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-ИБС32  очной формы обучения  Солодилов В.В.  Проверил: доцент кафедры ИБС Кожанова Е.Р. |

Саратов 2022

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

**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

**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:

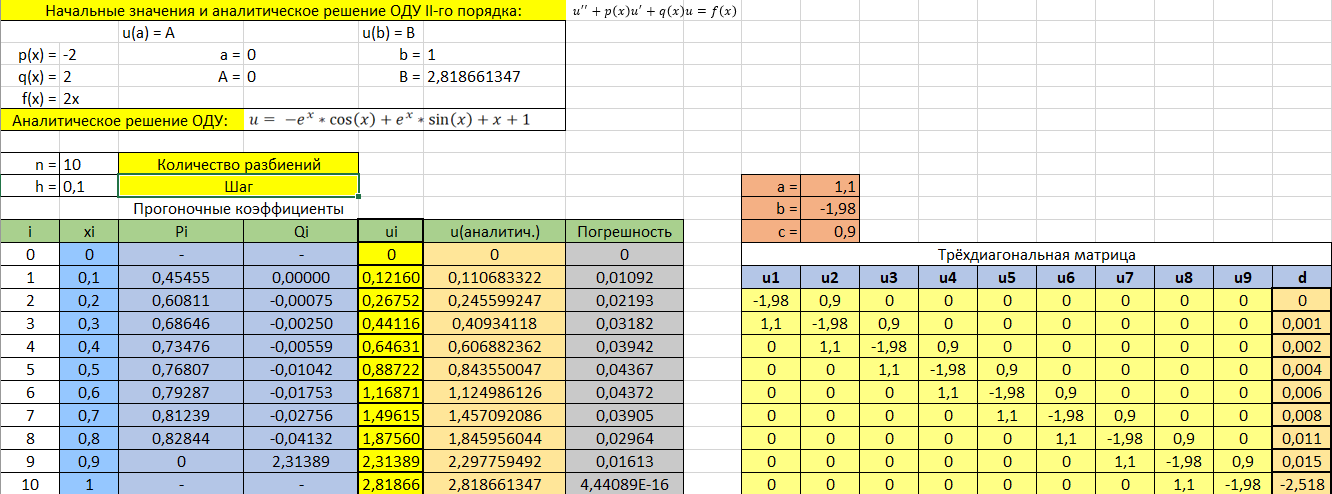
Здесь

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.

**Алгоритм решения задачи**

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

составляется трёхдиагональная матрица:

где

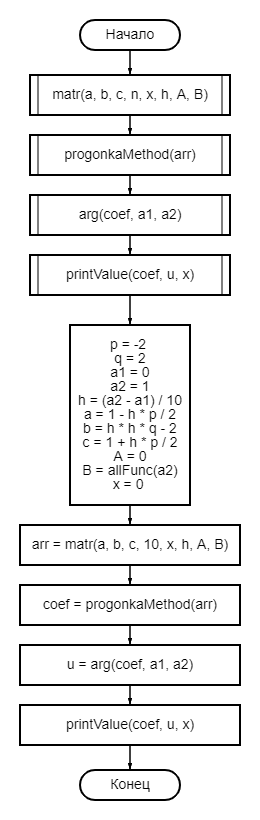
коэффициенты матрицы,

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

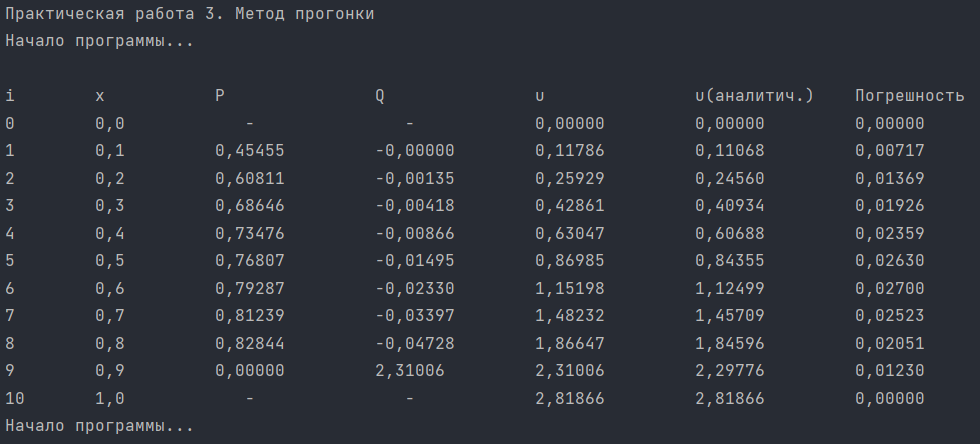


**Листинг программы**

*public class* Main {  
 *public static void* printValue(*double*[][] coef, *double*[] u, *double* x) {  
 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 <= 10; i++){  
 System.out.println(i + "\t\t " + String.format("%.1f", x)  
 + "\t\t " + ((i == 0) ? "\t-\t" : (i == 10) ? "\t-\t" : String.format("%.5f", coef[i - 1][0]))  
 + "\t\t " + ((i == 0) ? "\t-\t" : (i == 10) ? "\t-\t" : String.format("%.5f", coef[i - 1][1]))  
 + "\t\t " + String.format("%.5f", u[i])  
 + "\t\t " + String.format("%.5f", allFunc(x))  
 + "\t\t " + String.format("%.5f", Math.abs(allFunc(x) - u[i])));  
 x += 0.1;  
 }  
 }  
  
 *public static double*[][] progonkaMethod(*double*[][] matr) {  
 *int* length = matr.length;  
 *double*[][] tempArr = *new double*[length][2];  
  
 *for* (*int* i = 0; i < length; i++) {  
 *if* (i == 0) {  
 tempArr[0][0] = -matr[0][1] / matr[0][0];  
 tempArr[0][1] = matr[0][length] / matr[0][0];  
 } *else* {  
 *if*(i != length - 1)  
 tempArr[i][0] = matr[i][i + 1] / (-matr[i][i] - matr[i][i - 1] \* tempArr[i - 1][0]);  
 tempArr[i][1] = (matr[i][i - 1] \* tempArr[i - 1][1] - matr[i][length]) / (-matr[i][i] - matr[i][i - 1] \* tempArr[i - 1][0]);  
 }  
 }  
 *return* tempArr;  
 }  
  
 *public static double*[][] matr(*double* a, *double* b, *double* c, *int* n, *double* x, *double* h, *double* A, *double* B){ *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 \* rightFunc(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 \* rightFunc(x) - B \* c;  
 } *else* {  
 arr[i][i - 1] = a;  
 arr[i][i] = b;  
 arr[i][i + 1] = c;  
 arr[i][n - 1] = h \* h \* rightFunc(x);  
 }  
 x += h;  
 }  
  
 *return* arr;  
 }  
  
 *public static double*[] arg(*double*[][] matr, *double* a1, *double* a2){ *int* n = matr.length;  
 *double*[] x = *new double*[n + 2];  
  
 x[0] = allFunc(a1);  
 *for* (*int* i = n; i > 0; i--){  
 *if*(i == n)  
 x[i] = matr[i - 1][1];  
 *else* x[i] = matr[i - 1][0] \* x[i + 1] + matr[i - 1][1];  
 }  
 x[n + 1] = allFunc(a2);  
  
 *return* x;  
 }  
  
 *public static double* allFunc(*double* x) {  
 *return* -Math.exp(x) \* Math.cos(x) + Math.exp(x) \* Math.sin(x) + x + 1;  
 }  
  
 *public static double* rightFunc(*double* x) {  
 *return* 2 \* x;  
 }  
  
 *public static void* main(String[] args) {  
 System.out.println("Практическая работа 3. Метод прогонки");  
 System.out.println("Начало программы...");  
  
 *double* p = -2, q = 2;  
 *double* a1 = 0, a2 = 1, h = (a2 - a1) / 10;  
 *double* a = 1 - h \* p / 2, b = h \* h \* q - 2, c = 1 + h \* p / 2;  
 *double* A = 0, B = allFunc(a2);  
 *double* x = 0;  
 *double*[] u;  
 *double*[][] arr, coef;  
  
 arr = matr(a, b, c, 10, x, h, A, B);  
 coef = progonkaMethod(arr);  
 u = arg(coef, a1, a2);  
  
 printValue(coef, u, x);  
  
 System.out.println("Конец программы...");  
 }  
}

**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.

**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.