|  |  |
| --- | --- |
| МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  «Пермский государственный национальный исследовательский университет» | |
| **ЧИСЛЕННЫЕ МЕТОДЫ**  **«Решение нелинейных уравнений»** | |
| *Лабораторная работа №4* | |
| Вариант(ы) № 14, 20 | |
|  | Работу выполнили студенты группы ПМИ – 1,2  Заманов Мухтар  Толов Ярослав |
| Оценка отчета   |  |  | | --- | --- | | Конспект |  | | Сроки |  | | Попытки |  | | Замечания к отчету |  | | ИТОГО: |  | | Проверил:  профессор, доктор физико-математических наук  С. В. Русаков  “\_\_\_\_” 20\_\_ г. |
|  |  |
| Пермь 2017 | |

Оглавление

[*Задание* 3](#_Toc497436030)

[*Исходные данные* 4](#_Toc497436031)

[*Теоретическая справка* 5](#_Toc497436032)

[*Результаты тестирования* 7](#_Toc497436033)

[*Анализ скорости сходимости и точности решения рассмотренными методами* 15](#_Toc497436034)

[*Код на C++* 16](#_Toc497436035)

# *Задание*

Найти корни системы нелинейных уравнений

f1(x, y) = 0

f2(x, y) = 0

с точностью e = 10-4.

1) Приближенно определить корни геометрически.

2) Уточнить корни методом:

- простой итерации;

- Ньютона;

- градиентного спуска, сведя к нахождению минимума функции

F(x, y) = f12(x, y) + f22(x, y)

3) Провести анализ скорости сходимости и точности решения рассмотренными методами.

# *Исходные данные*

Варианты исходных данных: 14 и 20.

Вариант 14:



Вариант 20:



# *Теоретическая справка*

Постановка задачи



В векторном виде:



**Метод простой итерации**

Для применения метода требуется привести систему к равносильному виду:



или в векторной форме:

, где 

Тогда очередное приближение  вычисляется по формуле:



или



Условие остановки:



**Метод Ньютона**

Метод используется для решения систем вида (1).

Очередное приближение вычисляется по формуле:

, где



Так как процесс вычисления обратной матрицы достаточно трудоёмок, то преобразуем выражение:

 домножим слева на :



В результате получаем СЛАУ относительно , после решения которого вычисляем очередное приближение.

Условие остановки:



**Метод градиентного спуска**

Сведём задачу (1) к задаче поиска минимума функции :

Очередное приближение вычисляется по формуле:



Параметр  вычисляется следующим образом:



Здесь 

Условие остановки:

(с)Вадим

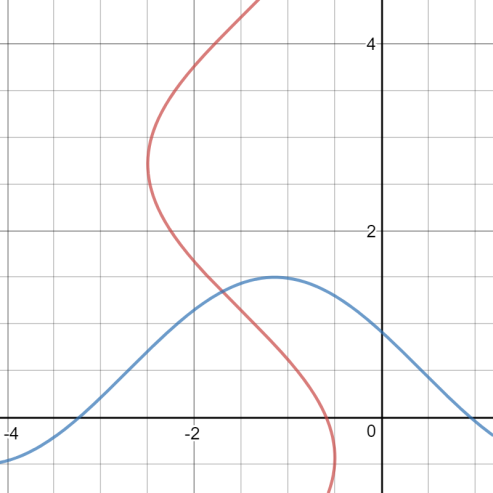
# *Результаты тестирования*

Входные данные: Вариант 20



Обозначим:

Построим графики заданных функций:



Начальное приближение зададим как: (x, y) = {-1.5, 1.5}

Якобиан:

Норма Якобиана ≈ 0.9 < 1

**Метод простой итерации:**

Функции  и :

1 XY0 (-1.5,1.5) XY (-1.85078,1.43646) ||XY-XY0|| 0.356492 ||F(x,y)|| 0.187486

2 XY0 (-1.85078,1.43646) XY (-1.79061,1.25889) ||XY-XY0|| 0.187486 ||F(x,y)|| 0.177648

3 XY0 (-1.79061,1.25889) XY (-1.61703,1.29668) ||XY-XY0|| 0.177648 ||F(x,y)|| 0.0997102

4 XY0 (-1.61703,1.29668) XY (-1.65446,1.38909) ||XY-XY0|| 0.0997102 ||F(x,y)|| 0.092242

5 XY0 (-1.65446,1.38909) XY (-1.74498,1.37134) ||XY-XY0|| 0.092242 ||F(x,y)|| 0.0509323

6 XY0 (-1.74498,1.37134) XY (-1.72773,1.32342) ||XY-XY0|| 0.0509323 ||F(x,y)|| 0.0478917

7 XY0 (-1.72773,1.32342) XY (-1.68082,1.33308) ||XY-XY0|| 0.0478917 ||F(x,y)|| 0.0267624

8 XY0 (-1.68082,1.33308) XY (-1.69032,1.3581) ||XY-XY0|| 0.0267624 ||F(x,y)|| 0.024989

9 XY0 (-1.69032,1.3581) XY (-1.71482,1.35319) ||XY-XY0|| 0.024989 ||F(x,y)|| 0.0138914

10 XY0 (-1.71482,1.35319) XY (-1.71002,1.34015) ||XY-XY0|| 0.0138914 ||F(x,y)|| 0.0130225

11 XY0 (-1.71002,1.34015) XY (-1.69726,1.34275) ||XY-XY0|| 0.0130225 ||F(x,y)|| 0.00726085

12 XY0 (-1.69726,1.34275) XY (-1.6998,1.34955) ||XY-XY0|| 0.00726085 ||F(x,y)|| 0.00679326

13 XY0 (-1.6998,1.34955) XY (-1.70646,1.3482) ||XY-XY0|| 0.00679326 ||F(x,y)|| 0.00378205

14 XY0 (-1.70646,1.3482) XY (-1.70515,1.34466) ||XY-XY0|| 0.00378205 ||F(x,y)|| 0.00354223

15 XY0 (-1.70515,1.34466) XY (-1.70167,1.34536) ||XY-XY0|| 0.00354223 ||F(x,y)|| 0.00197365

16 XY0 (-1.70167,1.34536) XY (-1.70236,1.34721) ||XY-XY0|| 0.00197365 ||F(x,y)|| 0.00184749

17 XY0 (-1.70236,1.34721) XY (-1.70417,1.34685) ||XY-XY0|| 0.00184749 ||F(x,y)|| 0.00102896

18 XY0 (-1.70417,1.34685) XY (-1.70381,1.34588) ||XY-XY0|| 0.00102896 ||F(x,y)|| 0.000963464

19 XY0 (-1.70381,1.34588) XY (-1.70287,1.34607) ||XY-XY0|| 0.000963464 ||F(x,y)|| 0.000536716

20 XY0 (-1.70287,1.34607) XY (-1.70306,1.34658) ||XY-XY0|| 0.000536716 ||F(x,y)|| 0.000502478

21 XY0 (-1.70306,1.34658) XY (-1.70355,1.34648) ||XY-XY0|| 0.000502478 ||F(x,y)|| 0.000279884

22 XY0 (-1.70355,1.34648) XY (-1.70345,1.34621) ||XY-XY0|| 0.000279884 ||F(x,y)|| 0.00026205

23 XY0 (-1.70345,1.34621) XY (-1.7032,1.34627) ||XY-XY0|| 0.00026205 ||F(x,y)|| 0.000145972

24 XY0 (-1.7032,1.34627) XY (-1.70325,1.3464) ||XY-XY0|| 0.000145972 ||F(x,y)|| 0.000136666

25 XY0 (-1.70325,1.3464) XY (-1.70338,1.34638) ||XY-XY0|| 0.000136666 ||F(x,y)|| 7.61258e-05

Ответ: (-1.70338,1.34638).

**Метод Ньютона:**

W =

1 XY0 (-1.5,1.5) XY (-1.71925,1.35955) invW\*F(x,y) (0.219254,0.140454) ||XY-XY0|| 0.260384 ||F(x,y)|| 0.0220126

2 XY0 (-1.71925,1.35955) XY (-1.7034,1.3464) invW\*F(x,y) (-0.0158566,0.0131453) ||XY-XY0|| 0.0205969 ||F(x,y)|| 0.000107253

3 XY0 (-1.7034,1.3464) XY (-1.70332,1.34634) invW\*F(x,y) (-8.00367e-05,6.30418e-05) ||XY-XY0|| 0.000101883 ||F(x,y)|| 2.74063e-09

Ответ: (-1.70332,1.34634).

**Метод градиентного спуска:**

f = 

df/dx = 

df/dy = 

Оптимальное значение параметров  и было найдено на промежутках  с шагом 0,01 и на  с шагом 0,1 соответственно.

 = 0.46  = 5.1

1 XY0 (-1.5,1.5) XY (-1.65002,1.32096) df (0.656987,0.784073) alfa 0.22835 ||XY-XY0|| 0.233588 ||F(x,y)|| 0.0597268 |f(x,y)| 0.00356729

2 XY0 (-1.65002,1.32096) XY (-1.7036,1.34541) df (0.107928,-0.049255) alfa 0.496414 ||XY-XY0|| 0.0588925 ||F(x,y)|| 0.00142531 |f(x,y)| 2.03152e-06

3 XY0 (-1.7036,1.34541) XY (-1.70324,1.3463) df (-0.00155548,-0.00389624) alfa 0.22835 ||XY-XY0|| 0.000957989 ||F(x,y)|| 8.58293e-05 |f(x,y)| 7.36667e-09

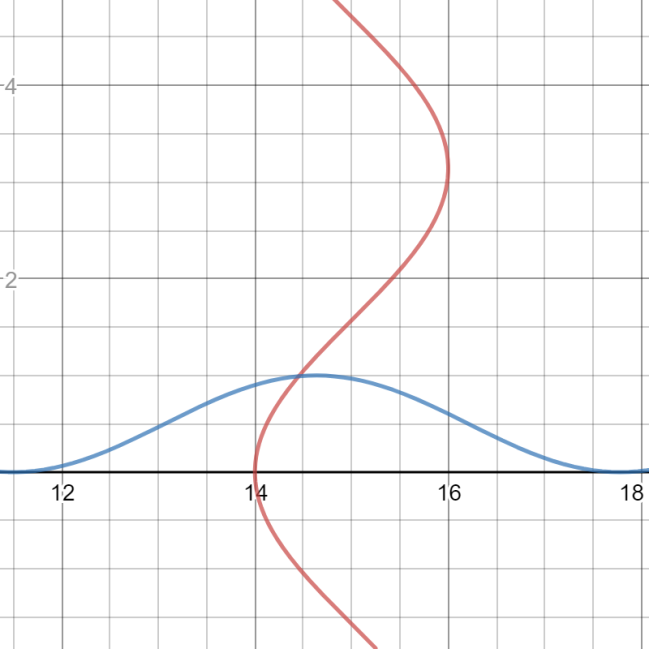
Ответ: (-1.70324,1.3463).

Входные данные: Вариант 14



Обозначим:

Построим графики заданных функций:



Начальное приближение зададим как: (x, y) = { 14, 0.5 }

Якобиан:

Норма Якобиана ≈ 0.479 < 1

**Метод простой итерации:**

Функции  и :

1 XY0 (14,0.5) XY (0.622417,0.901892) ||XY-XY0|| 13.3836 ||F(x,y)|| 0.728671

2 XY0 (0.622417,0.901892) XY (0.879873,0.561056) ||XY-XY0|| 0.427145 ||F(x,y)|| 0.336422

3 XY0 (0.879873,0.561056) XY (0.653306,0.685401) ||XY-XY0|| 0.258446 ||F(x,y)|| 0.22984

4 XY0 (0.653306,0.685401) XY (0.725835,0.576353) ||XY-XY0|| 0.130965 ||F(x,y)|| 0.095941

5 XY0 (0.725835,0.576353) XY (0.661544,0.61196) ||XY-XY0|| 0.0734924 ||F(x,y)|| 0.0661518

6 XY0 (0.661544,0.61196) XY (0.681476,0.580421) ||XY-XY0|| 0.0373093 ||F(x,y)|| 0.0264439

7 XY0 (0.681476,0.580421) XY (0.663768,0.590241) ||XY-XY0|| 0.0202485 ||F(x,y)|| 0.0182688

8 XY0 (0.663768,0.590241) XY (0.669193,0.581519) ||XY-XY0|| 0.0102719 ||F(x,y)|| 0.007203

9 XY0 (0.669193,0.581519) XY (0.664371,0.584194) ||XY-XY0|| 0.00551505 ||F(x,y)|| 0.00497863

10 XY0 (0.664371,0.584194) XY (0.665843,0.581816) ||XY-XY0|| 0.00279689 ||F(x,y)|| 0.0019553

11 XY0 (0.665843,0.581816) XY (0.664534,0.582542) ||XY-XY0|| 0.00149707 ||F(x,y)|| 0.00135165

12 XY0 (0.664534,0.582542) XY (0.664933,0.581896) ||XY-XY0|| 0.000759151 ||F(x,y)|| 0.000530274

13 XY0 (0.664933,0.581896) XY (0.664578,0.582093) ||XY-XY0|| 0.000406001 ||F(x,y)|| 0.000366579

14 XY0 (0.664578,0.582093) XY (0.664686,0.581918) ||XY-XY0|| 0.000205875 ||F(x,y)|| 0.000143773

15 XY0 (0.664686,0.581918) XY (0.66459,0.581971) ||XY-XY0|| 0.000110078 ||F(x,y)|| 9.93912e-05

Ответ: (0.66459,0.581971).

**Метод Ньютона:**

W =

1 XY0 (14,0.5) XY (-1.37796,-3.67244) invW\*F(x,y) (15.378,4.17244) ||XY-XY0|| 15.934 ||F(x,y)|| 8.28415

2 XY0 (-1.37796,-3.67244) XY (3.83451,-0.764602) invW\*F(x,y) (-5.21247,-2.90784) ||XY-XY0|| 5.9687 ||F(x,y)|| 3.84759

3 XY0 (3.83451,-0.764602) XY (-2.01927,3.27673) invW\*F(x,y) (5.85378,-4.04133) ||XY-XY0|| 7.1133 ||F(x,y)|| 7.61555

4 XY0 (-2.01927,3.27673) XY (3.18933,-1.90752) invW\*F(x,y) (-5.2086,5.18425) ||XY-XY0|| 7.34888 ||F(x,y)|| 5.425

5 XY0 (3.18933,-1.90752) XY (-3.47846,3.71721) invW\*F(x,y) (6.66779,-5.62474) ||XY-XY0|| 8.72337 ||F(x,y)|| 8.13871

6 XY0 (-3.47846,3.71721) XY (5.5304,-2.14576) invW\*F(x,y) (-9.00886,5.86298) ||XY-XY0|| 10.7487 ||F(x,y)|| 5.56834

7 XY0 (5.5304,-2.14576) XY (0.837714,-0.708588) invW\*F(x,y) (4.69269,-1.43717) ||XY-XY0|| 4.90783 ||F(x,y)|| 2.75022

8 XY0 (0.837714,-0.708588) XY (0.0792546,0.307857) invW\*F(x,y) (0.758459,-1.01644) ||XY-XY0|| 1.26824 ||F(x,y)|| 0.468383

9 XY0 (0.0792546,0.307857) XY (0.617838,0.541585) invW\*F(x,y) (-0.538584,-0.233728) ||XY-XY0|| 0.587113 ||F(x,y)|| 0.04268

10 XY0 (0.617838,0.541585) XY (0.663715,0.581562) invW\*F(x,y) (-0.045877,-0.0399773) ||XY-XY0|| 0.0608513 ||F(x,y)|| 0.000693371

11 XY0 (0.663715,0.581562) XY (0.664594,0.581926) invW\*F(x,y) (-0.000878992,-0.00036378) ||XY-XY0|| 0.000951295 ||F(x,y)|| 8.38748e-08

Ответ: (0.664594,0.581926).

**Метод градиентного спуска:**

f = 

df/dx = 

df/dy = 

Оптимальное значение параметров  и было найдено на промежутках  с шагом 0,01 и на  с шагом 0,1 соответственно.

 = 0.14  = 4.7

1 XY0 (14,0.5) XY (11.4472,1.97781) df (27.7115,-16.0422) alfa 0.09212 ||XY-XY0|| 2.94969 ||F(x,y)|| 10.3376 |f(x,y)| 106.866

2 XY0 (11.4472,1.97781) XY (-1.37393,3.1124) df (19.485,-1.7243) alfa 0.658 ||XY-XY0|| 12.8712 ||F(x,y)|| 7.29292 |f(x,y)| 53.1867

3 XY0 (-1.37393,3.1124) XY (-1.00012,0.814657) df (-4.05785,24.9429) alfa 0.09212 ||XY-XY0|| 2.32795 ||F(x,y)|| 2.43663 |f(x,y)| 5.93715

4 XY0 (-1.00012,0.814657) XY (-0.644745,-0.0279289) df (-3.85776,9.14661) alfa 0.09212 ||XY-XY0|| 0.914464 ||F(x,y)|| 1.15431 |f(x,y)| 1.33243

5 XY0 (-0.644745,-0.0279289) XY (0.783253,0.39646) df (-2.17021,-0.644968) alfa 0.658 ||XY-XY0|| 1.48973 ||F(x,y)|| 0.52825 |f(x,y)| 0.279048

6 XY0 (0.783253,0.39646) XY (0.659286,0.590382) df (1.34572,-2.1051) alfa 0.09212 ||XY-XY0|| 0.23016 ||F(x,y)|| 0.024297 |f(x,y)| 0.000590346

7 XY0 (0.659286,0.590382) XY (0.665155,0.581196) df (-0.0637111,0.0997184) alfa 0.09212 ||XY-XY0|| 0.0109009 ||F(x,y)|| 0.00223185 |f(x,y)| 4.98115e-06

8 XY0 (0.665155,0.581196) XY (0.664612,0.582035) df (0.00589676,-0.00911198) alfa 0.09212 ||XY-XY0|| 0.000999831 ||F(x,y)|| 0.000205127 |f(x,y)| 4.20771e-08

9 XY0 (0.664612,0.582035) XY (0.664656,0.581957) df (-0.000480978,0.000849439) alfa 0.09212 ||XY-XY0|| 8.99237e-05 ||F(x,y)|| 4.4791e-05 |f(x,y)| 2.00624e-09

Ответ: (0.664656,0.581957).

# *Анализ скорости сходимости и точности решения рассмотренными методами*

Вариант 20

|  |  |  |
| --- | --- | --- |
| Название метода | Количество итераций | на последней итерации |
| Простой итерации | 25 | 7.61258e-05 |
| Метод Ньютона | 3 | 2.74063e-09 |
| Градиентного спуска | 3 | 8.58293e-05 |

Вариант 14

|  |  |  |
| --- | --- | --- |
| Название метода | Количество итераций | на последней итерации |
| Простой итерации | 15 | 9.93912e-05 |
| Метод Ньютона | 11 | 8.38748e-08 |
| Градиентного спуска | 9 | 4.4791e-05 |

Метод простой итерации имеет самую медленную скорость сходимости по сравнению с остальными методами, при этом также проигрывает им в точности. С другой стороны, это самый простой метод в плане вычислительных затрат на одну итерацию и подготовки исходных данных (требует только приведения функций к специальному виду).

Градиентный метод наискорейшего спуска имеет более высокую скорость сходимости по сравнению с методом простых итераций, но проигрывает по этому показателю методу Ньютона. Метод требует вычисления частных производных исходной функции, а также сильно зависит от параметров и .

Метод Ньютона, хотя и является наиболее трудоёмким в плане вычислительных затрат,– это и вычисление частных производных для составления матрицы Якоби, и поиск обратной матрицы (или решение СЛАУ) на каждой итерации, – но сходится на порядок быстрее остальных, при этом являясь самым точным.

# *Код на C++*

#include <iostream>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include "MAtrix.h"

#include <cmath>

using namespace std;

const double e = 0.0001;

struct XY

{

double x;

double y;

double norm()

{

return sqrt(x\*x+y\*y);

}

XY operator + (XY xy)

{

return { x + xy.x,y + xy.y };

}

XY operator - (XY xy)

{

return { x - xy.x,y - xy.y };

}

XY operator \* (XY xy)

{

return { x \* xy.x,y \* xy.y };

}

XY operator \* (double a)

{

return { x \* a , y \* a };

}

XY operator / (XY xy)

{

return { x / xy.x,y / xy.y };

}

XY operator ^ (double a)

{

return { pow(x, a) , pow(y,a) };

}

XY\* print()

{

cout << "(" << x << "," << y << ")";

return this;

}

};

XY F\_14(XY xy)

{

return { cos(xy.y) + xy.x - 1.5 , 2 \* xy.y - sin(xy.x - 0.5) - 1 };

}

XY F\_20(XY xy)

{

return { sin(xy.y + 2) - xy.x - 1.5 , xy.y + cos(xy.x - 2) - 0.5 };

}

XY f\_14(XY xy)

{

return { 1.5 - cos(xy.y) , (1 + sin(xy.x - 0.5)) / 2 };

}

XY f\_20(XY xy)

{

return { sin(xy.y + 2) - 1.5 , 0.5 - cos(xy.x - 2) };

}

XY df\_14(XY xy)

{

return{ 2\*(-1.5 + xy.x + cos(xy.y) - cos(0.5 - xy.x)\*(-1 + 2\*xy.y + sin(0.5 - xy.x))) , 4\*(-1 + 2\*xy.y + sin(0.5 - xy.x)) - 2\*(-1.5 + xy.x + cos(xy.y))\*sin(xy.y) };

}

XY df\_20(XY xy)

{

return { 2\*(-0.5 + xy.y + cos(2 - xy.x))\*sin(2 - xy.x) - 2\*(-1.5 - xy.x + sin(2 + xy.y)) , 2\*(-0.5 + xy.y + cos(2 - xy.x)) + 2\*cos(2 + xy.y)\*(-1.5 - xy.x + sin(2 + xy.y)) };

}

double(\*W[2][2])(XY) = {};

int K = 0;

XY IM(XY xy0, XY(F)(XY), XY(f)(XY));

XY Newton(XY xy0, XY(F)(XY), double(\*W[2][2])(XY));

XY GradDescent(XY xy0, XY(F)(XY), XY(df)(XY), double alfa, double lambda);

double\*\* init(int n, int m, bool fillByZero=false);

void main()

{

XY(\*F)(XY) = NULL;

XY(\*f)(XY) = NULL;

XY(\*df)(XY) = NULL;

double alfa = 0, lambda = 0;

XY xy0;

switch (14)

{

case 14:

F = F\_14;

f = f\_14;

df = df\_14;

xy0 = { 14, 0.5 };

W[0][0] = [](XY a) -> double { return 1; };

W[0][1] = [](XY a) -> double { return -sin(a.y); };

W[1][0] = [](XY a) -> double { return -cos(a.x - 0.5); };

W[1][1] = [](XY a) -> double { return 2; };

alfa = 4.7;

lambda = 0.14;

break;

case 20:

F = F\_20;

f = f\_20;

df = df\_20;

xy0 = { -1.5, 1.5 };

W[0][0] = [](XY a) -> double { return -1; };

W[0][1] = [](XY a) -> double { return cos(a.y+2); };

W[1][0] = [](XY a) -> double { return -sin(a.x-2); };

W[1][1] = [](XY a) -> double { return 1; };

alfa = 5.1;

lambda = 0.46;

break;

}

IM(xy0, F, f);

cout << endl;

Newton(xy0, F, W);

cout << endl;

GradDescent(xy0, F, df, alfa, lambda);

/\*int Kold = 100;

double j = 0;

double alfamin, lambdamin;

double kf = (0.89 / 0.01)\*((99.9 - 0.4) / 0.1) / 100;

for (lambda = 0.01; lambda < 0.9; lambda += 0.01)

for (alfa = 0.5; alfa < 100; alfa += 0.1)

{

j++;

GradDescent(xy0, F, df, alfa, lambda);

//cout << K << " ";

cout << j / kf << "%" << endl;

if (K < Kold)

{

cout << alfa << " " << lambda << endl;

Kold = K;

alfamin = alfa;

lambdamin = lambda;

}

}

cout << "FINISH:"<< endl << "alfa: " << alfamin << " lambda: " << lambdamin;\*/

cout << endl;

cin.get();

cin.get();

}

XY IM(XY xy0, XY(F)(XY), XY(f)(XY))

{

double delta;

double Fdelta;

int k = 1;

XY xy = xy0;

do

{

xy = f(xy);

cout << k++ << " XY0 ";

xy0.print();

cout << " XY ";

xy.print();

delta = (xy - xy0).norm();

cout << " ||XY-XY0|| " << delta;

Fdelta = F(xy).norm();

cout << " ||F(x,y)|| " << Fdelta;

xy0 = xy;

cout << endl;

} while (e < delta && e < Fdelta);

return xy;

}

XY Newton(XY xy0, XY(F)(XY), double(\*W[2][2])(XY))

{

double delta;

XY xy = xy0;

double invW[2][2] = {};

double Fdelta;

int k = 1;

do

{

double q = W[0][0](xy);

q = W[0][1](xy);

q = W[1][0](xy);

q = W[1][1](xy);

invW[0][0] = -W[1][1](xy) / (W[0][1](xy) \* W[1][0](xy) - W[0][0](xy) \* W[1][1](xy));

invW[0][1] = W[0][1](xy) / (W[0][1](xy) \* W[1][0](xy) - W[0][0](xy) \* W[1][1](xy));

invW[1][0] = W[1][0](xy) / (W[0][1](xy) \* W[1][0](xy) - W[0][0](xy) \* W[1][1](xy));

invW[1][1] = -W[0][0](xy) / (W[0][1](xy) \* W[1][0](xy) - W[0][0](xy) \* W[1][1](xy));

XY b = F(xy);

XY sub\_res = { invW[0][0]\*F(xy).x+invW[0][1]\*F(xy).y , invW[1][0] \* F(xy).x + invW[1][1] \* F(xy).y };

cout << k++ << " XY0 ";

xy0.print();

cout << " XY ";

xy = xy0 - sub\_res;

xy.print();

cout << " invW\*F(x,y) ";

sub\_res.print();

delta = (xy - xy0).norm();

cout << " ||XY-XY0|| " << delta;

Fdelta = F(xy).norm();

cout << " ||F(x,y)|| " << Fdelta;

xy0 = xy;

cout << endl;

} while (e < delta && e < Fdelta);

return xy;

}

XY GradDescent(XY xy0, XY(F)(XY), XY(df)(XY), double alfa, double lambda)

{

double delta;

XY xy = xy0;

double Fdelta;

int k = 1;

double alfak = alfa;

do

{

alfak = alfa;

while ( (F(xy0 - df(xy0) \* alfak)^2).x + (F(xy0 - df(xy0) \* alfak)^2).y >= (F(xy0)^2).x + (F(xy0)^2).y)

{

alfak = alfak \* lambda;

}

xy = xy0 - df(xy0) \* alfak;

cout << k++ << " XY0 ";

//k++;

xy0.print();

cout << " XY ";

xy.print();

cout << " df ";

df(xy0).print();

cout << " alfa " << alfak;

delta = (xy - xy0).norm();

cout << " ||XY-XY0|| " << delta;

Fdelta = F(xy).norm();

cout << " ||F(x,y)|| " << Fdelta;

cout << " |f(x,y)| " << fabs((F(xy) ^ 2).x + (F(xy) ^ 2).y);

xy0 = xy;

cout << endl;

} while (e < delta && e < Fdelta);

K = k;

return xy;

}

double\*\* init(int n, int m, bool fillByZero)

{

double\*\* A = new double\*[n];

for (int i = 0; i < n; i++)

A[i] = new double[m];

if (fillByZero)

for (int i = 0; i < n; i++)

for (int j = 0; j < n; j++)

A[i][j] = 0.0;

return A;

}