**Федеральное государственное бюджетное образовательное учреждение «Сибирский государственный университет телекоммуникаций и информатики»**

Кафедра ПМ и К

**КУРСОВАЯ РАБОТА**

По дисциплине «Вычислительная математика»

Вариант 9

**Выполнил**:

студент гр. ИВ-621

Сенченко А. П.

**Проверил**:

Чирихин К. С.

Новосибирск, 2018 г.

Оглавление

[Постановка задачи 3](#_Toc515035010)

[Описание алгоритма 4](#_Toc515035011)

[Результат работы программы 5](#_Toc515035012)

[Заключение 6](#_Toc515035013)

[Литература 6](#_Toc515035014)

[Листинг 7](#_Toc515035015)

Постановка задачи

Решить краевую задачу методом Рунге-Кутта II порядка с усреднением по производной.

Построить графики функции y(x) и кубического сплайна S(x) (интерполяция по точкам x=0; 0.2;0.4; 0.6; 0.8; 1.0). Найти интеграл

Описание алгоритма

Этапы решения краевой задачи:

1. С помощью метода стрельбы находим значение первой производной.
2. Решаем задачу Коши методом Рунге-Кутта II порядка с усреднением по времени.

Метод стрельбы:

Выбираем параметры: a ─ y(0) из краевой задачи, а b ─ произвольно; для того, чтобы найти отрезок, в котором будет искомое значение; корректируем исходные параметры в зависимости от перелёта или недолёта (y(a) – y1 > 0 или y(a) – y1 < 0 соответственно). Как только по a перелет, а по b недолет, останавливаем корректировку, на данном этапе отрезок найден. Для нахождения первой производной остается решить нелинейное уравнение любым известным способом, в частности методом бисекции: y(b) = y2, где y(b) – решение задачи Коши.

Задача Коши:

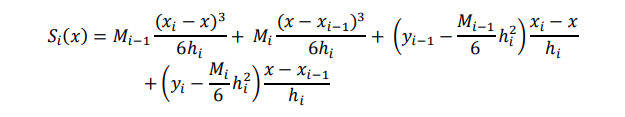
Применяем метод Рунге-Кутта:



Так как по условию дано уравнение, которое не может быть разрешено относительно старшей производной, то каждый раз решаем нелинейное уравнение относительно старшей производной.

Интерполяция:

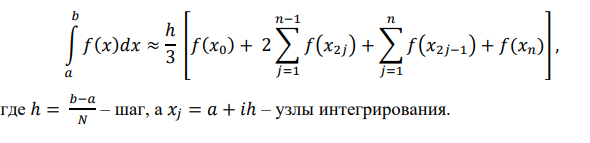
Для вычисления кубического сплайна на заданной сетке будем использовать формулу:



В данной задаче подразумевается интерполяция естественным кубическим сплайном, т. е. 𝑀0 = 𝑀𝑛. Шаг для сетки: h = const. Чтобы найти другие 𝑀𝑖 составим СЛАУ; получим трехдиагональную матрицу, решаем систему методом прогонки и вычисляем значения сплайна в текущей точке.

Вычисление интеграла:

Численное интегрирование по формуле Симпсона; отрезок [a, b] разбивается на N = 2n частей:

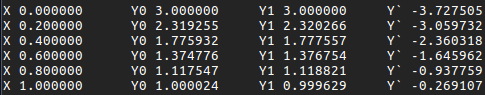


Результат работы программы

Решение краевой задачи:

Рисунок

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| i | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 1.0 |
|  | 3.00000 | 2.319255 | 1.775932 | 1.374776 | 1.117547 | 1.000024 |
|  | -3.727505 | -3.059732 | -2.360318 | -1.645962 | -0.937759 | -0.269107 |



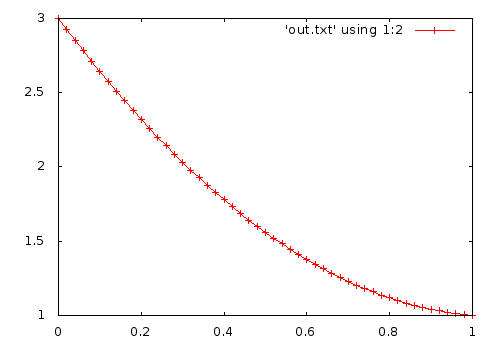
Интерполяция:

Рисунок 2

Численное интегрирование:

Заключение

В рамках курсовой работы была решена краевая задача, результаты которой удовлетворяют заданным граничным условиям в концах интервала. Проведена интерполяция кубическими сплайнами, построен график сеточной функции, который иллюстрирует решение дифференциального уравнения. По формуле Симпсона вычислено приближенное значение интеграла для заданной подынтегральной функции.

Литература

1. Шарый С. П. Курс вычислительных методов.─ Новосибирск. НГУ, 2011.
2. Краснов М.Л., Киселев А.И., Макаренко Г.И. и др. Вся высшая математика. - т. 6 М.: Едиториал УРСС, 2003. — 256 с.
3. Wikipedia –[электронный ресурс]: <https://ru.wikipedia.org>
4. Волков Е. А. Глава 1. Приближение функций многочленами. § 11. Сплайны // Численные методы. — Учеб. пособие для вузов. — 2-е изд., испр.. — М.: Наука, 1987. — С. 63-68. — 248 с
5. Wolfram Mathworld. [электронный ресурс]: http://mathworld.wolfram.com/

Листинг

**main.c**

#include "main.h"

float d2f(float x, float y, float d1, float d2)

{

return powf(d2, 3) - 9 \* sin(x) \* d2 - sin(x) + 9 \* exp(x) \* d1 - (y / powf(x + 2, 3));

}

float Method\_half\_division(float a, float b, float x, float y, float d1)

{

double c = 0;

float eps = 1E-2;

while(fabs(b - a) > eps) {

c = (a + b) / 2;

if(d2f(x, y, d1, a) \* d2f(x, y, d1, c) < 0) {

b = c;

} else if (d2f(x, y, d1, c) \* d2f(x, y, d1, b) < 0) {

a = c;

}

}

return (a + b) / 2;

}

float f(float x, float y, float d1)

{

float a = 1, b = 0;

float fa = 0, fb = 0;

do {

fa = d2f(x, y, d1, a--);

fb = d2f(x, y, d1, b++);

} while (fa \* fb > 0);

return Method\_half\_division(a, b, x, y, d1);

}

float RungeKutt2\_time(float x0, float x1, float h, float y, float d1)

{

float x = x0;

float yt = 0, d1t = 0;

for (; x < x1; x += h) {

yt = y + (h / 2) \* d1;

d1t = d1 + (h / 2) \* f(x, y, d1);

y += h \* d1t;

d1 += h \* f(x + h / 2, yt, d1t);

}

out\_d1 = d1;

return y;

}

float ShootingMethod(float x0, float x1, float y0, float y1, float h)

{

float a = 1, b = 0;

float fa = 0, fb = 0;

float eps = 1E-4;

do {

fa = RungeKutt2\_time(x0, x1, h, y0, a) - y1;

fb = RungeKutt2\_time(x0, x1, h, y0, b) - y1;

a -= h;

b += h;

} while (fa \* fb > 0);

float g = 0;

while(fabs(b - a) > eps) {

g = (a + b) / 2;

if((RungeKutt2\_time(x0, x1, h, y0, a) - y1) \*

(RungeKutt2\_time(x0, x1, h, y0, g) - y1) < 0) {

b = g;

} else if ((RungeKutt2\_time(x0, x1, h, y0, g) - y1) \*

(RungeKutt2\_time(x0, x1, h, y0, b) - y1) < 0) {

a = g;

}

}

return (a + b) / 2;

}

float DoubleCounting(float x0, float x1, float y0, float y1, float h)

{

float delta = 1;

float eps = 1E-4;

float d1[2];

int k = 0;

for (float h0 = h; delta >= eps; h0 /= 2, k ^= 1) {

d1[k] = ShootingMethod(x0, x1, y0, y1, h0);

if (h > h0)

delta = fabsf(d1[k] - d1[k ^ 1]);

}

return d1[k];

}

float\* DoubleCountingRunge(float \*X, int n, float h, float x0, float x1, float y0, float d1)

{

int n1 = n;

int exit = 1, k = 0;

float h1 = h;

float \*Y0 = malloc(sizeof(float) \* n1);

float \*Y1 = malloc(sizeof(float) \* n1);

printf("Write Y0\n");

for (int i = 0; i < n1; i++)

Y0[i] = RungeKutt2\_time(x0, X[i], h1, y0, d1);

#if 1

printf("k = %d\n", k);

printf("h1 = %.8f\n", h1);

for (int i = 0; i < n1; i++) {

printf("X[%d] %f\t", i, X[i]);

printf("Y[%d] %f\t", i, Y0[i]);

printf("Y`[%d] %f\n", i, out\_d1);

}

#endif

printf("\n");

do {

k ^= 1;

h1 /= 2;

if (k == 0) {

for (int i = 0; i < n1; i++) {

Y0[i] = RungeKutt2\_time(x0, X[i], h1, y0, d1);

printf("X[%d] %f\t", i, X[i]);

printf("Y[%d] %f\t", i, Y0[i]);

printf("Y`[%d] %f\n", i, out\_d1);

}

printf("\n");

} else {

for (int i = 0; i < n1; i++) {

Y1[i] = RungeKutt2\_time(x0, X[i], h1, y0, d1);

printf("X[%d] %f\t", i, X[i]);

printf("Y[%d] %f\t", i, Y1[i]);

printf("Y`[%d] %f\n", i, out\_d1);

}

printf("\n");

}

exit = 0;

for (int i = 0; i < n1; i++) {

printf("%f\n", fabsf(Y0[i] - Y1[i]));

if (fabsf(Y0[i] - Y1[i]) > eps) {

printf("Break\n");

exit = 1;

if (k == 0) {

for (int i = 0; i < n1; i++)

Y1[i] = 0;

} else {

for (int i = 0; i < n1; i++)

Y0[i] = 0;

}

break;

}

}

printf("\n");

} while (exit);

if (k == 0) {

free(Y1);

return Y0;

} else {

free(Y0);

return Y1;

}

return NULL;

}

float NIntegr(float a, float b, float y0, float d1)

{

float n0 = 100;

float n = n0;

float h;

float S0 = 0;

float delta = 1;

float eps = 1E-2;

for (;delta >= eps; n \*= 2) {

float S1 = 0;

h = (b - a) / n;

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

RungeKutt2\_time(a, a + i \* h, h, y0, d1);

float y1 = out\_d1;

RungeKutt2\_time(a, a + (i + 1) \* h, h, y0, d1);

float y2 = out\_d1;

S1 += (y1 + y2) / 2;

}

S1 \*= h;

delta = fabsf(S1 - S0);

S0 = S1;

}

return S0;

}

int main()

{

float x0 = 0, x1 = 1, y0 = 3, y1 = 3;

float h = 0.2;

int n = 6;

float d1 = DoubleCounting(x0, x1, y0, y1, h);

float \*X = malloc(sizeof(float) \* n);

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

X[i] = h \* i;

float \*Y = DoubleCountingRunge(X, n, h, x0, x1, y0, d1);

if (Y == NULL) {

printf("kek\n");

return 0;

}

FILE \*res = fopen("res.txt", "w");

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

fprintf(res, "%.3f %.3f\n", X[i], Y[i]);

}

fclose(res);

FILE \*out = fopen("out.txt", "w");

for (float i = x0; i <= x1; i += (h / 10)) {

fprintf(out, "%.3f %.3f\n", i, Lagrange(i, X, Y, n));

}

fclose(out);

float I = NIntegr(x0, x1, y0, d1);

printf("I = %.3f\n", I);

free(X);

free(Y);

return 0;

}

**main.h**

#ifndef MAIN\_H

#define MAIN\_H

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include "Lagrange.h"

float out\_d1 = 0;

float eps = 1E-2;

float d2f(float x, float y, float d1, float d2);

float Method\_half\_division(float a, float b, float x, float y, float d1);

float f(float x, float y, float d1);

float RungeKutt2\_time(float x0, float x1, float h, float y, float d1);

float ShootingMethod(float x0, float x1, float y0, float y1, float h);

float DoubleCounting(float x0, float x1, float y0, float y1, float h);

float\* DoubleCountingRunge(float \*X, int n, float h, float x0, float x1, float y0, float d1);

float NIntegr(float a, float b, float y0, float d1);

#endif

**Lagrange.c**

#include "Lagrange.h"

float Lagrange(float z, float \*x, float \*y, int n)

{

float \*Px = calloc(n, sizeof(float));

float fz = 0.0, mult;

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

mult = 1.0;

for (int j = 0; j < n; j++) {

if (i != j)

mult \*= (z - x[j]) / (x[i] - x[j]);

}

Px[i] = mult \* y[i];

}

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

fz += Px[i];

free(Px);

return fz;

}

**Lagrange.h**

#ifndef LAGRANGE\_H

#define LAGRANGE\_H

#include <stdlib.h>

#include <math.h>

float Lagrange(float z, float \*x, float \*y, int n);

#endif