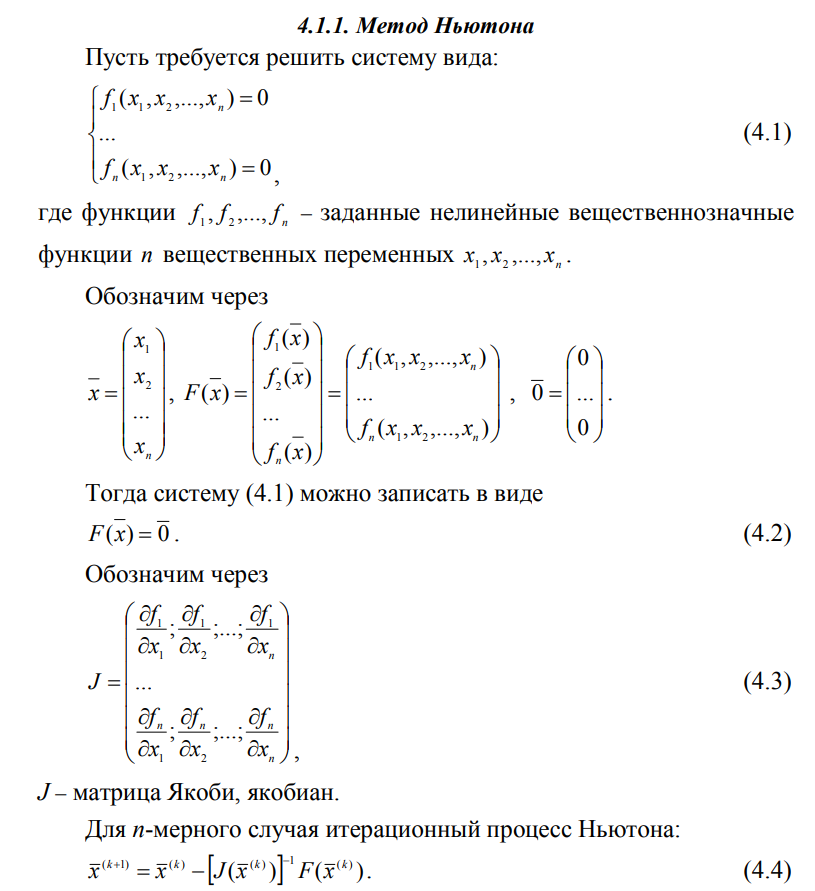
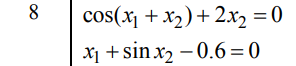
****

Моя система уравнений:



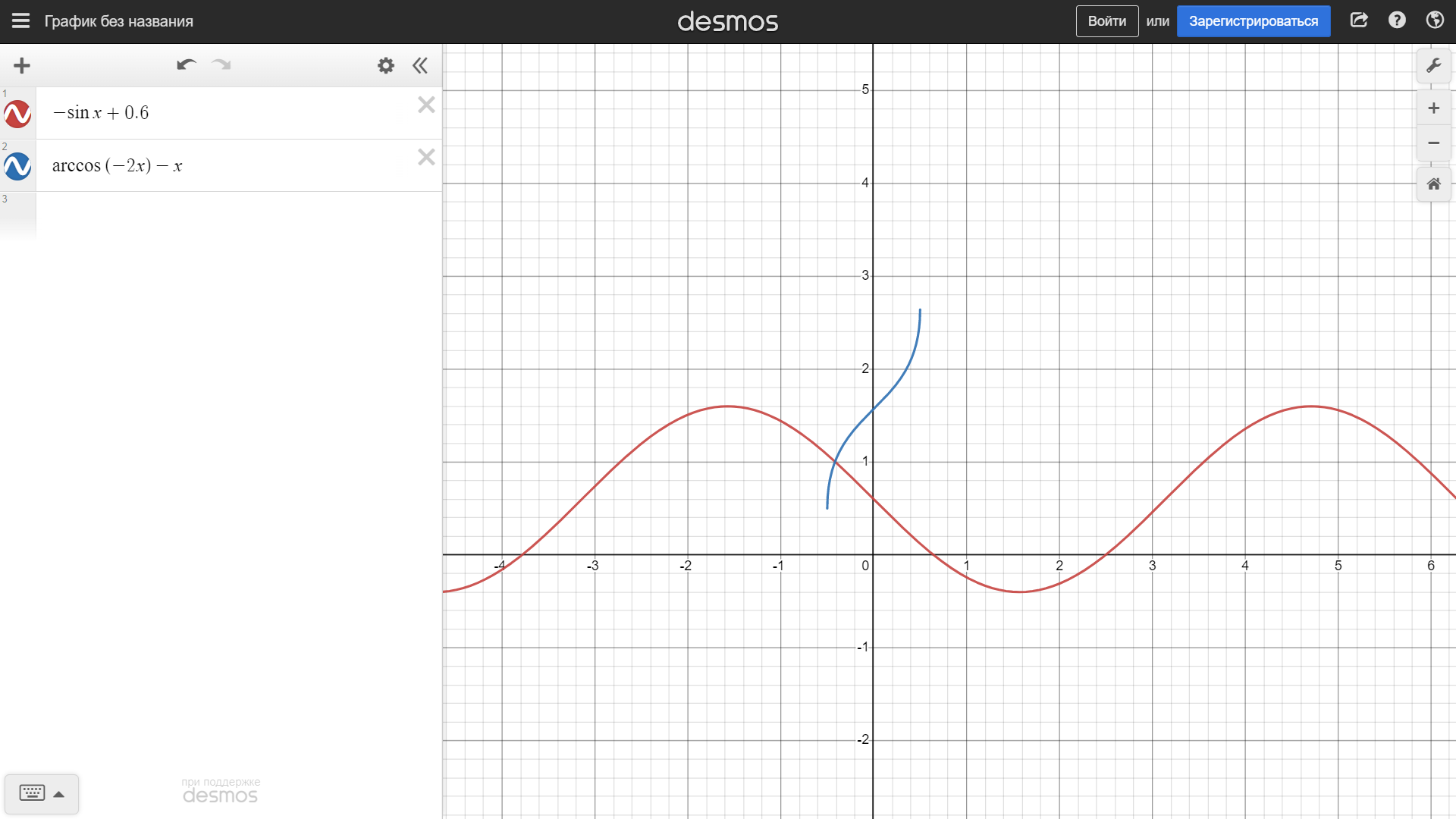
Я приму х1 за у, х2 за х для удобства дальнейшей работы.

Выразим у из каждого уравнения:

у = arccos(-2x)-x

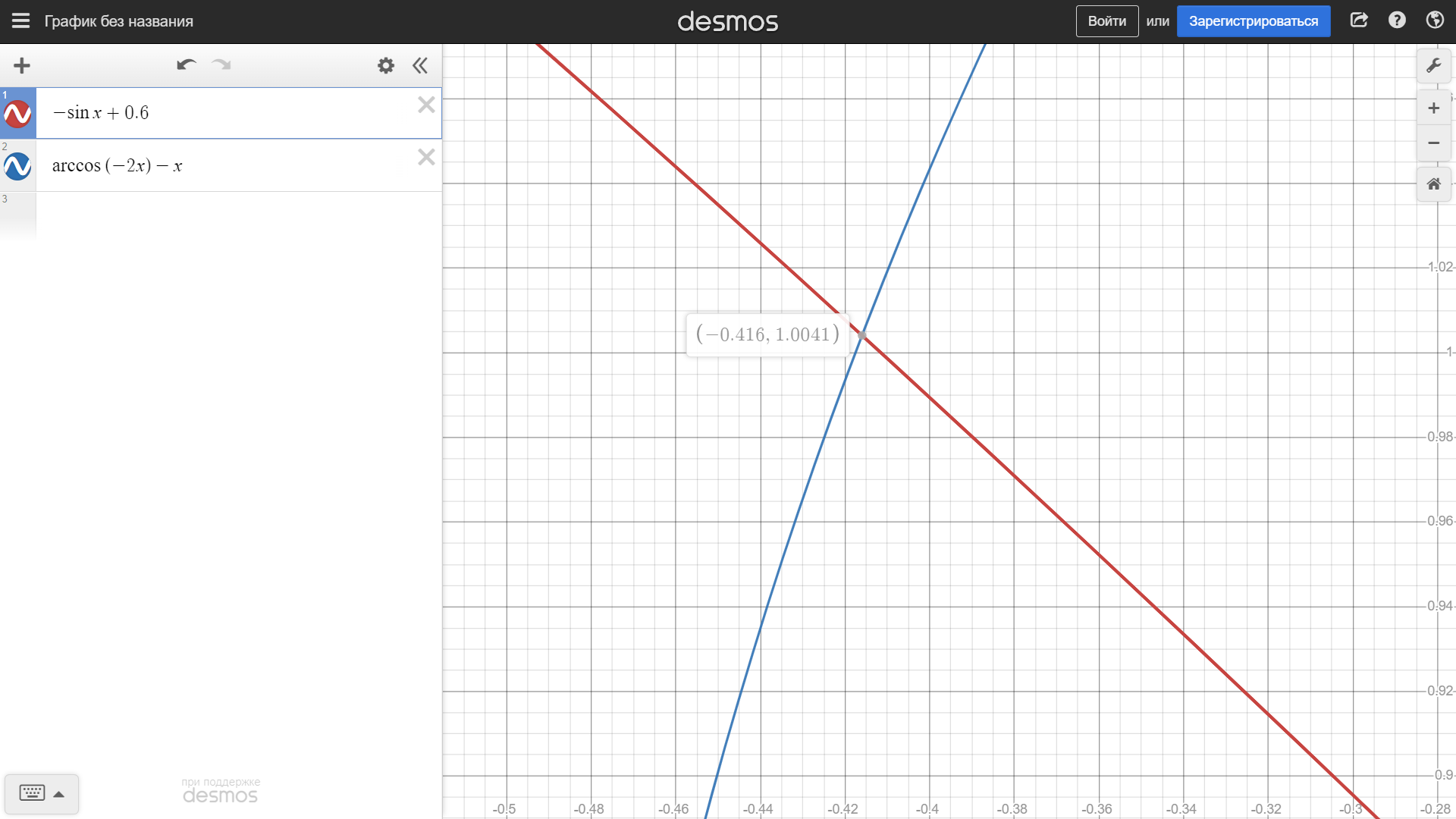
y = 0,6-sinx

и построим их графики

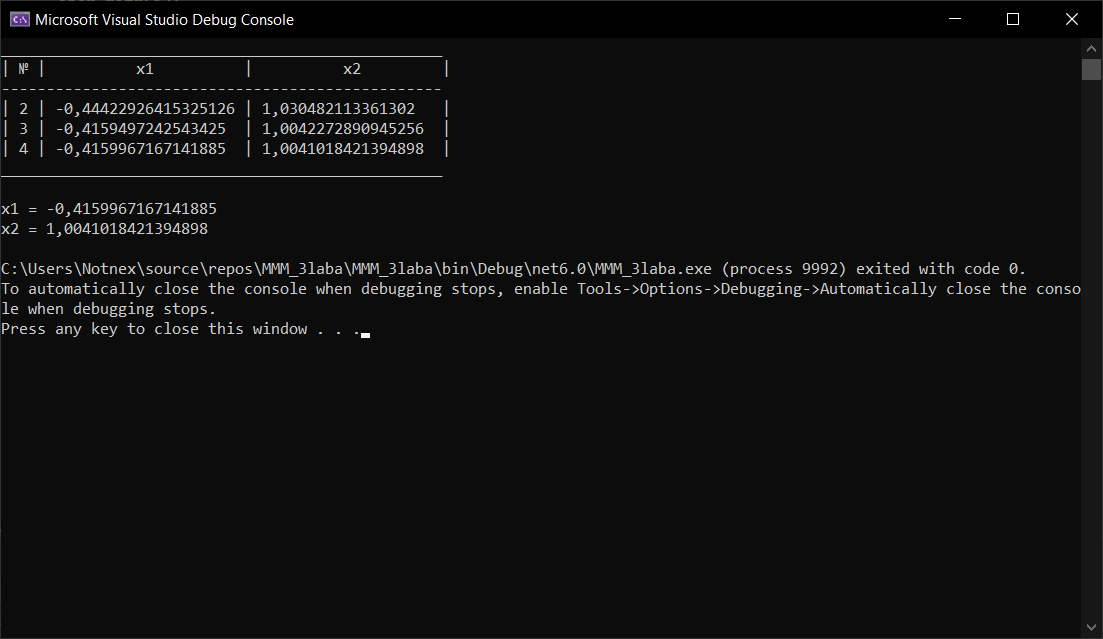
Сам график функций имеет вид: 

Если увеличить, то можно увидеть, что точка их пересечения лежит на промежутке

х = (-1;0), у = (1;2).



**Скриншот работы:**



**Код:**

double x = -0.5, y = 1.5;

nuton(x, y);

double function1(double x, double y)

{

return Math.Cos(x + y) + 2 \* x;

}

double function2(double x, double y)

{

return y + Math.Sin(x) - 0.6;

}

double func11(double x, double y)

{

return -Math.Sin(x+y) + 2;

}

double func12(double x, double y)

{

return -Math.Sin(x + y);

}

double func21(double x, double y)

{

return Math.Cos(x);

}

double func22(double x, double y)

{

return 1;

}

void ober\_matr(double[][] a)

{

double det, aa;

det = a[0][0] \* a[1][1] - a[0][1] \* a[1][0];

aa = a[0][0];

a[0][0] = a[1][1] / det;

a[1][1] = aa / det;

aa = a[0][1];

a[0][1] = -a[0][1] / det;

a[1][0] = -a[1][0] / det;

}

void nuton(double x, double y)

{

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| № | x1 | x2 |");

Console.WriteLine("-------------------------------------------------");

int i = 1;

double [][] a = new double [2][];

for (int o = 0; o < 2; o++)

{

a[o] = new double[2];

}

double [] b = new double[2];

double dx, dy, norm, eps = 1e-6;

do

{

a[0][0] = func11(x, y);

a[0][1] = func12(x, y);

a[1][0] = func21(x, y);

a[1][1] = func22(x, y);

ober\_matr(a);

dx = -a[0][0] \* function1(x, y) + -a[0][1] \* function2(x, y);

dy = -a[1][0] \* function1(x, y) + -a[1][1] \* function2(x, y);

x = x + dx;

y = y + dy;

b[0] = function1(x, y);

b[1] = function2(x, y);

norm = Math.Sqrt(b[0] \* b[0] + b[1] \* b[1]);

i++;

Console.WriteLine("| {0} | {1,-20} | {2,-19} |", i, x, y);

}

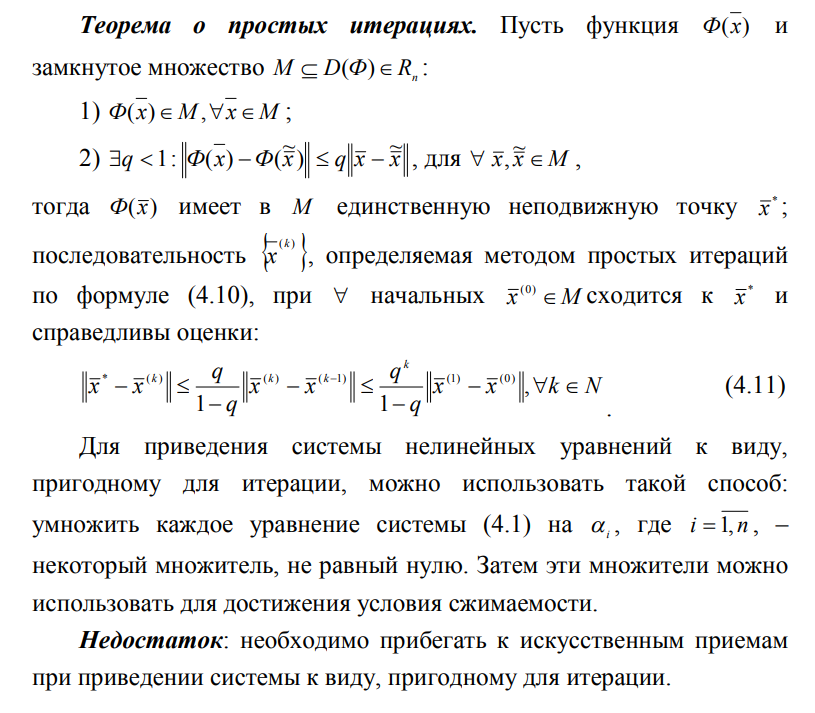
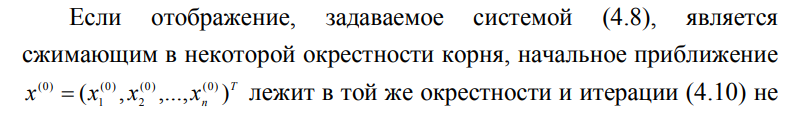
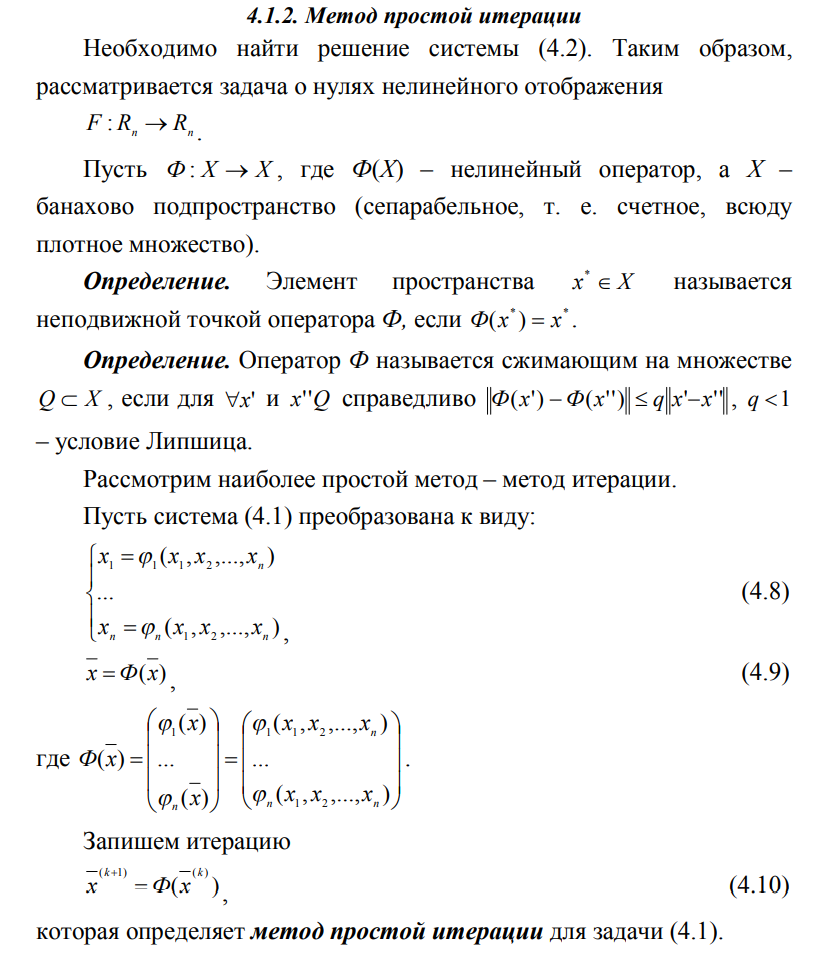
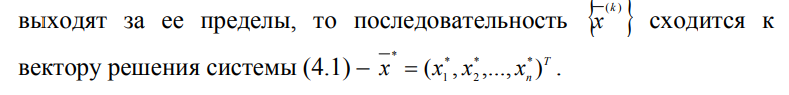
while (norm >= eps);

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

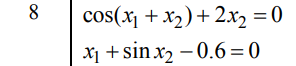
Console.WriteLine("\nx1 = "+x+"\nx2 = "+y);

}

**Вывод:** Метод Ньютона достаточно трудоемкий – на каждом шаге итерационного процесса необходимо найти матрицу, обратную якобиану, но достаточно быстро приводит к нахождению необходимого результата

****

Моя система уравнений имеет вид:



Приводим её к виду f(x):

f1(x) = cos(x+y)/2

f2(x) = 0,6 – sin(y)

Дальше найдем их производные по х и у:

f1(x)/dx = sin(x+y)/2

f1(x)/dy = sin(y+x)/2

f2(x)/dx = 0

f2(x)/dy = -cos(y)

**Код:**

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

Console.WriteLine("| № | x1 | x2 |");

Console.WriteLine("--------------------------------------------------");

double x = -0.5, y = 1.5, q1, q2, e = 1e-6, x\_0 = 0, y\_0 = 0;

q1 = Math.Abs(fi1\_dx(x, y)) + Math.Abs(fi1\_dy(x, y));

q2 = Math.Abs(fi2\_dx(x, y)) + Math.Abs(fi2\_dy(x, y));

int i = 0;

while(Math.Abs(x-x\_0) > e && Math.Abs(y-y\_0) > e)

{

x\_0 = x;

y\_0 = y;

x = fi1(x\_0, y\_0);

y = fi2(x\_0, y\_0);

i++;

Console.WriteLine("| {0,-2} | {1,-20} | {2,-20} | ", i, x, y);

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

double fi1(double x, double y)//x1(x)

{

return 0.6-Math.Sin(y);

}

double fi2(double x, double y)//x2(y)

{

return -Math.Cos(y+x)/2;

}

double fi1\_dx(double x, double y)

{

return 0;

}

double fi1\_dy(double x, double y)

{

return -Math.Cos(y);

}

double fi2\_dx(double x, double y)

{

return Math.Sin(y+x)/2;

}

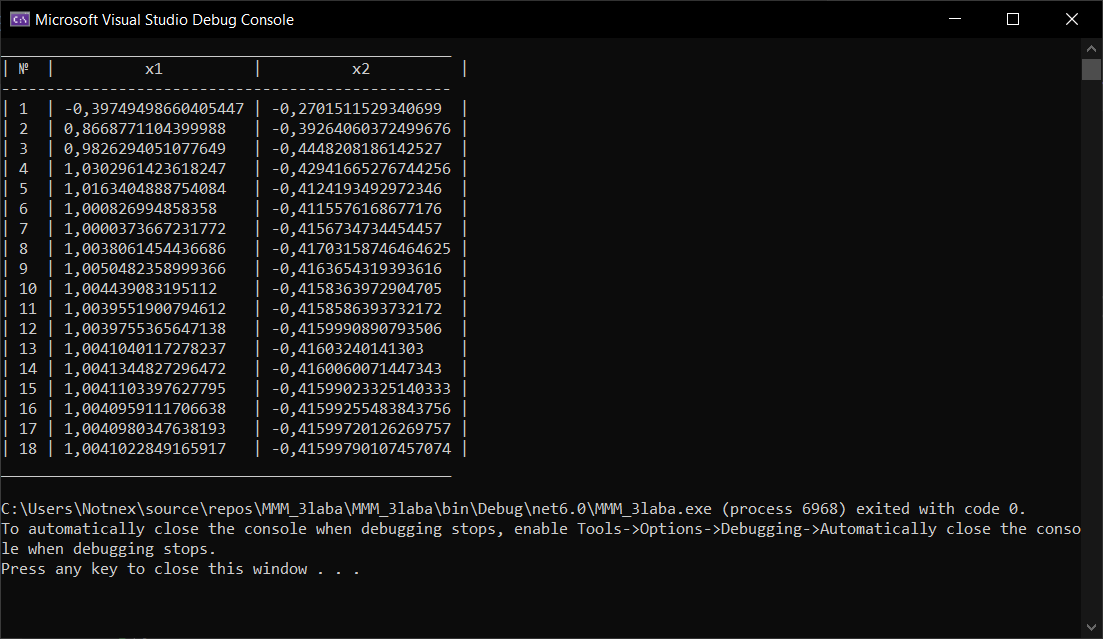
double fi2\_dy(double x, double y)

{

return Math.Sin(y + x) / 2;

}

**Скриншот:**



**Вывод:** Данный метод является более долгим, так же необходимо прибегать к искусственным приемам при приведении системы к виду, пригодному для итерации.

**Общий вывод:** Метод Ньютона хоть и является более трудоемким, но позволяет вычислить за 3 итерации корни системы. Метод итераций легче ведь уравнение достаточно привести к нужному виду лишь единожды, но вычисляет гораздо дольше (требуется 18 итераций вместо 3).