**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Отчёт о КСР

**Принцип сжимающих отображений**

Вариант 14

**Выполнил:**

Кендысь Алексей Максимович

студент 2 курса, 9 группы,

специальность

“прикладная математика”

**Преподавательница:**

Доцентка кафедры компьютерных

технологий и систем ФПМИ,

Чеб Елена Сергеевна

Минск, 2022

**Содержание:**

1. Принцип сжимающих отображений для решения уравнений --------------- 2-5

1.1 Постановка задачи и разработка алгоритма ---------------------------- 2-4

1.2 Листинг программы --------------------------------------------------------- 4

1.3 Результаты -------------------------------------------------------------------- 5

2. Принцип сжимающих отображений для решения СЛАУ -------------------- 6-9

1.1 Постановка задачи и разработка алгоритма --------------------------- 6-7

1.2 Листинг программы --------------------------------------------------------- 8-9

1.3 Результаты -------------------------------------------------------------------- 9

3. Принцип сжимающих отображений для решения интегральных

уравнений -------------------------------------------------------------------------------- 10-12

1.1 Постановка задачи и разработка алгоритма -------------------------- 10-13

1.2 Листинг программы --------------------------------------------------------- 13

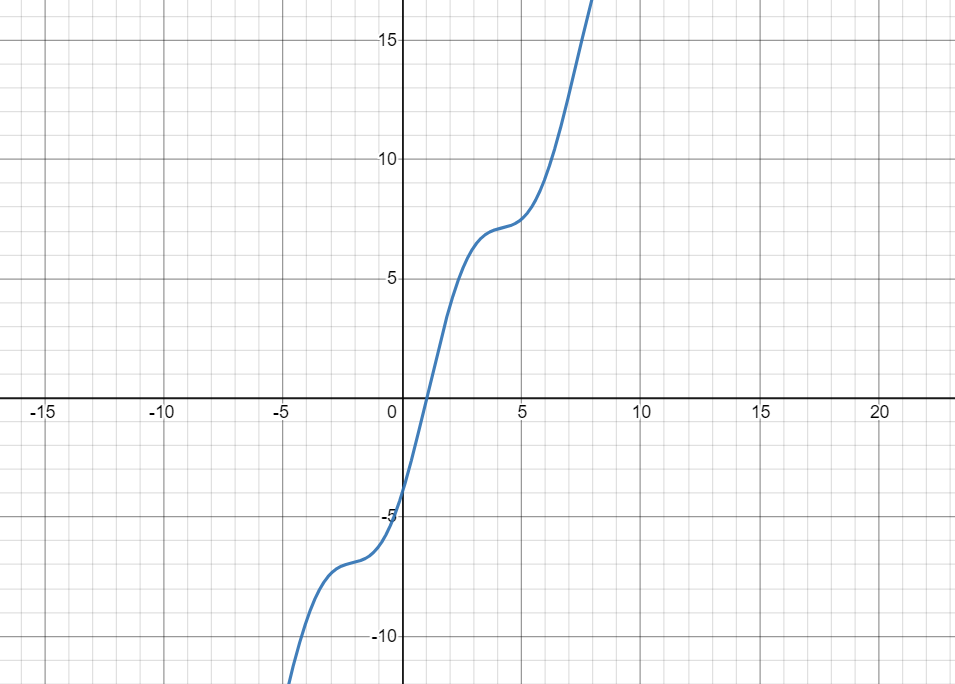
**Принцип сжимающих отображений для решения уравнений**

**Постановка задачи и разработка алгоритма**

Приводя уравнение g(x) = 0 к виду, для которого справедлив принцип сжимающих отображений, найти корни уравнения с точностью . Составить алгоритм и написать программный код, реализующий метод последовательных приближений, предусматривающий: построение графика g(x), вычисление априорной оценки количества итераций, вывод на печать последней итерации и её номер.

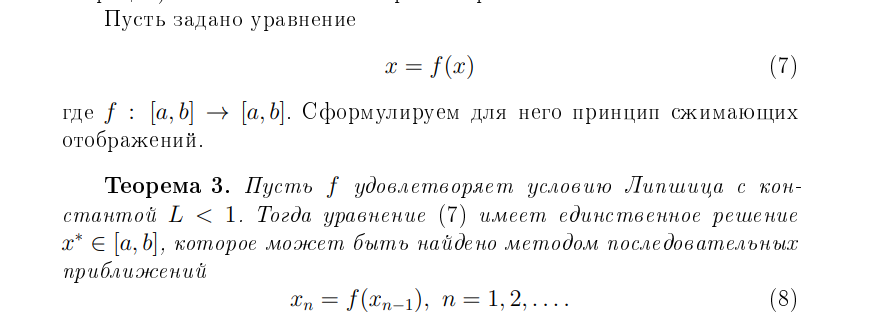
Уравнение:

График функции g(x) (взято с Desmos):



Из графика видно, что уравнение имеет только один корень. Для простоты отделим его, выбрав отрезок .

Для доказательства справедливости принципа сжимающих отображений используем следующую теорему:



Для начала рассмотрим наше уравнение в общем виде .

Ограничим :

Перепишем уравнение в виде .

Параметр выбираем таким образом, чтобы отображение переводило отрезок в себя и при этом являлось сжимающим.

можем взять как точку минимума функции

Тогда Все условия теоремы выполнены.

Начальное приближение выберем . Получаем алгоритм:

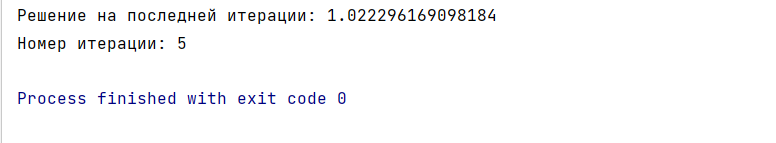
Для оценки достаточного числа итераций используем априорную оценку

Условие окончания итерационного процесса: .

**Листинг программы**

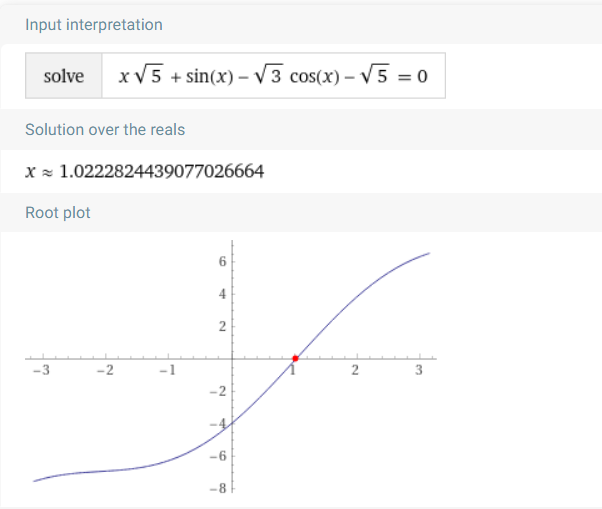
**class** Equation {  
 **private double xk**;  
 **private int k**;  
 **private final static double *E*** = 0.0001;  
 **private final static double *ALPHA*** = 0.1352;  
  
 **public void** algorithm() {  
 **double** x1 = 0.;  
 **double** x2 = f(x1);  
 **k** = 1;  
 **double** c = ***ALPHA*** / (1 - ***ALPHA***);  
 **while**(c \* Math.*abs*(x2 - x1) > ***E***) {  
 x1 = x2;  
 x2 = f(x1);  
 **k**++;  
 }  
 **xk** = x2;  
 }  
  
 **public void** showRes() {  
 System.***out***.print(**"Решение на последней итерации: "**);  
 System.***out***.println(**xk**);  
 System.***out***.print(**"Номер итерации: "**);  
 System.***out***.println(**k**);  
 }  
 **private double** f(**double** x) {  
 **return** x - 0.268 \* g(x);  
 }  
 **private double** g(**double** x) {  
 **return** x \* Math.*sqrt*(5.) + Math.*sin*(x) - Math.*sqrt*(3.) \* Math.*cos*(x) - Math.*sqrt*(5.);  
 }  
}  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 Equation myEquation = **new** Equation();  
 myEquation.algorithm();  
 myEquation.showRes();  
 }  
}

**Результаты**

****

Количество итераций совпало с априорной оценкой, хотя в общем случае априорная оценка будет больше.

Проверим результат вычисления корня (взято с Wolfram Mathematica):



**Принцип сжимающих отображений для решения СЛАУ**

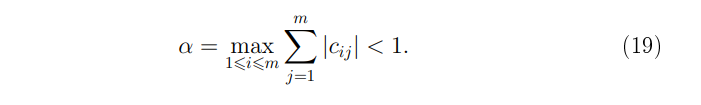
**Постановка задачи и разработка алгоритма**

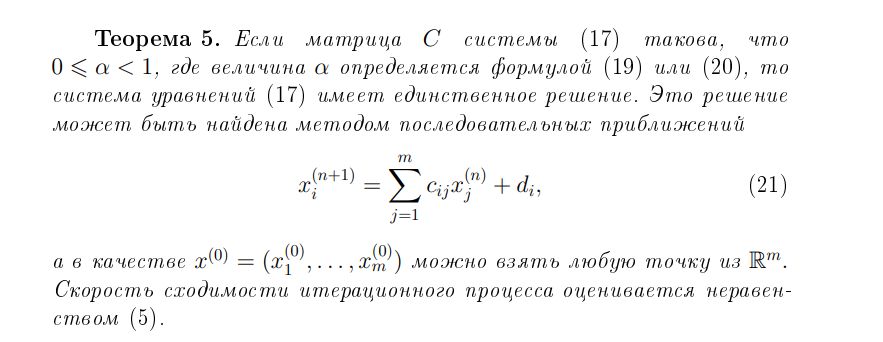
Найти решение системы линейных алгебраических уравнений с точностью . Составить алгоритм и написать программный код, реализующий метод последовательных приближений, предусматривающий: приведение системы к специальному виду для применения метода последовательных приближений, вычисление коэффициента сжатия, вычисление априорной оценки количества итераций, вывод на печать последней итерации и её номер.

Система уравнений:

Имеем систему вида Преобразуем её в систему вида для применения принципа сжимающих отображений.

Для доказательства справедливости принципа сжимающих отображений используем следующую теорему:

**



Для доказательства требуется, чтобы кубическая норма полученной матрицы C была < 1. Из нашей системы будем выражать (из -го уравнения), но нужно следить за нормой получаемой матрицы C. Тогда, фактически, для выполнения этого условия нам требуется исходную матрицу A преобразовать к виду, обладающему диагональным доминированием. Тогда немного поменяем порядок уравнений в системе:

C = ; ;

Т.е. все условиями теоремы выполнены. Выберем начальное приближение

Полученный алгоритм:

n = 0, 1, 2, ...

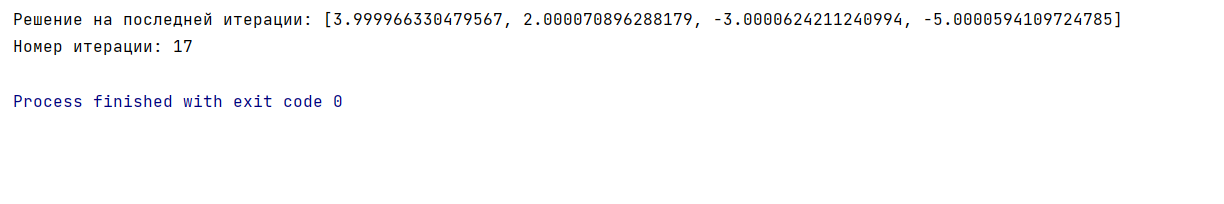
Для оценки достаточного числа итераций используем априорную оценку

Условие окончания итерационного процесса: .

**Листинг программы**

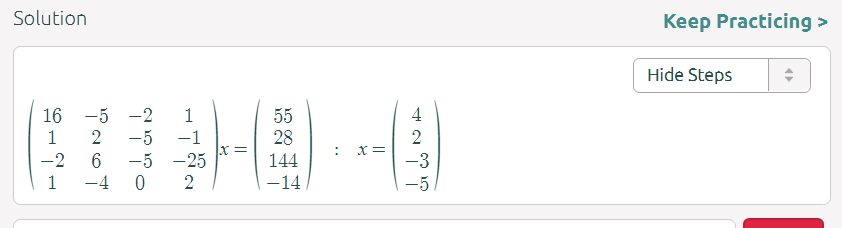
**import** java.util.\*;  
  
**class** Equation {  
 **private final double**[][] **c**;  
 **private final double**[] **d**;  
 **private double**[] **xk**;  
 **private int k**;  
 **private final static double *E*** = 0.001;  
 **private final static double *ALPHA*** = 0.8;  
  
 **public** Equation() {  
 **c** = **new double**[4][4];  
 **for**(**int** i = 0; i < 4; i++) {  
 **c**[i][i] = 0.;  
 }  
 **c**[0][1] = 5. / 16.;  
 **c**[0][2] = 2. / 16.;  
 **c**[0][3] = - 1. / 16.;  
 **c**[1][0] = 1. / 4.;  
 **c**[1][2] = 0.;  
 **c**[1][3] = 2. / 4.;  
 **c**[2][0] = 1. / 5.;  
 **c**[2][1] = 2. / 5.;  
 **c**[2][3] = - 1. / 5.;  
 **c**[3][0] = - 2. / 25.;  
 **c**[3][1] = 6. / 25.;  
 **c**[3][2] = - 5. / 25.;  
  
 **d** = **new double**[4];  
 **d**[0] = 55. / 16.;  
 **d**[1] = 14. / 4.;  
 **d**[2] = - 28. / 5.;  
 **d**[3] = - 144. / 25.;  
 }  
  
 **public void** algorithm() {  
 **double**[] x1 = {0., 0., 0., 0.};  
 **double**[] x2 = f(x1);  
 **k** = 1;  
 **double** c = ***ALPHA*** / (1 - ***ALPHA***);  
 **while**(c \* norm(diff(x1, x2)) > ***E***) {  
 x1 = x2;  
 x2 = f(x1);  
 **k**++;  
 }  
 **xk** = x2;  
 }  
  
 **public void** showRes() {  
 System.***out***.print(**"Решение на последней итерации: "**);  
 System.***out***.println(Arrays.*toString*(**xk**));  
 System.***out***.print(**"Номер итерации: "**);  
 System.***out***.println(**k**);  
 }  
  
 **private double**[] f(**double**[] x) {  
 **double**[] res = Arrays.*copyOf*(**d**, **d**.**length**);  
 **for**(**int** i = 0; i < x.**length**; i++) {  
 **for**(**int** j = 0; j < x.**length**; j++) {  
 res[i] += **c**[i][j] \* x[j];  
 }  
 }  
 **return** res;  
 }  
  
 **private double**[] diff(**double**[] x1, **double**[] x2) {  
 **double**[] res = **new double**[x1.**length**];  
 **for**(**int** i = 0; i < res.**length**; i++) {  
 res[i] = x1[i] - x2[i];  
 }  
 **return** res;  
 }  
 **private double** norm(**double**[] x) {  
 **double** max = 0.;  
 **for** (**double** i : x) {  
 **if** (Math.*abs*(i) > max) {  
 max = Math.*abs*(i);  
 }  
 }  
 **return** max;  
 }  
}  
  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 Equation myEquation = **new** Equation();  
 myEquation.algorithm();  
 myEquation.showRes();  
 }  
}

**Результаты**

****

Действительно, количество реальных итераций меньше, чем с априорной оценкой.

Проверим результат вычисления (взято с Symbolab):



**Принцип сжимающих отображений для решения интегральных уравнений**

**Постановка задачи и разработка алгоритма**

1. Выяснить при каких значениях параметра к интегральному уравнению Фредгольма второго рода применим принцип сжимающих отображений в пространстве и в пространстве При найти приближённое решение уравнения с точностью и сравнить его с точным решением.

Условие:

Перепишем уравнение в виде :

Отображение F задаёт отображение, так как представляет собой сумму двух непрерывных функций.

Докажем, что отображение является сжимающим при определённых :



– коэффициент сжатия и при

к исходному интегральному уравнению в пространстве можно применить принцип сжимающих отображений.

Пусть . Тогда .

Пусть , тогда .

Оценим количество приближений по формуле:

Найдем с помощью Wolfram Mathematica (в листинге):

x2 = 

x3 = 

x4 = 

x5 = 

Теперь найдём точное решение. Пусть . Тогда:

;

Вычислим || - x|| = :

 . Т.е. приближённое решение найдено верно.

Теперь рассмотрим пространство .

Оценим ядро K(t, s):

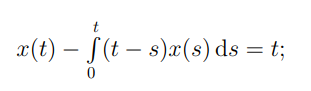


=> F: [0; 1] [0; 1]

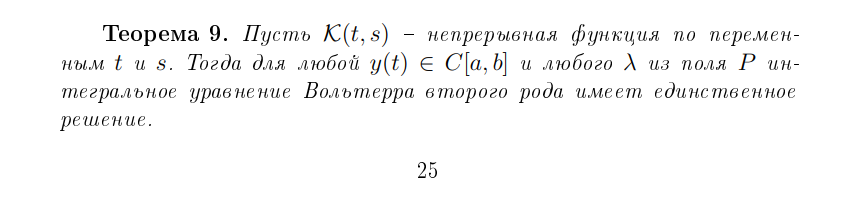
F сжимающее, если .Пусть . Тогда число операций:



=> n = 4

2. Методом последовательных приближений найти решение уравнения Вольтера второго рода в пространстве Уравнение: 

Известно, что интегральное уравнение Вольтерра разрешимо в пространстве непрерывных функций при любой правой части:



Построим последовательные приближения по формуле:



Пусть x0 = 0. Проведём первые 4 итерации на Wolfram Mathematica (в листинге).

Получим:

(t) = t

(t) = t  +

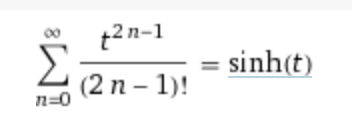
(t) = t + +

(t) = t + + +

Очевидно, что:



А значит, Проверим это:



**Листинг программы**

**1**

lam = 1/4

iters = 1

eps = 10 ^ (-3)

x[t\_] = 0

F[x] := t + 1 + lam \* (t + 1)^(1/4) \* Integrate[x[s] / (1 + s), {s, 0, 1}]

While[ iters <= 5,

x[t\_] = F[x];

Print["итерация:"]

Print[iters];

Print["x(t):"]

Print[x[t]];

iters = iters + 1]

Print["всего итераций:"];

Print[iters - 1];

**2**

iters = 4

x[t\_] = 0

G[x] := t + Integrate[(t -s)x[s], {s, 0, t}]

While[iters > 0,

x[t\_] = G[x];

Print[(x[t])];

iters = iters - 1]