**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра компьютерных технологий и систем**

**КСР**

**«Принцип сжимающих отображений»**

Возовикова Никиты Александровича

студента 2 курса группы 10

специальности «Компьютерная Безопасность»

дневной формы получения

высшего образования

Научный руководитель:

Доцент:

Чеб Елена Сергеевна

Минск, 2021

**ОГЛАВЛЕНИЕ**

[ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ УРАВНЕНИЙ 3](#_Toc71760588)

[Теоритические сведения 3](#_Toc71760589)

[Постановка задачи 3](#_Toc71760590)

[Решение 4](#_Toc71760591)

[Листинг кода(C#) 5](#_Toc71760592)

[Результаты 6](#_Toc71760593)

[ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ СЛАУ 7](#_Toc71760594)

[Теоритические сведения 7](#_Toc71760595)

[Постановка задачи 8](#_Toc71760596)

[Листинг кода(C#) 9](#_Toc71760597)

[Результаты 11](#_Toc71760598)

[ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ 12](#_Toc71760599)

[Теоритические сведения 12](#_Toc71760600)

[Постатовка задачи №1 12](#_Toc71760601)

[Решение задачи №1 13](#_Toc71760602)

[Постановка задачи №2 16](#_Toc71760603)

[Решение задачи №2 16](#_Toc71760604)

# ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ УРАВНЕНИЙ

## Теоритические сведения

Одним из подходов для приближенного решения уравнений можно отнести метод последовательных приближений (последовательных итераций).

Пусть задано уравнение

*(1)*

где

**Теорема.** *Пусть удовлетворяет условию Липшица с константой*

*. Тогда уравнение (1) имеет единственное решение , которое может быть найдено методом последовательных приближений*

Теорема вытекает из принципа сжимающихся отображений, где в качестве множества A выступает отрезок [a,b].

С помощью априорной оценки можно предварительно оценить достаточное число итераций для нахождения приближённого решения с заданной точностью из неравнства:

Откуда:

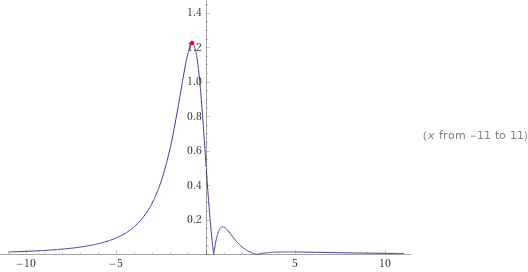
Фактическое число итераций всегда не привышает

## Постановка задачи

Приводя уравнение к виду, для которого справедлив принци сжимающихся отображений, найти корни уравнения с точностью . Составить алгоритм и написать программный код, реализующий метод последовательных приближений, предусамтривающий:

* Построение графика ;
* Вычисление априорной оценки количества итераций;
* Вывод на печать последней итераци и её номер;

## Решение



Следовательно на отрезке [0.5, 2]

## Листинг кода(C#)

**public** **static** **double** g(**double** x)

{

**return** x + Math.Atan(x) + x / (1 + Math.Pow(x, 2)) - 3;

}

**public** **static** **double** f(**double** x)

{

**return** (3 - Math.Atan(x)) \* (1 + Math.Pow(x, 2)) / (2 + Math.Pow(x, 2));

}

**public** **static** **int** apriorCount(**double** x)

{

**return** (**int**)Math.Round(Math.Log(precision \* (1 - alpha) / Math.Abs(x - f(x)), alpha)) + 1;

}

**public** **const** **double** alpha = 0.161;

**public** **const** **double** precision = 0.0001;

**public** **static** **double** leftBound = -10;

**public** **static** **double** rightBound = 10;

**public** Form1()

{

InitializeComponent();

**int** count = 0;

**double** x = 0;

**while** (Math.Abs(x - f(x)) > precision)

{

x = f(x);

count++;

}

richTextBox1.AppendText("Априорная оценка количества итераций:\t" + apriorCount(0) + '\n');

richTextBox1.AppendText($"Последняя итерация №{count}:\t" + x +'\n');

richTextBox1.AppendText($"{f(5)}");

chart1.Series[0].Name = "g(x)";

chart1.Series[0].Color = Color.Red;

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Line;

**for** (**double** i = leftBound; i <= rightBound; i++)

chart1.Series[0].Points.AddXY(i, g(i));

}

## Результаты

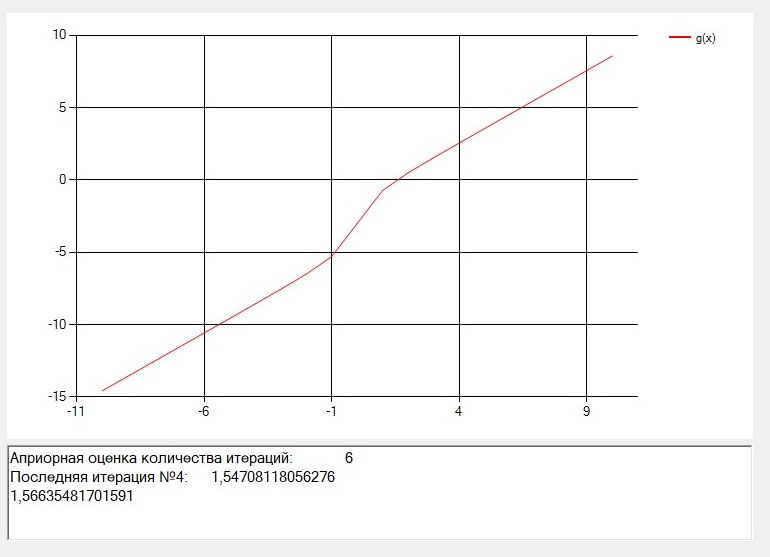


Рис 1.1 График функции g(x) и выходные данные

# ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ СЛАУ

## Теоритические сведения

Пусть дана система линейных алгебраических ураний вида:

которую можно записать в матричном виде

*.* (1)

Предположим, что определитель матрицы A не равен нулю, тогда существует единственное решение системы. Для применения принципа сжимающихся отображений перепишем уравнение (1) в виде

*.*

Обозначим через , тогда отображение задаётся системой линейных уравнений:

Если отображение F – сжатие, то можно применить метод последовательных приближений к решению уравнений .

Условие сжатия имеет вид:

**Теорема.** *Если матрица C системы такова, что , где*

*,*

*то система уравнений (2) имеет единственное решение. Это решение может быть найдено методом последовательных приближений*

*,*

*а в качестве можно взять любую точку из*

## Постановка задачи

Найти решение системы линейных алгебраических уравнений с точностью . Составить алгоритм и написать программный код, реализующий метод последовательных приближений, предусамтривающий:

* Приведение системы к специальному виду для применения метода последовательных приближений;
* Вычисление коэффициента сжатия;
* Вычисление априорной оценки количества итераций;
* Вывод на печать последней итераци и её номер;

## Листинг кода(C#)

**private** **static** **double**[,] matrix = { { -1.1, 0, 0.1 },

{ -0.1, 1.4, 0 },

{ -0.2, 0.4, 0.9 } };

**private** **static** **double**[,] x = { { 0, 0, 0 } };

**private** **static** **double**[,] y = { { 0, 1, -1 } };

**private** **const** **double** precision = 0.001;

**public** **static** **double**[,] transMat(**double**[,] matrix, **double**[,] y)

{ //ПРИВЕДЕНИЕ СИСТЕМЫ К СПЕЦИАЛЬНОМУ ВИДУ

**int** rows = matrix.GetUpperBound(0) + 1;

**int** columns = matrix.Length / rows;

**for** (**int** i = 0; i < rows; i++)

**if** (matrix[i, i] < 0)

matrix[i, i] += 1;

**else**

{

**for** (**int** j = 0; j < columns; j++)

{

matrix[i, j] \*= -1;

y[0, j] \*= -1;

}

matrix[i, i] += 1;

}

**return** matrix;

}

**public** **static** **double**[,] MatrixOperations(**double**[,] one, **double**[,] two, **bool** addition = **true**)

{ // СУММА/РРАЗНОСТЬ МАТРИЦ

**int** sign;

**if** (addition == **true**)

sign = 1;

**else**

sign = -1;

**int** rows = one.GetUpperBound(0) + 1;

**int** columns = one.Length / rows;

**double**[,] newMat = **new** **double**[rows, columns];

**for** (**int** i = 0; i < rows; i++)

**for** (**int** j = 0; j < columns; j++)

newMat[i, j] = one[i, j] + sign \* two[i, j];

**return** newMat;

}

**public** **static** **double**[,] MatVecMultiplication(**double**[,] matrix, **double**[,] Vec)

{ // УМНОЖЕНИЕ МАТРИЦЫ НА ВЕКТОР

**int** rows = matrix.GetUpperBound(0) + 1;

**int** columns = matrix.Length / rows;

**double**[,] newVec = **new** **double**[1, Vec.Length];

**for** (**int** i = 0; i < Vec.Length; i++)

newVec[0, i] = 0;

**for** (**int** i = 0; i < rows; i++)

**for** (**int** j = 0; j < columns; j++)

newVec[0, i] += matrix[i, j] \* Vec[0, j];

**return** newVec;

}

**public** **static** **double** matrixNorm(**double**[,] matrix)

{ //НОРМА МАТР�ЦЫ

**int** rows = matrix.GetUpperBound(0) + 1;

**int** columns = matrix.Length / rows;

**double** max = 0;

**for** (**int** i = 0; i < rows; i++)

{

**double** temp = 0;

**for** (**int** j = 0; j < columns; j++)

temp += Math.Abs(matrix[i, j]);

**if** (temp > max)

max = temp;

}

**return** max;

}

**public** **static** **int** apriorCount(**double** alpha)

{

**return** (**int**)Math.Round(Math.Log(precision \* (1 - alpha) / matrixNorm(MatrixOperations(x, y, **false**)), alpha)) + 1;

}

**static** **void** Main()

{

**int** count = 0;

matrix = transMat(matrix, y);

**double** alpha = matrixNorm(matrix);

**int** apr = apriorCount(alpha);

**double**[,] x0 = MatrixOperations(MatVecMultiplication(matrix, x), y);

**while** (matrixNorm(MatrixOperations(x, x0, **false**)) > precision)

{

x0 = x;

x = MatrixOperations(MatVecMultiplication(matrix, x), y);

count++;

}

Console.WriteLine($"Коэффициент сжатия:\t{alpha}");

Console.Write($"Последняя итерация №{count}:\t{{ ");

**foreach** (**double** el **in** x)

Console.Write(el.ToString() + ' ');

Console.Write("}\n'");

Console.WriteLine($"Априорная оценка количества итераций:\t{apr}\n");

}

## Результаты

**Коэффициент сжатия: 0,7**

**Последняя итерация №10: { -0,132128749 0,704732598 -1,453841859 }**

**Априорная оценка количества итераций: 26**

# 

# ПРИМЕНЕНИЕ ПРИНЦИПА СЖИМАЮЩИХ ОТОБРАЖНИЙ ДЛЯ РЕШЕНИЯ ИНТЕГРАЛЬНЫХ УРАВНЕНИЙ

## Теоритические сведения

Пусть задано линейное неоднородное уравнение Фредгольма

**Теорема.** *Пусть – непрерывная функция на множестве и , тогда для любого* такого, что интегральное уравнение Фредгольма второго рода имеет единственное решение для любой правой части

Процесс последовательных приближений строится по формуле

Пусть задано линейное неоднородное уравнение Вольтерра

**Теорема.** *Пусть – непрерывная функция по переменным и . Тогда для любой и любого из поля интегральное уравнение Вольтерра второго рода имеет единственное решение.*

## Постатовка задачи №1

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

## Решение задачи №1

При к интегральному уравнению можно применить принцип сжимающихся отображений.

Пусть

**Приближённое решение:**

Пусть

**Точное решение:**

Сравним приближенные и точное решение в пространствах

## Постановка задачи №2

**Задание 2.** Методом последовательных приближений найти решение уравнения Вольтерра второго рода в пространстве

## Решение задачи №2