Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Саратовский государственный технический университет

имени Гагарина Ю.А.»

Институт электронной техники и приборостроения

Кафедра Информационная безопасность автоматизированных систем

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Практическая работа № 3

Тема «Решение краевой задачи методом прогонки»

|  |  |
| --- | --- |
|  | Выполнил: студент 3 курса  учебной группы с-ИБС 31  очной формы обучения  Савельев С.А.  Проверил: доцент кафедры ИБС Кожанова Е.Р. |

Саратов 2021

Целью данной работы является формирование практических навыков решения краевой задачи для линейного обыкновенного дифференциального уравнения методом прогонки с их программной реализацией.

**ЗАДАНИЕ НА ПРАКТИЧЕКСУЮ РАБОТУ**

1. Изучить основные понятия по данной теме.

2. Выполнить расчет по заданному варианту № 21.

3. Разработать алгоритм решения задачи.

4. Написать программу для реализации алгоритма.

5. Выполнить контрольный тест работы программы и сравнить с результатами, полученными в п.2.

6. Выводы.

**1. ОСНОВНЫЕ ПОНЯТИЯ**

*Краевая задача (граничная задача)* – задача о нахождении решения заданного дифференциального уравнения (системы дифференциальных уравнений), удовлетворяющего [краевым (граничным) условиям](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D0%BB%D0%BE%D0%B2%D0%B8%D1%8F) в концах интервала или на границе области.

Одним из методов приближенного решения краевых задач для линейного обыкновенного дифференциального уравнения второго порядка является *метод прогонки.*

Постановка задачи. Требуется найти функцию , которая является решением следующей краевой задачи:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

Задачу (1), (2) называют краевой, поскольку дополнительные условия (2) задаются на концах отрезка

Разностная аппроксимация производных. Введем на отрезке равномерную сетку Записывая уравнение (1) во внутренних узлах сетки получим -но уравнение для определения неизвестных

|  |  |
| --- | --- |
|  | (3) |

где

Для того чтобы получить замкнутую систему уравнений относительно , необходимо первые и вторые проивзодные функции в узловых точках выразить через значения в этих точках. Будем предполагать, что функция имеет все необходимые по ходу рассуждения непрерывные производные. Выразим значения по формуле Тейлора, беря точку в качестве точки разложения:

|  |  |
| --- | --- |
|  | (4) |

Отсюда можно получить следующие выражения для точного значения первой производной функции в точке

|  |  |
| --- | --- |
|  | (5) |

а также выражение для точного значения второй производной функции в той же точке

|  |  |
| --- | --- |
|  | (6) |

Отношения

в формуле (5) называются правой разностной производной, левой разностной проиводной и центральной разностной производной, соответственно. Отношение

в (6) называется второй разностной производной.

Из формулы (5) следует, что левая и правая разностные производные аппроксимируют производную с первым порядком точности относительно шага а центральная разностная производная – со вторым порядком точности относительно Из (6) следует, что вторая разностная производная аппроксимирует производную со вторым порядком точности.

Решение задачи методом прогонки. Пусть – приближенное значение, соответствующее точному значению функции в точке Заменим второй разностной производной и первой центральной разностной производной:

|  |  |
| --- | --- |
|  | (7) |

соответственно, подставляя в них вместо величины . В результате вместо дифференциальной задачи (1), (2) получим следующую разностную задачу:

|  |  |
| --- | --- |
|  | (8) |
|  | (9) |

Подставляя краевые условия (9) в (8), получим относительно значений систему линейных алгебраических уравнений *(n-1)*-го порядка с трёхдиагональной матрицей.

|  |  |
| --- | --- |
|  | (10) |

Система (10) решается методом прогонки. Запишем систему в виде

где

Введем дополнительные переменные

Чтобы сделать схему вычислений однородной, положим Тогда Кроме того,

Тогда решение системы определяется формулой

**2. Расчет по заданному варианту №21**

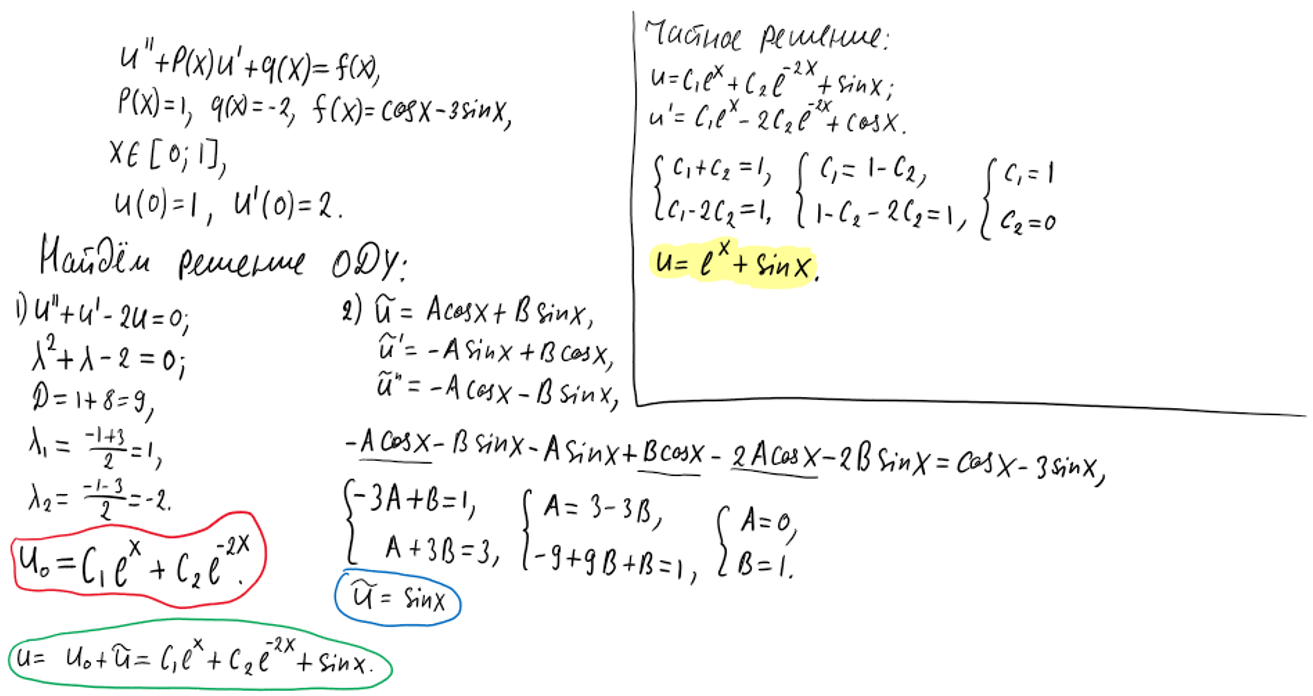
Задание:

Решить краевую задачу методом прогонки. Для задания краевого условия в точке предварительно необходимо решить аналитически соответствующую задачу Коши:

Значение находится постановкой точки в точное решение задачи Коши:

Здесь

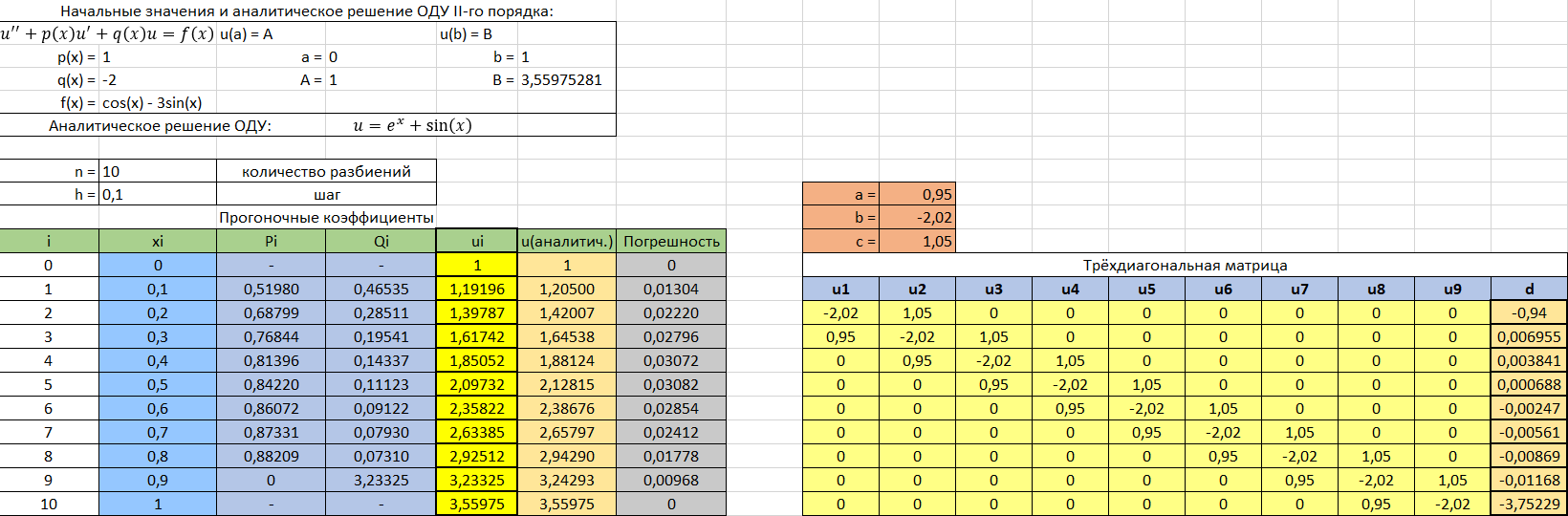
Аналитическое решение задачи Коши для ОДУ II-го порядка:



Итак, аналитическим решение ОДУ II-го порядка является выражение

Путём подстановки точки в точное решение задачи Коши находится значение

Вычисления производились в программе Microsoft Excel. Для задания была составлена таблица, где было указано количество разбиений отрезка соответственно был рассчитан шаг разбиения Были обозначены коэффициенты системы уравнений и составлена трёхдиагональная матрица. При помощи составленной матрицы вычисляются прогоночные коэффициенты и находятся численные значения функции в точках:



Также точные значения сравниваются со значениями, вычисленными методом прогонки, и показываются погрешности данного метода.

**3. Алгоритм решения задачи**

Для решения краевой задачи методом прогонки для линейного ОДУ II-го порядка

составляется трёхдиагональная матрица:

где

коэффициенты матрицы,

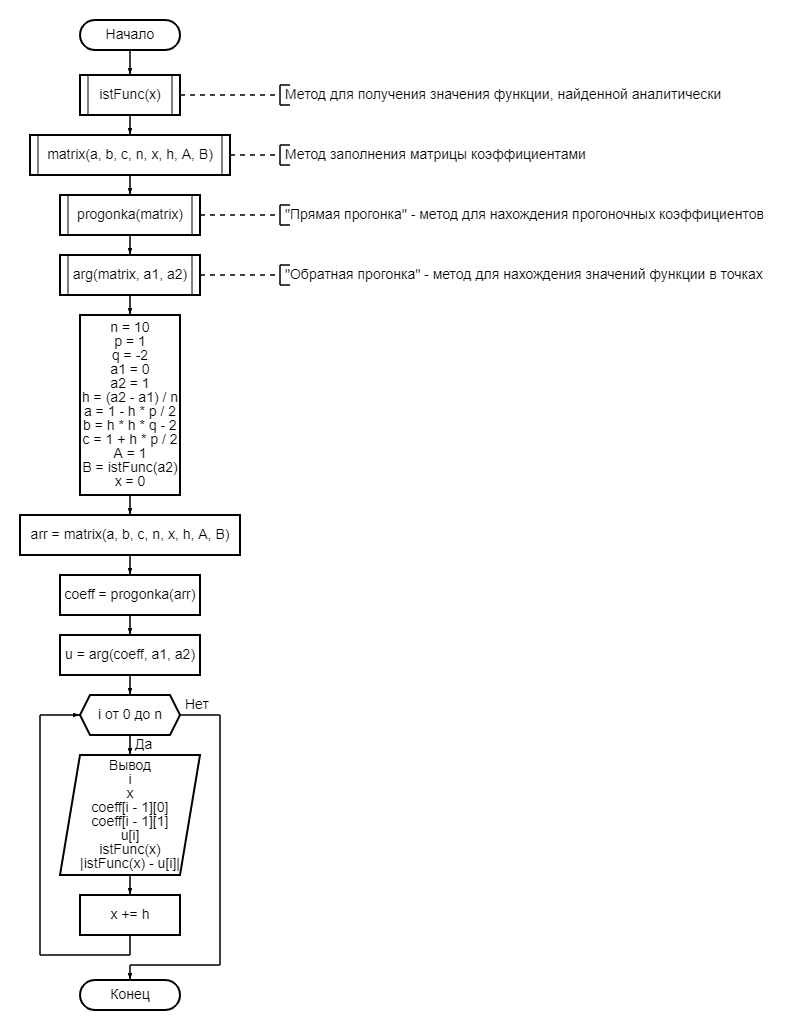
В условиях поставленной задачи, эти коэффициенты не зависят от аргумента поэтому коэффициенты матрицы можно переписать следующим образом:

Нужно обратить внимание на то, что эти коэффициенты выводятся при помощи формул разностных производных второго порядка точности.

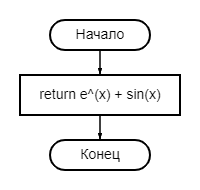
Дальше находятся прогоночные коэффициенты (прямая прогонка):

После чего следует метод обратной прогонки

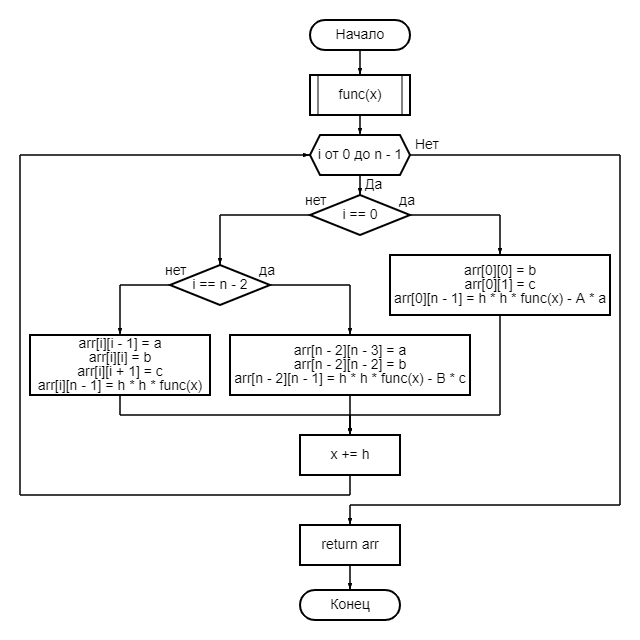
Для основной программы была составлена блок-схема:

****

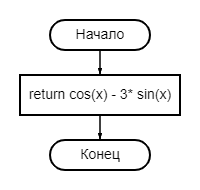
Для программы был составлен метод istFunc(x):

****

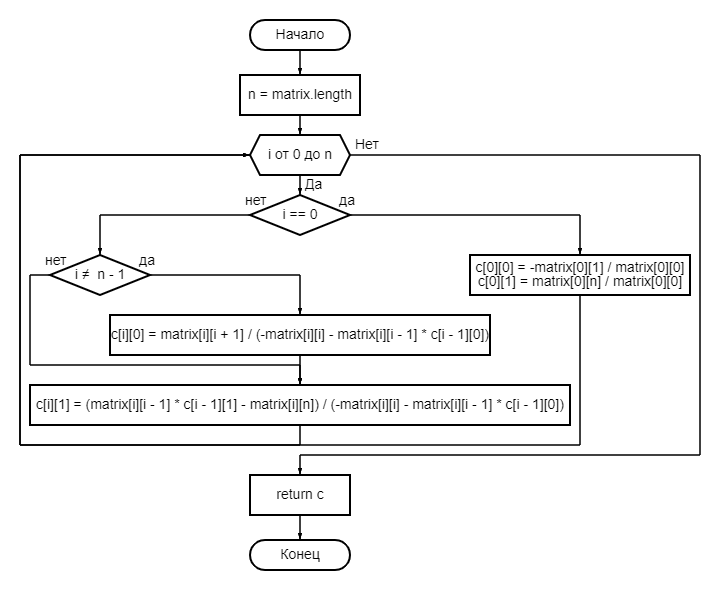
Для программы был составлен метод matrix(a, b, c, n, x, h, A, B):

****

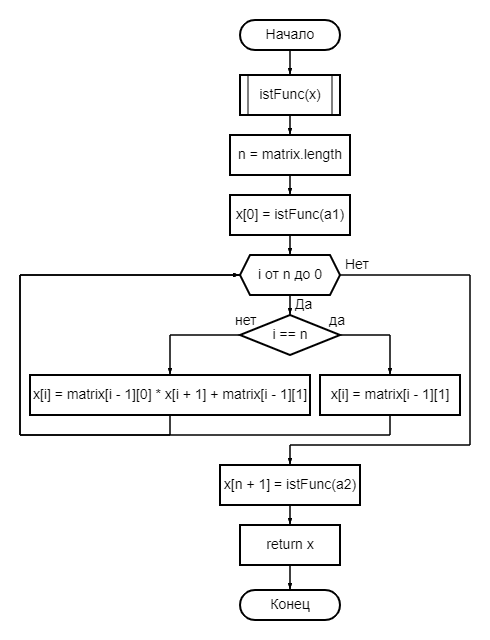
Для программы был составлен метод func(x):

****

Для программы был составлен метод progonka(matrix):

****

Для программы был составлен метод arg(matrix, a1, a2):

****

**4. Программа для реализации алгоритма**

Данная программа была реализована на языке Java:

public class Main {

public static void main(String[] args){

int n = 10;

double p = 1, q = -2, a1 = 0, a2 = 1, h = (a2 - a1) / n;

double a = 1 - h \* p / 2, b = h \* h \* q - 2, c = 1 + h \* p / 2;

double A = 1, B = istFunc(a2);

double[][] coeff;

double[] u;

double x = 0;

double[][] arr;

arr = matrix(a, b, c, n, x, h, A, B);

coeff = progonka(arr);

u = arg(coeff, a1, a2);

System.out.println("Программа находит численные значения ОДУ II-го порядка методом прогонки.");

System.out.println("\ni\t\t|x\t\t\t|P\t\t\t\t|Q\t\t\t\t|u\t\t\t\t|u(аналитич.)\t|Погрешность");

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

System.out.println(i + "\t\t|" + String.format("%.1f", x)

+ "\t\t|" + ((i == 0) ? "\t-\t" : (i == n) ? "\t-\t" : String.format("%.5f", coeff[i - 1][0]))

+ "\t\t|" + ((i == 0) ? "\t-\t" : (i == n) ? "\t-\t" : String.format("%.5f", coeff[i - 1][1]))

+ "\t\t|" + String.format("%.5f", u[i])

+ "\t\t|" + String.format("%.5f", istFunc(x))

+ "\t\t|" + String.format("%.5f", Math.abs(istFunc(x) - u[i])));

x += h;

}

}

public static double[][] progonka(double[][] matrix){ // Метод для нахождения прогоночных коэффициентов

int n = matrix.length;

double[][] c = new double[n][2];

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

if (i == 0){

c[0][0] = -matrix[0][1] / matrix[0][0];

c[0][1] = matrix[0][n] / matrix[0][0];

}else{

if(i != n - 1)

c[i][0] = matrix[i][i + 1] / (-matrix[i][i] - matrix[i][i - 1] \* c[i - 1][0]);

c[i][1] = (matrix[i][i - 1] \* c[i - 1][1] - matrix[i][n]) / (-matrix[i][i] - matrix[i][i - 1] \* c[i - 1][0]);

}

}

return c;

}

public static double[][] matrix(double a, double b, double c, int n, double x, double h, double A, double B){ // Метод для заполнения матрицы коэффициентами

double[][] arr = new double[n - 1][n];

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

if (i == 0) {

arr[0][0] = b;

arr[0][1] = c;

arr[0][n - 1] = h \* h \* func(x) - A \* a;

} else if (i == n - 2) {

arr[n - 2][n - 3] = a;

arr[n - 2][n - 2] = b;

arr[n - 2][n - 1] = h \* h \* func(x) - B \* c;

} else {

arr[i][i - 1] = a;

arr[i][i] = b;

arr[i][i + 1] = c;

arr[i][n - 1] = h \* h \* func(x);

}

x += h;

}

return arr;

}

public static double[] arg(double[][] matrix, double a1, double a2){ // Метод для нахождения значений функции в точках

int n = matrix.length;

double x[] = new double[n + 2];

x[0] = istFunc(a1);

for (int i = n; i > 0; i--){

if(i == n)

x[i] = matrix[i - 1][1];

else

x[i] = matrix[i - 1][0] \* x[i + 1] + matrix[i - 1][1];

}

x[n + 1] = istFunc(a2);

return x;

}

public static double func(double x){ // Метод для обозначения правой части исходного ОДУ

return Math.cos(x) - 3 \* Math.sin(x);

}

public static double istFunc(double x){ // Метод для нахождения аналитически найденного значения функции в точке

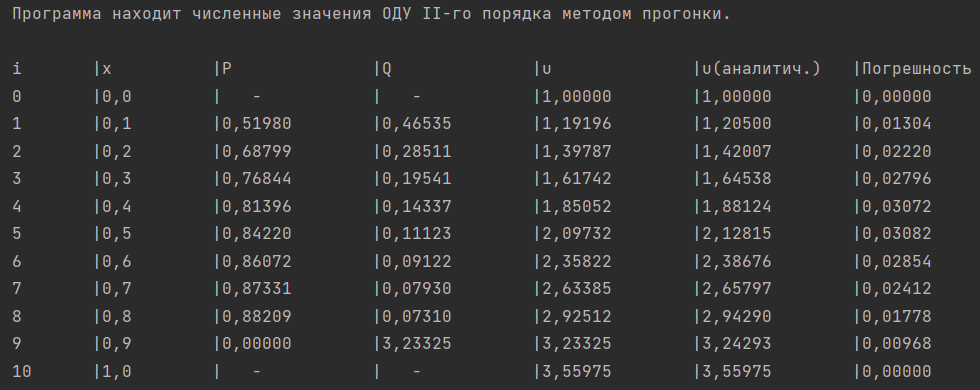
return Math.exp(x) + Math.sin(x);

}

}

**5. Контрольный тест**

Результаты работы программы:



Если сравнить полученные программой данные, с данными, которые были получены при расчёте в Microsoft Excel (п. 2), то можно увидеть, что значения совпадают друг с другом.

**6. ВЫВОДЫ**

В рамках данной работы были применены основные навыки решения краевой задачи для линейного обыкновенного дифференциального уравнения II-го порядка методом прогонки, а также была написана программа, реализующая решение задания на языке Java. Результаты программы и ручного расчета совпали.