Федеральное агентство связи
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Сибирский государственный университет
телекоммуникаций и информатики»

Кафедра ПМиК

Лабораторная работа №4 по дисциплине «Вычислительная математика» по теме: «Многомерный вариант метода Ньютона»

Выполнил:
ст. гр. ИВ-823
Шиндель Э. Д.

Проверила:
Ассистент
Кафедры ПМиК
Петухова Я. В.

Новосибирск, 2020

Содержание:

1. Изложение метода

МПД и МХ применимы только для решения НУ, метод Ньютона может быть легко видоизменен, и его можно применять для решения СНУ.
Рассмотрим СНУ n на n (n – уравнений, n – неизвестных):

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \dots \dots \dots \dots \dots \dots \dots \dots \dots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases} \qquad F = \begin{pmatrix} f_1 \\ f_2 \\ \dots \\ f_n \end{pmatrix}$$

F(X) = 0, X = $(x_1, x_2, \dots, x_n)$

При решении СНУ поступаем таким же образом, как и при решении НУ:
1) Выбираем стартовую точку x_0, достаточно близкую к корню.
2) В одномерном варианте мы заменяли функцию на касательную и приравнивали её к нулю. Аналогичным образом поступаем и для функции многих переменных, только там заменяем f'на дифференциал, т.е.:
$f(x) = f(x_0) + f'(x_0)(x - x_0) = 0$
$F(X) = F(X_0) + \partial F|x_0 \cdot (X - X_0) = 0$
Решаем данное уравнение относительно X:
$F(X_0) + \partial F|x_0 \cdot (X - X_0) = 0$
$\partial F|x_0 \cdot (X - X_0) = -F(X_0)$
$\partial F|x_0 = W$
W – матрица частных производных (матрица Якоби)
умножим на матрицу обратную матрице W слева:
$X - X_0 = -W^{-1}|x_0 \cdot F(X_0)$

Окончательный вид формулы многомерного варианта метода Ньютона:
$X^{k+1} = X^k - W^{-1}|x_k \cdot F(X^k)$

2. Пример решения методом Ньютона

$$\begin{cases} 2x^2 + y = 4 \\ x^2 + \ y^2 = 4 \end{cases}$$

e = 0,0001 (точность)

Приводим к виду F(X)=0:

$$\begin{cases} 2x^2 + y - 4 = 0 \\ x^2 + \ y^2 - 4 = 0 \end{cases}$$

$$F = \begin{pmatrix} 2x^2 + y - 4 \\ x^2 + \ y^2 - 4 = 0 \end{pmatrix} \qquad W = \begin{pmatrix} 4x & 1 \\ 2x & 2y \end{pmatrix},$$

в качестве стартовой точки возьмем: $X^0 = \begin{pmatrix} 1 \\ 1{,}5 \end{pmatrix}$

Сделаем один шаг по многомерному методу Ньютона:

$$F(X^0) = \begin{pmatrix} -0{,}5 \\ -0{,}75 \end{pmatrix} \quad W|x_0 = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix} \qquad W^{-1}\Big|x_0 = \begin{pmatrix} 0{,}3 & -0{,}1 \\ -0{,}2 & 0{,}4 \end{pmatrix}$$

$$X^1 = X^0 - W^{-1}|x_0 \cdot F(X^0)$$

$$X^1 = \begin{pmatrix} 1 \\ 1{,}5 \end{pmatrix} - \begin{pmatrix} 0{,}3 & -0{,}1 \\ -0{,}2 & 0{,}4 \end{pmatrix} \cdot \begin{pmatrix} -0{,}5 \\ -0{,}75 \end{pmatrix} = \begin{pmatrix} 1{,}1 \\ 1{,}608 \end{pmatrix}$$

Отсюда последовательно получаем:

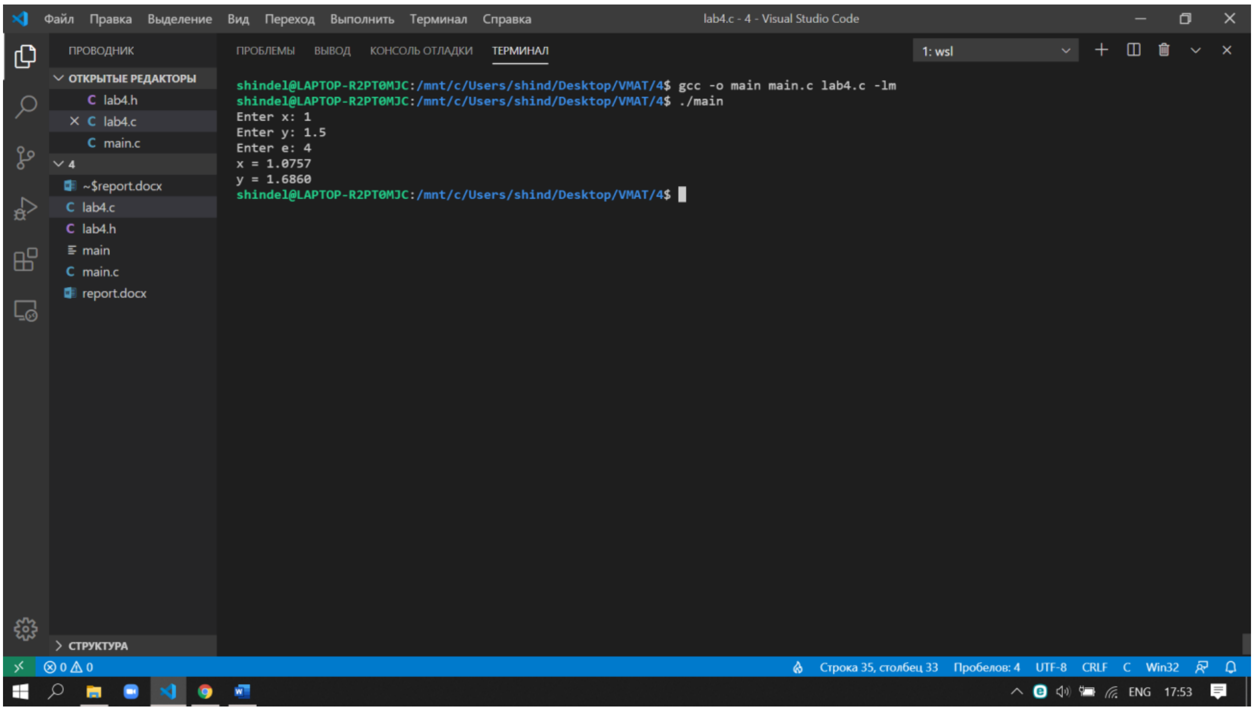
$$X^2 = \begin{pmatrix} 1{,}0948 \\ 1{,}6477 \end{pmatrix};\ X^3 = \begin{pmatrix} 1{,}0864 \\ 1{,}6665 \end{pmatrix};\ X^4 = \begin{pmatrix} 1{,}0864 \\ 1{,}6665 \end{pmatrix};\ X^5 = \begin{pmatrix} 1{,}0786 \\ 1{,}6808 \end{pmatrix};$$

$$X^6 = \begin{pmatrix} 1{,}0771 \\ 1{,}6834 \end{pmatrix};\ X^7 = \begin{pmatrix} 1{,}0764 \\ 1{,}6847 \end{pmatrix};\ X^8 = \begin{pmatrix} 1{,}0760 \\ 1.6854 \end{pmatrix};\ X^9 = \begin{pmatrix} 1{,}0758 \\ 1{,}6858 \end{pmatrix};$$

$$X^{10} = \begin{pmatrix} 1{,}0757 \\ 1{,}6859 \end{pmatrix};\ X^{11} = \begin{pmatrix} 1{,}0757 \\ 1{,}6860 \end{pmatrix}.$$

Ответ: x = 1,0757
y = 1,6869.

3. Пример работы программы



4. Листинг

```
#include "lab4.h"


double f1(double x, double y, int i) {
    double f;
    if (i == 1) f = 2 * (x * x) + y - 4;
    else f = x * x + y * y - 4;
    return f;
}


double f2(double x, double y, int i, double dif) {
    double f2;
    if (i == 1) f2 = (f1(x + dif, y, 1) - f1(x - dif, y, 1)) / (2 * dif);
    else if (i == 2) f2 = (f1(x, y + dif, 1) - f1(x, y - dif, 1)) / (2 * dif);
    else if (i == 3) f2 = (f1(x + dif, y, 2) - f1(x - dif, y, 2)) / (2 * dif);
    else f2 = (f1(x, y + dif, 2) - f1(x, y - dif, 2)) / (2 * dif);
    return f2;
}


void inverse_matrix(double **m) {
    double dop, opr = m[0][0] * m[1][1] - m[0][1] * m[1][0];
    dop = m[0][0];
    m[0][0] = m[1][1] / opr;
    m[1][1] = dop / opr;
    m[0][1] = -m[0][1] / opr;
    m[1][0] = -m[1][0] / opr;
}


double aim(int e) {
    double dif = 1.0;
```

```
    for (int i = 0; i < e; i++) dif /= 10;
    return dif;
}

void Newton(int e, double x, double y) {
    double x0, y0, dif = aim(e);
    double **W = malloc(2 * sizeof(double *));
    W[0] = malloc(2 * sizeof(double));
    W[1] = malloc(2 * sizeof(double));

    do {
        x0 = x;
        y0 = y;
        W[0][0] = f2(x, y, 1, dif);
        W[0][1] = f2(x, y, 2, dif);
        W[1][0] = f2(x, y, 3, dif);
        W[1][1] = f2(x, y, 4, dif);
        inverse_matrix(W);
        x = x - (W[0][0] * f1(x, y, 1) + W[0][1] * f1(x, y, 1));
        y = y - (W[1][0] * f1(x, y, 2) + W[1][1] * f1(x, y, 2));
    } while ((fabs(x - x0) > dif) || (fabs(y - y0) > dif));
    printf("x = %.*f\ny = %.*f\n", e, x, e, y);
}
```