## группа 4311

Хасаншин Ш.Р Ахметзянов Д.

Лабораторная работа по Вычислительной математике №3-4 (Отчет)

**ИТЕРАЦИОННЫЕ МЕТОДЫ РЕШЕНИЯ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

## Цель работы: научиться решать системы нелинейных уравнений (СНУ)

методом простых итераций (МПИ) и методом Ньютона с помощью ЭВМ. Содержание работы:

## Изучить МПИ и метод Ньютона для решения систем нелинейных урав- нений.

1. На конкретном примере усвоить порядок решения систем нелинейных уравнений МПИ и методом Ньютона с помощью ЭВМ.

## Составить программу и с ее помощью решить систему уравнений с точностью   0.001.

1. Изменить

##    /100

и снова решить задачу. Сделать вывод о влиянии

## точности на количество итераций.

1. Составить отчет о проделанной работе.

## Задание. (Вариант 20)

1. Аналитически решить СНУ вида:

## *x* + *y2* -2 0;

*x*  *y*  2  0



## (1)

1. Построить рабочие формулы МПИ и метода Ньютона для численного решения системы (1) при начальном приближении

(*x*(0), *y*(0)) = ( 2, 2 )

(2)

## Составить программу (программы) на любом языке программирования, реализующие построенные итерационные процессы.

Решение.

## Аналитическим решением СНУ (1) являются точки

( 2, 0) и ( 1 , 1 ).

## Для построения рабочих формул МПИ для численного решения систе- мы (1) необходимо вначале привести ее к виду:

*x*  1*x*, *y*

 *y*   2



*x*, *y*

(3)

## Для этого умножим первое уравнение системы (1) на неизвестную посто-

янную

##  , второе - на  , затем сложим их и добавим в обе части уравнения *x* .

Получим первое уравнение преобразуемой системы

*x* = *x* + *α f*1(*x*, *y*) + *βf*2(*x*, *y*) = *x* + *α*(*x* + *y2* - 2) + *β*(*x*  *y*  2) = Φ1(*x*, *y*)

## (4)

Далее, умножим первое уравнение системы (1) на неизвестную постоянную  ,

## второе - на  , затем сложим их и добавим в обе части уравнения *y* . Тогда вто-

рое уравнение преобразуемой системы будет иметь вид:

*y* = *y* + *γ f*1(*x*, *y*) + *δf*2(*x*, *y*) = *y* + *γ* (*x* + *y2* -2) + *δ*(*x*  *y*  2) = Φ2(*x*, *y*),

## (5)

Неизвестные постоянные

## , , , 

определим из достаточных условий

## сходимости итерационного процесса:

  1

1

*x*

 2

*x*

## и

  1

## .

1

*y*

 2

*y*

Запишем эти условия более подробно:

1   *f*1  

*x*

  *f*1    1

*x*

*f* 2

*x*

*f* 2

*x*

 *f*1  

*y*

##  1   *f*1    1

*y*

*f* 2

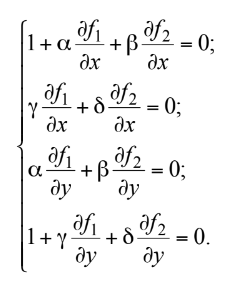
*y*

*f* 2

*y*

## Полагая равными нулю выражения под знаком модуля, получим систему линейных алгебраических уравнений (СЛАУ) 4 порядка с 4 неизвестными

, , ,  :



## (6)

## Для решения системы (6) необходимо вычислить частные производные

*δf*1 *δx*

= 1,

*δf*2 *δx*

= 1,

*δf*1 *δy*

=2y,

*δf*

*δy* = 1

3

*δf*1 *δy*

= 2*y* | *y* = 2 = 4, подставим в (1)

## Тогда СЛАУ (6) запишется так:

1 + *α* + *β* = 0

*γ* + *δ* = 0

*4α + β* = 0

# 1+ 4*γ* + *δ* = 0

Решением этой системы являются следующие значения:

Тогда рабочие формулы (4), (5) для решения СНУ (1) примут вид:

Для реализации на ЭВМ рабочие формулы можно переписать так:

k = 0,1,2,... (7)

## Итерационный процесс (7) можно начать, задав начальное приближение

(2). Процесс (7) заканчивается при одновременном выполнении двух условий:

*xk* 1  *xk*   и

*yk* 1  *yk*

##   . В этом случае значения

*xk* 1

и *yk* 1

## являются при-

ближенным значением одного из решений СНУ (1).

## 3. Для построения рабочих формул метода Ньютона в виде

## 

## Найдем матрицу частных производных

=

*W*(*x*, *y*) =

*δf*1 *δx δf*2

*δx*

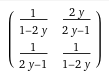
*δf*1 *δy δf*2

*δy*

Определитель этой матрицы: *detW*(*x*, *y*) = 1 - 2y

## Обратная матрица:

(*W*(*x*, *y*))−1 =



## Проведя несложные преобразования получим рабочую формулу метода Ньютона (8) в виде:

## 

Листинг программы:

#include <iostream>

#include "locale.h"

#include <math.h>

#include <iomanip>

using namespace std;

const float eps = 0.001;          // Точность (эпсилон)

const float x = 2, y = 2;       // Начальное приближение

const int rr = 10000;             // Округление

void method\_simple\_implications() // Метод простых итераций

{

    int k = 0;

    float x0 = x, y0 = y, x1 = x, y1 = y; // x0 = x(k), x1 = x(k+1)

    float modx, mody;                     // y0 = y(k), y1 = y(k+1)

    do                                    // modx: |x1 - x0|, mody: |y1 - y0|

    {

        x0 = x1, y0 = y1;

        x1 = pow(y0, 2) / 3 - 4 \* y0 / 3 + 2;

        y1 = pow(y0, 2) / -3 + 4 \* y0 / 3;

        modx = fabs(x1 - x0);

        mody = fabs(y1 - y0);

        cout << "|" << setw(4) << k << setw(4) << "|" << setw(8) << round(x0 \* rr) / rr << setw(5) << "|" << setw(10) << round(x1 \* rr) / rr << setw(5) << "|" << setw(14) << round(modx \* rr) / rr << setw(8) << "|" << setw(9) << round(y0 \* rr) / rr << setw(4) << "|" << setw(9) << round(y1 \* rr) / rr << setw(4) << "|" << setw(11)

             << round(mody \* rr) / rr << setw(7) << "|" << endl;

        k++;

    } while ((modx > eps) || (mody > eps));

}

void method\_Newton() // Метод Ньютона

{

    int k = 0;

    float x0 = x, y0 = y, x1 = x, y1 = y; // x0 = x(k), x1 = x(k+1)

    float modx, mody;                     // y0 = y(k), y1 = y(k+1)

    do                                    // modx: |x1 - x0|, mody: |y1 - y0|

    {

        x0 = x1, y0 = y1;

        x1 = (pow(y0, 2) - 4 \* y + 2) / (1 - 2 \* y);

        y1 = -(pow(y0, 2)) / (1 - 2 \* y);

        modx = fabs(x1 - x0);

        mody = fabs(y1 - y0);

        cout << "|" << setw(4) << k << setw(4) << "|" << setw(8) << round(x0 \* rr) / rr << setw(5) << "|" << setw(10) << round(x1 \* rr) / rr << setw(5) << "|" << setw(14) << round(modx \* rr) / rr << setw(8) << "|" << setw(9) << round(y0 \* rr) / rr

             << setw(4) << "|" << setw(9) << round(y1 \* rr) / rr << setw(4) << "|" << setw(11)

             << round(mody \* rr) / rr << setw(7) << "|" << endl;

        k++;

    } while ((modx > eps) || (mody > eps));

}

void show\_table()

{

    cout << "-------------------------------------------------------------------------------------------------------\n";

    cout << "| k | X(k) | X(k + 1) | |(X(k+1) - X(k))| | Y(k) | Y(k+1) | |Y(k+1) - Y(k)| |\n";

    cout << "-------------------------------------------------------------------------------------------------------\n";

}

void func()

{

    int n;

    while (true)

    {

        cout << "Команда 1 - поиск корней на отрезке по МПИ" << endl;

        cout << "Команда 2 - поиск корней на отрезке по Методу Ньютона" << endl;

        cout << "Введите команду: ";

        cin >> n;

        switch (n)

        {

        case 1:

            cout << "\nМетод простых итераций" << endl;

            show\_table();

            method\_simple\_implications();

            cout << "\nПроцесс поиска корня на отрезке прошел успешно !" << endl;

            break;

        case 2:

            cout << "\nМетод Ньютона" << endl;

            show\_table();

            method\_Newton();

            cout << "\nПроцесс поиска корня на отрезке прошел успешно !" << endl;

            break;

        default:

            cout << "Ошибка ввода !";

            break;

        }

        cout << "\n";

    }

}

int main()

{

    setlocale(0, "");

    cout << "Лабораторная работа по Вычислительной математике No3-4\n"

         << endl;

    cout << "Выполнили студенты группы 4311 Ахметзянов Дамир и Хасаншин Шамиль\n"

         << endl;

    cout << "Вариант No12\nСистема нелинейных уравнений:" << endl;

    cout << "x + y^2 - 2 = 0\nx + y - 2 = 0\nНачальное приближение: (2, 2)\n"

         << endl;

    func();

    system("pause");

    return 0;

}

## Таблица результатов:

