int r, R;
	cout << "	X\t\tY\t\tY'" << endl;
	double result;
	do {
		R = (int)((xn - x0) / h);
		r = (int)((xn - x0) / H);
		YY = new double[r];
		cout << "H = " << H << endl;
		RK4(H, r, YY, m);
		yy = new double[R];
		cout << "h = " << h << endl;
		result = RK4(h, R, yy, m);
		cout << endl;
		h *= 0.5;
		H = 2 * h;
	} while (abs(yy[R - 1] - YY[r - 1]) > eps);
	h = 0.1; H = 2 * h;
	return result;
}

double RK4(double h, int r, double yy[], double z0) {
	double xc = x0, yc = yo, zc = z0, y1c = 1, k1, k2, k4, k3, k11, k22,
k44, k33;
	for (int i = 0; i < r; i++) {
		printf("%0.7f\t%0.7f\t%0.7f\t\n", xc, y1c, zc);
		k1 = h * f(xc, yc, zc);
		k11 = h * g(xc, yc, zc);
		k2 = h * f(xc + h / 2.0, yc + k11 / 2.0, zc + k1 / 2.0);
		k22 = h * g(xc + h / 2.0, yc + k11 / 2.0, zc + k1 / 2.0);
		k3 = h * f(xc + h / 2.0, yc + k22 / 2.0, zc + k2 / 2.0);
		k33 = h * g(xc + h / 2.0, yc + k22 / 2.0, zc + k2 / 2.0);
		k4 = h * f(xc + h, yc + k33, zc + k3);
		k44 = h * g(xc + h, yc + k33, zc + k3);
		zc = zc + (k1 + 2.0 * k2 + 2.0 * k3 + k4) / 6.0;//y'
		yy[i] = y1c = yc + (k11 + 2.0 * k22 + 2.0 * k33 + k44) / 6.0;//y
		yc = y1c;
		xc += h;
	}
	printf("%0.7f\t%0.7f\t%0.7f\t\n", xc, y1c, zc);
	return y1c;
}
```