|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | |  |  |  |
| МИНОБРНАУКИ РОССИИ | | | |  |
| |  | | --- | |  | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | | | |  |
|  | **Институт кибернетики**  *(наименование института)* | | | |
|  | **Кафедра высшей математики**  *(наименование кафедры)* | | | |

|  |  |
| --- | --- |
| **Отчет по лабораторной работе №3** | |
| **по дисциплине** | |
| **«** Численные методы **»** | |
| **Вариант 13** | |
| Студент 3-го курса  группы КМБО-2-16 | Савин В. О. |
| Преподаватель | Даева С.Г. |
| Рецензент |  |
|  |  |

|  |  |  |
| --- | --- | --- |
| Работа представлена к защите | «\_\_»\_\_\_\_\_\_\_2018 г. |  |
|  |  |  |
| «Допущен к защите» | «\_\_»\_\_\_\_\_\_\_2018 г. |  |

Москва 2018

Содержание

Задание 3

Теоретическая часть 3

Практическая часть 4

Приложения 6

# Задание

Решить систему нелинейных уравнений с точностью до 0.001 методами:

1) Итерации

2) Ньютона(модифицированный)

## Теоретическая часть

Пусть дана система ***n***нелинейныхуравнений с ***n***неизвестными. Общий вид системы:

или где - нелинейные функции.

Требуется найти такой вектор который при подстановки в систему превращает каждое уравнение в верное числовое равенство.

Все функции непрерывны и соответственно дифференцируемы по всем своим неизвестным в некоторой выпуклой области существования неизвестной. Под выпуклой областью понимается такая область, в которой производная по каждой неизвестной не меняет свои знаки.

Методы решения таких систем, которые используют анализ частных производных называют аналитическими, а методы последовательных приближений — итерационными.

К итерационным методам относится «метод простой итерации». Для применения этого метода требуется преобразовать нашу систему нелинейных уравнений к виду:

или

Итеративный цикл имеет вид:

- начальное приближение

Если существует , то - это вектор, который содержит корни этой системы.

Окончание итерационного процесса произойдет при достижении точности и при условии:

К аналитическим методам относится «модифицированный метод Ньютона». Формула для нахождения решения:

где -матрица Якоби в начальном приближении **x(0)**

причем а - обратная матрица Якоби в начальном приближении **x(0)** соответственно.

Матрица не изменяется от итерации к итерации.

Окончание итерационного процесса произойдет при достижении точности и при условии:

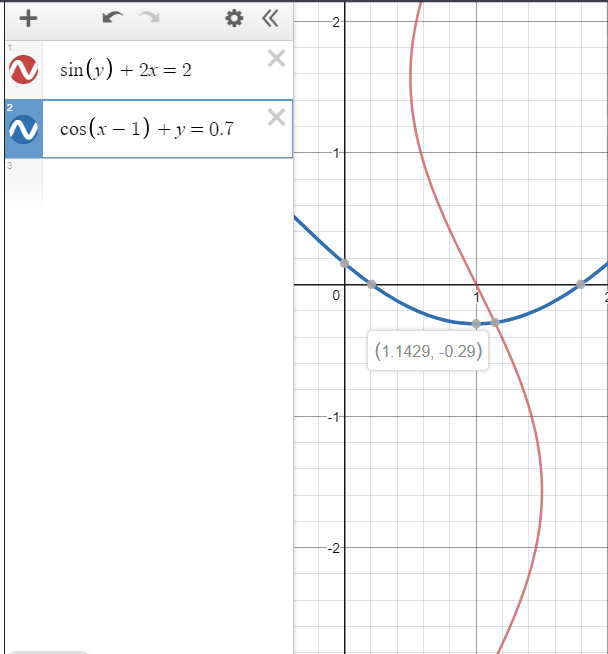
**Практическая часть**

Для решения нелинейных систем уравнений методами «простой итерации» и «модифицированный методом Ньютона» были разработаны программы «Lab3.1» и «Lab3.2» соответственно. Программа написана на языке С в операционной системе «Linux» с использованием компилятора «GCC». Для отладки проводились тесты программы на примерах с заранее известным ответом.

*Код каждой из программ размещен в разделе «Приложения»*

**1)** Преобразуем исходную систему, чтобы ее можно было решить методом простой итерации:

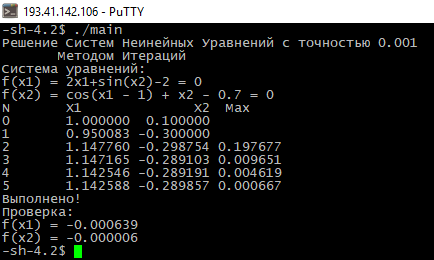
Графически уточним корни:



Возьмем начальное приближение

Организуем итерационный цикл:

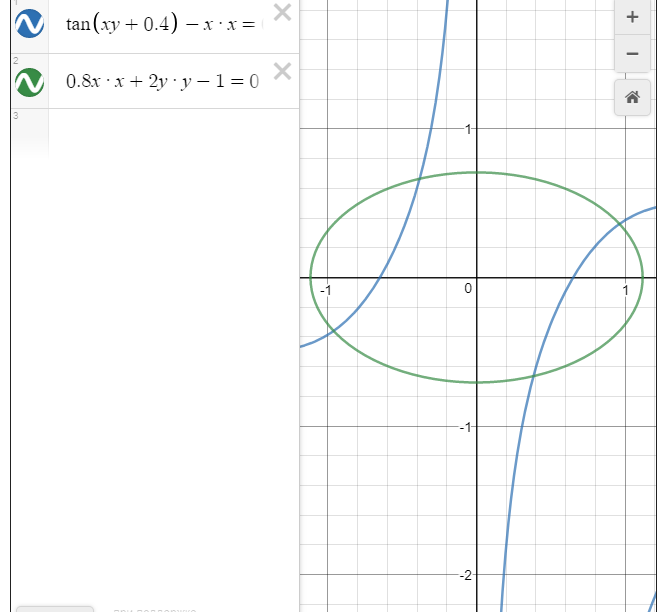
Вывод программы:

****

**Ответ:**

**2)** Решим систему модифицированным методом Ньютона.

Графически уточним корни:



Найдем корень на отрезке [0.9, 1].

Напишем матрицу Якоби для нашей системы:

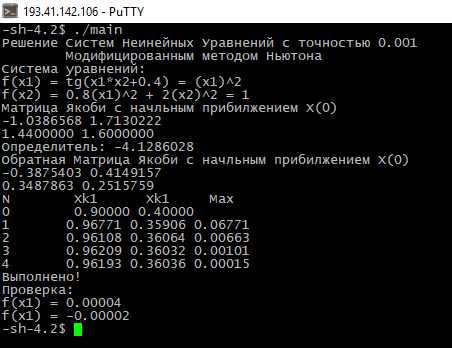
В качестве начального приближения возьмем:

Определитель отличен от нуля, значит существует обратная матрица.

Теперь запишем обратную матрицу Якоби в начальном приближении :

Организуем итерационный процесс:

Вывод программы:



**Ответ:**

**Приложения**

***Код программы «Lab3.1»***

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

double First(double x2)

{

return 1 - 0.5\*sin(x2);

}

double Second(double x1)

{

return 0.7 - cos(x1 - 1);

}

double calcFirst(double x1, double x2)

{

return 2 \* x1 + sin(x2) - 2;

}

double calcSecond(double x1, double x2)

{

return cos(x1 - 1) + x2 - 0.7;

}

int main()

{

printf( "Решение Систем Неинейных Уравнений с точностью 0.001 \n Методом Итераций\n");

printf("Система уравнений:\nf(x1) = 2x1+sin(x2)-2 = 0\nf(x2) = cos(x1 - 1) + x2 - 0.7 = 0\n");

const int SIZE = 2;

float vec[SIZE];

vec[0] = 1.0;

vec[1] = 0.1;

double matrix[SIZE][SIZE];

double iterArr[100][SIZE];

int i,j;

for ( i = 0; i < 100; ++i)

for (j = 0; j < SIZE; ++j)

{

iterArr[i][j] = 0;

}

iterArr[0][0] = First(vec[1]);

iterArr[0][1] = Second(vec[0]);

printf("N X1 X2 Max\n");

printf("0 %.6f %.6f\n",vec[0],vec[1]);

printf("1 %.6f %.6f\n",iterArr[0][0],iterArr[0][1]);

int n = 1;

double max = 0;

double max1 = 0;

double max2 = 0;

int c;

for (c = 1;; c++)

{

int i;

for (i = 0; i < SIZE; ++i)

{

if (i == 0) iterArr[c][0] = First(iterArr[c - 1][1]);

if (i == 1) iterArr[c][1] = Second(iterArr[c - 1][0]);

if (i == 0)

{

printf("%d ",c+1);

double k1 = fabs(iterArr[c][i] - iterArr[c - 1][i]);

if (k1 > max1) max1 = k1;

}

if (i == 1)

{

double k2 = fabs(iterArr[c][i] - iterArr[c - 1][i]);

if (k2 > max2) max2 = k2;

}

double k = fabs(iterArr[c][i] - iterArr[c - 1][i]);

if (k > max) max = k;

printf("%.6f ",iterArr[c][i]);

}

printf("%.6f",max);

if (max < 0.001)

{

printf("\nВыполнено!\n");

printf("Проверка: \n");

printf("f(x1) = %.6f\n",calcFirst(iterArr[c][0], iterArr[c][1]));

printf("f(x2) = %.6f\n",calcSecond(iterArr[c][0], iterArr[c][1]));

break;

}

else

{

max = 0;

max1 = 0;

max2 = 0;

}

printf("\n");

}

return 0;

}

***Код программы «Lab3.2»***

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

double First(double x1, double x2)

{

return tan(x1\*x2 + 0.4) - x1\*x1;

}

double Second(double x1, double x2)

{

return 0.8\*x1\*x1 + 2 \* x2\*x2 - 1;

}

double derX1First(double x1, double x2)

{

return x2 / (cos(x1\*x2 + 0.4)\*cos(x1\*x2 + 0.4)) - 2 \* x1;

}

double derX2First(double x1, double x2)

{

return x1 / (cos(x1\*x2 + 0.4)\*cos(x1\*x2 + 0.4));

}

double derX1Second(double x1)

{

return 1.6\*x1;

}

double derX2Second(double x2)

{

return 4 \* x2;

}

int main()

{

int i,j,c;

printf( "Решение Систем Неинейных Уравнений с точностью 0.001 \n Модифицированным методом Ньютона\n");

printf("Система уравнений:\nf(x1) = tg(x1\*x2+0.4) = (x1)^2\nf(x2) = 0.8(x1)^2 + 2(x2)^2 = 1\n");

const int SIZE = 2;

long double vec[SIZE];

vec[0] = 0.9;

vec[1] = 0.4;

double matrix[SIZE][SIZE];

matrix[0][0] = derX1First(vec[0], vec[1]);

matrix[0][1] = derX2First(vec[0], vec[1]);

matrix[1][0] = derX1Second(vec[0]);

matrix[1][1] = derX2Second(vec[1]);

printf( "Матрица Якоби с начльным прибилжением Х(0)\n");

for (i = 0; i < SIZE; ++i)

{

for (j = 0; j < SIZE; ++j)

{

printf("%.7f ", matrix[i][j]);

}

printf("\n");

}

double det = matrix[0][0] \* matrix[1][1] - matrix[0][1] \* matrix[1][0];

printf("Определитель: %.7f \n",det);

double temp1 = matrix[0][0];

matrix[0][0] = matrix[1][1];

matrix[1][1] = temp1;

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

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

for (i = 0; i < SIZE; ++i)

for (j = 0; j < SIZE; ++j)

{

matrix[i][j] = (1 / det)\*matrix[i][j];

}

printf("Обратная Матрица Якоби с начльным прибилжением Х(0)\n");

for ( i = 0; i < SIZE; ++i)

{

for ( j = 0; j < SIZE; ++j)

{

printf("%.7f ", matrix[i][j]);

}

printf("\n");

}

double F[2];

F[0] = First(vec[0], vec[1]);

F[1] = Second(vec[0], vec[1]);

double iterArr[100][SIZE];

for ( i = 0; i < 100; ++i)

for ( j = 0; j < SIZE; ++j)

{

iterArr[i][j] = 0;

}

for ( i = 0; i < SIZE; ++i)

{

iterArr[0][i] = vec[i];

}

printf("N Xk1 Xk1 Max\n");

printf("0 %.5f %.5f\n", iterArr[0][0],iterArr[0][1]);

int n = 0;

double max = 0;

for (c = 1;; c++)

{

for (i = 0; i < SIZE; ++i)

{

for ( j = 0; j < SIZE; ++j)

{

iterArr[c][i] += matrix[i][j] \* F[j];

}

iterArr[c][i] = iterArr[c - 1][i] - iterArr[c][i];

vec[i] = iterArr[c][i];

double k = fabs(iterArr[c][i] - iterArr[c - 1][i]);

if (k > max) max = k;

if (i == 0) printf("%d ", c);

printf("%.5f ",iterArr[c][i]);

}

F[n] = First(vec[0], vec[1]);

F[++n] = Second(vec[0], vec[1]);

n = 0;

printf("%.5f ", max);

if (max < 0.001)

{

printf("\nВыполнено!\n");

printf("Проверка: \n");

printf("f(x1) = %.5f\n", First(vec[0], vec[1]));

printf("f(x1) = %.5f\n",Second(vec[0], vec[1]));

break;

}

else max = 0;

printf("\n");

}

return 0;

}