|  |
| --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  Учреждение образования «Полоцкий государственный университет»    Факультет информационных технологий  Кафедра технологий программирования  ОТЧЕТ по  ЛАБОРАТОРНОЙ РАБОТЕ №5  По дисциплине  **«Методы численного анализа»**  На тему: «Метод коллокаций и Галёркина для решения краевой задачи ОДУ»  ВЫПОЛНИЛ студент группы 15-КБ  Осипенко В.В.  ПРОВЕРИЛА ассистент кафедры ТП  Рудькова Т.С.  Полоцк, 2017 г. |

**Название**: «Метод коллокаций и Галѐркина для решения краевой задачи обыкновенных дифференциальных уравнений».

**Цель работы**: Изучить метод коллокаций и Галѐркина для нахождения решения краевой задачи обыкновенных дифференциальных уравнений.

**Теоретическая часть**:

**Метод коллокаций**.

Пусть имеем задачу, в основе которой лежит линейное дифференциальное уравнение второго порядка следующего вида:

Где p(x), q(x), f(x)- известные функции и задача имеет нулевые граничные условия: u(0)=0 u(1)=0

Будем искать приближенное решение этой задачи (1)-(2) в виде:

Функция , т.к. имеем нулевые граничные условия.

Базисные функции i=1,2,...,n, дважды дифференцированы и должны удовлетворять нулевым граничным условиям задачи, т.е.

На отрезке [0,1] построим сетку, т.е. выберем n точек ={0 = < <...< =1} , где – узлы сетки , k=1,…,n и называются точками коллокации.

Потребуем, чтобы приближенное решение точно удовлетворяло дифференциальному уравнению в этих n точках, т.е. требуем выполнения следующего равенства:

Так как невязка для исходной задачи определяется по:

соотношение означает, что в точках коллокации невязка уравнения равна нулю и, следовательно, эта невязка будет ортогональна любому набору функций.

Решив систему , найдем значения параметров и, следовательно, конкретное выражение приближенного решения исходной задачи. Таким образом, решение линейной двухточечной краевой задачи методом коллокаций состоит в следующем:

1. На отрезке поиска решения нужно выбрать точки коллокаций i=1,2,…,n

т.е. построить сетку

2. Нужно подобрать функцию так, чтобы выполнялись граничные условия и выбрать систему базисных функций i 1,2,...,n, = удовлетворяющих нулевым граничным условиям;

3. Выразить приближенное решение;

4. построить систему, найти коэффициенты и вектор F;

5. решить систему, т.е. определить значения компонент вектора C;

6. записать решение в явном виде;

7. найти решение задачи в любых точках отрезка, в том числе и в точках коллокаций.

**Метод Галеркина.**

Метод Галеркина основывается на том, что для отыскания коэффициентов в линейном разложении приближенного решения краевой задачи требуют выполнения ортогональности невязки полученной системы алгебраических уравнений к системе базисных функций.

Рассмотрим применение метода Галеркина к двухточечной граничной задаче

Граничные условия выражаются следующим образом:

Параметры α ,β, γ конкретные числовые значения, причем выполняется условие

Будем искать приближенное решение этой задачи в виде

, отвечает за выполнение граничных условий.

i=1,2,…,n – базисные функции, удовлетворяющие нулевым граничным условиям.

Потребуем ортогональности невязки и системы базисных функций k=1,2,…,n, т.е. для k=1,2,…,n выполнения:

Подставим выражение приближенного решения в виде в равенство для k=1,2,…,n, поменяем местами суммирование и интегрирование, получим систему уравнений

Вычислив все коэффициенты, решив систему относительно коэффициентов i c i=1,2,…,n, найдем приближенное решение исходной задачи.

**Вариант №9**

Найти решения граничных задач на отрезке [a,b] с шагом h=0.1:

1. используя метод коллокаций;

2. используя метод Галѐркина.

Оценить погрешность полученного решения.

**Ход работы:**

Составим график, на котором изображена функция значения которой будем брать за точные (Рисунок 1).

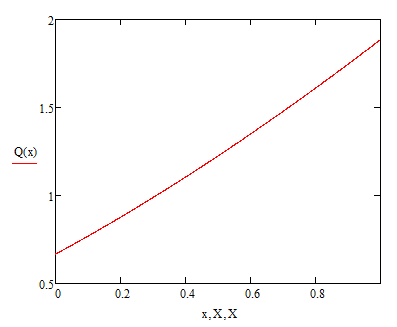


Рисунок 1 – График точного решения.

Напишем программу для метода колокаций и оценим её точность.

Базисные функции для метода:

Листинг 1 – Программа метода колокаций написанная на JAVA.

**public class** main1 {  
  
 **public static double** F(**double** x){  
 **return** (2.0/3.0)\*Math.*sqrt*(Math.*pow*(x+1,3));  
 }  
  
 **public static double** QWER(**double** x, **double** c1, **double** c2){  
 *// (1 + c1\*fn1(X(i)) + c2\*fn2(X(i)))* **return** 100\*Math.*abs*(*F*(x)-(*fn0*(x) + c1\**fn1*(x) + c2\**fn2*(x)))/((*fn0*(x) + c1\**fn1*(x) + c2\**fn2*(x)));  
 }  
  
 **public static double** X(**double** i){  
 **return** 0.1\*i;  
 }  
  
 **public static double** fn1(**double** x){  
 **return** (Math.*pow*(x,2)/2)-x-1.0/3.0;  
 }  
  
 **public static double** fs1(**double** x){  
 **return** x-1;  
 }  
  
 **public static double** f21(**double** x){  
 **return** 1;  
 }  
  
 **public static double** fn2(**double** x){  
 **return** (((Math.*pow*(x,2)\*Math.*sqrt*(2))/2)-x\*Math.*sqrt*(2)-Math.*sqrt*(2)/3);  
 }  
  
 **public static double** fs2(**double** x){  
 **return** x\*Math.*sqrt*(2)-Math.*sqrt*(2);  
 }  
  
 **public static double** f22(**double** x){  
 **return** Math.*sqrt*(2);  
  
 }  
  
 **public static double** fn0(**double** x){  
 **return** (x\*Math.*sqrt*(2)+((1+Math.*sqrt*(2))/3.0));  
 }  
  
 **public static double** fs0(**double** x){  
 **return** Math.*sqrt*(2);  
 }  
  
 **public static void** main(String[] args){  
 **int**[] k = {1, 2};  
 **int** a = 0, b = 1;  
 System.***out***.println(**"Проверка функций: "**);  
 System.***out***.println(**"f1"**);  
 System.***out***.println((3\**fn1*(0)-*fs1*(0)));  
 System.***out***.println(*fs1*(1));  
 System.***out***.println(**"f2"**);  
 System.***out***.println(3\**fn2*(0)-*fs2*(0));  
 System.***out***.println(*fs2*(1));  
 System.***out***.println(**"f0"**);  
 System.***out***.println(3\**fn0*(0)-*fs0*(0));  
 System.***out***.println(*fs0*(1));  
 System.***out***.println(**"x1 = "** + *X*(k[0]));  
 System.***out***.println(**"x2 = "** + *X*(k[1]));  
 System.***out***.println(3.0\**fn1*(0)-*fs1*(0));  
 System.***out***.println(3.0\**fn2*(0)-*fs2*(0));  
 System.***out***.println(*fs1*(1));  
 System.***out***.println(*fs2*(1));  
 **double**[][] A = **new double**[2][2];  
 **for**(**int** i = 0; i < 2; i++){  
 A[i][0] = *f21*(*X*(k[i])) + *fn1*(*X*(k[i]));  
 A[i][1] = *f22*(*X*(k[i])) + *fn2*(*X*(k[i]));  
 }  
 System.***out***.println(A[0][0] + **" \* c1 + "** + A[0][1] + **" \* c2 = "** + *X*(k[0]));  
 System.***out***.println(A[1][0] + **" \* c1 + "** + A[1][1] + **" \* c2 = "** + *X*(k[1]));  
 **double** c1 = 376.673;  
 **double** c2 = -266.24;  
 **double** f0=0.173;  
 *//f0=0;* System.***out***.println(**"Коефициенты:\nc1 = "** + c1 + **"\nc2 = "** + c2);  
 **for**(**int** i = a; *X*(i)<=b;i++){  
 *//System.out.println(c1\*fn1(X(i)));  
 //System.out.println(c2\*fn2(X(i)));* System.***out***.println(**"f["** + (**float**)*X*(i) + **"] = "** + (*fn0*(*X*(i)) + c1\**fn1*(*X*(i)) + c2\**fn2*(*X*(i))));  
 System.***out***.println(**"Погрешность: "** + *QWER*(*X*(i),c1,c2));  
 }  
 }

Проверка функций происходит в программе:

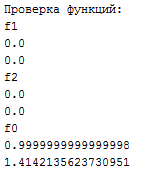


Рисунок 2 – Проверка функций.

Результатом работы программы будет:

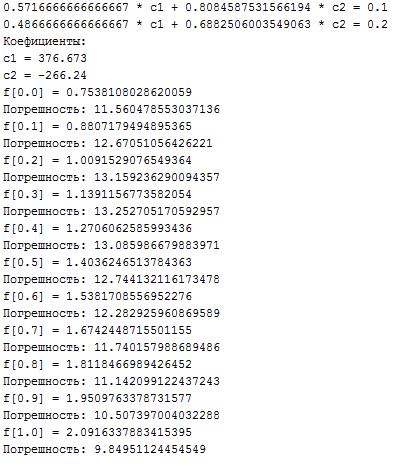


Рисунок 3 – Результат работы программы.

Коэффициенты c1 и c2 находятся вручную из системы уравнений которая выводится в программе, воспользуемся средствами MathCad и найдем их (Рисунок 3).

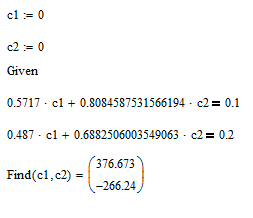


Рисунок 4 – Нахождение коэффициентов c.

Как мы видим погрешность высока (13%), но не превышает максимальную допустимую.

Напишем программу для второго метода – метода Галеркина.

Базисными функциями для программы будут:

Проверим что функции подходят с помощью MathCad

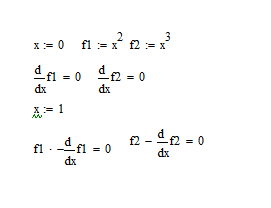


Рисунок 5 – проверка базисных функций

Листинг 2 – Программа - метод Галеркина написанная на JAVA.

**public class** main2 {  
 **public static double** F(**double** x){  
 **return** (2.0/3.0)\*Math.*sqrt*(Math.*pow*(x+1,3));  
 }  
  
 **public static double** QWER(**double** x, **double** c1, **double** c2){  
 *// (1 + c1\*fn1(X(i)) + c2\*fn2(X(i)))* **return** 100\*Math.*abs*(*F*(x)-(*f0*(x) + c1\**f1*(x) + c2\**f2*(x)))/((*f0*(0) + c1\**f1*(x) + c2\**f2*(x)));  
 }  
  
 **public static double** X(**double** i){  
 **return** i\*0.1;  
 }  
  
 **public static double** f0(**double** x){  
 **return** (Math.*sqrt*(2)/3)+Math.*sqrt*(2)\*x;  
 }  
  
 **public static double** f1(**double** x){  
 **return** Math.*pow*(x,1);  
 }  
  
 **public static double** f2(**double** x){  
 **return** Math.*pow*(x,2);  
 }  
  
 **public static void** main(String[] args){  
 **int** a = 0, b = 1;  
 **double**[][] A = **new double**[2][2];  
 **double** c1 = 0.083;  
 **double** c2 = 0.0046;  
 System.***out***.println(**"Коефициенты:\nc1 = "** + c1 + **"\nc2 = "** + c2);  
 **for**(**int** i = a; *X*(i)<=b;i++){  
 System.***out***.println(**"f["** + (**float**)*X*(i) + **"] = "** + (*f0*(*X*(i)) + c1\**f1*(*X*(i))+c2\**f2*(*X*(i))));  
 System.***out***.println(**"Погрешность: "** + *QWER*(*X*(i),c1,c2));  
 }  
 }  
  
}

Результатом работы программы будет:

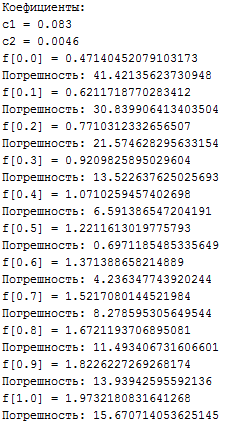


Рисунок 6 – Результат работы программы.

Коэффициенты c находятся вручную из системы.

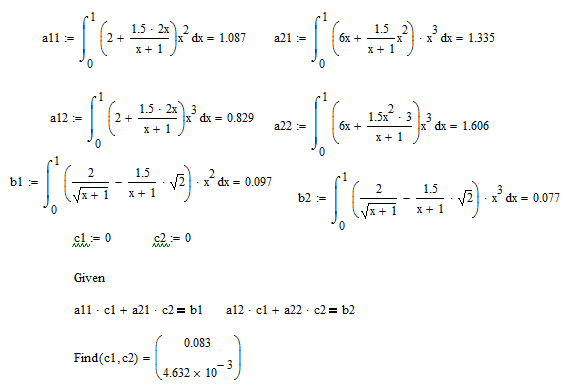


Рисунок 7 – Нахождение коэффициентов.

Как мы видим решение является менее точным, максимальная погрешность составила 41%. Наибольшая погрешность находиться в начале и конце, ближе к середине погрешность приближается к нулю.

Построим график со всеми решениями.

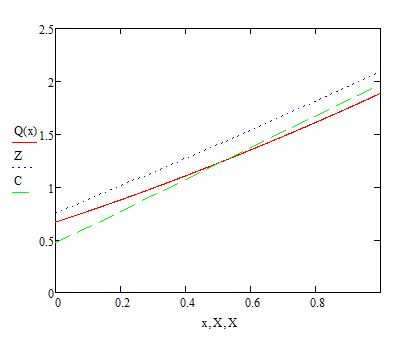


Рисунок 8 – График решений.

На графике Q – точное решение; Z – первый метод; C – второй метод;

Как мы видим оба метода имеют высокую погрешность.

**Выводы о проделанной работе.**

В ходе данной лабораторной работы мы изучили два метода решения краевой задачи ОДУ, оценили их точность. Метод колокации оказался более точным чем метод Галеркина, однако точность сильно варьируется от выбранных базисных функций, и в обоих случаях погрешность была выше чем планировалось.