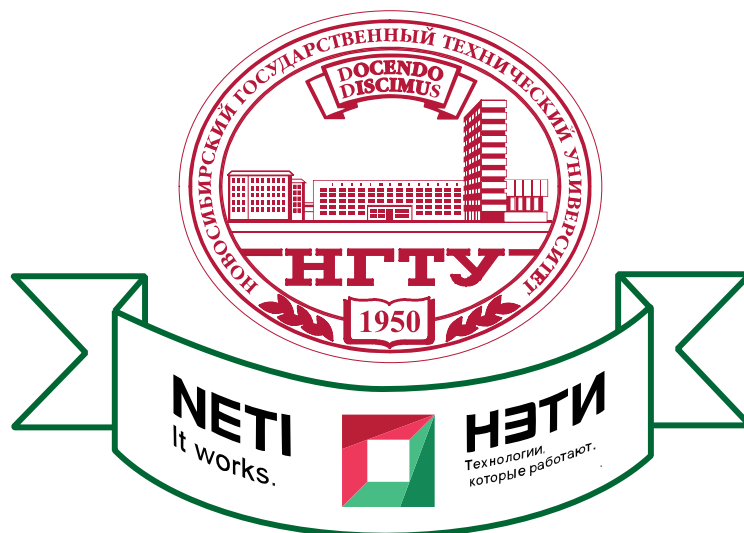


Министерство науки и высшего образования
Российской Федерации

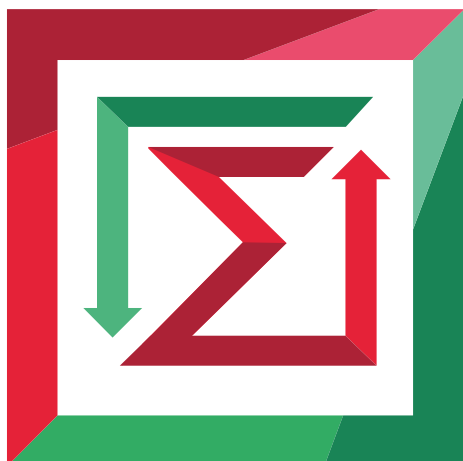
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Кафедра теоретической и прикладной информатики

Лабораторная работа № 4
по дисциплине «Численные методы»

РЕШЕНИЕ СЛУУ МЕТОДОМ НЬЮТОНА



Факультет: ПМИ
Группа: ПМ-81
Бригада: 14
Студенты: Редут Анатолий
Ефремов Артур

Преподаватели: Патрушев Илья Игоревич
Задорожный Александр Геннадьевич

Новосибирск
2020

1. Цель работы

Разработать программу решения системы нелинейных уравнений (СНУ) методом Ньютона. Провести исследования метода для нескольких систем размерности от 2 до 10.

2. Анализ

Вариант № 1

$m \leq n$. Для нахождения Δx^k , являющегося решением системы (4.2), фиксировать как нулевые те ее $(n-m)$ компонентов с номерами j , для которых

$\max_i \left(\left| \frac{\partial (F_i(x^k))}{\partial x_j} \right| \right)$ минимальны. Производные при формировании матрицы

Якоби считать аналитически.

Вариант № 2

$m \geq n$. Для нахождения Δx^k из системы (4.2) те ее $(m-n)$ уравнений, для которых абсолютные значения $F_i(x^k)$ минимальны, исключаются из системы. При вычислении нормы вектора F^k в процессе подбора параметра β^k учитывать все уравнения системы. Производные при формировании матрицы Якоби вычислять аналитически.

Вариант № 6

В задании 2 при формировании матрицы Якоби вычислять численно.

Теоретическая часть

Пусть дана СНУ в виде:

$$\begin{aligned} F_1(x_1, x_2, \dots, x_n) &= 0; \\ F_2(x_1, x_2, \dots, x_n) &= 0; \\ &\dots \\ F_m(x_1, x_2, \dots, x_n) &= 0. \end{aligned} \quad (4.1)$$

Обозначим через x^k решение, полученное на k -й итерации процесса Ньютона (для первой итерации x^0 – начальное приближение). Запишем исходную систему в виде $F_i(x^k + \Delta x) = 0$, $i = 1 \dots m$, где $\Delta x^k = \bar{x} - x^k$, \bar{x} – искомое решение. Выполним линеаризацию i -го уравнения системы (4.1) с использованием его разложения в ряд Тейлора в окрестности точки x^k :

$$F_i(x) \approx F_i(x^k) + \sum_{j=1}^n \frac{\partial (F_i(x))}{\partial x_j} \bigg|_{x=x^k} \Delta x_j^k, \quad i = 1 \dots m,$$

или в матричном виде:

$$A^k \Delta x^k = -F^k, \quad (4.2)$$

где F^k – значение вектор-функции F при $x=x^k$; A^k – матрица Якоби $\left(A_{ij}^k = \frac{\partial (F_i(x))}{\partial x_j} \Big|_{x=x^k} \right)$.

Это система уравнений, линейных относительно приращений Δx_j^k . Решив эту систему, найдем направление Δx^k поиска решения.

Для поиска следующего приближения x^{k+1} вдоль направления Δx^k организуем итерационный процесс:

$$x_v^{k+1} = x^k + \beta_v^k \Delta x^k,$$

где β_v^k – параметр итерационного процесса поиска x^{k+1} , ($0 < \beta^k < 1$), v – номер итерации поиска оптимального значения β^k . Параметр β^k будем искать следующим образом: сначала (т. е. после нахождения направления Δx^k) β^k принимается равным 1 и вычисляется значение $F_v^k = F(x^k + \beta_v^k \Delta x^k)$; далее, пока норма F_v^k больше, чем норма F^{k-1} , β_v^k уменьшается вдвое.

3. Программа

Файл main.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include "Vector.h"
#include "Newton.h"

using namespace std;
typedef double real;

int main()
{
    Test2_2 test;
    Newton newtonSolver = Newton(test);

    newtonSolver.read_info("tests/info1.txt");
    newtonSolver.solve(1, "results/result_1.txt");

    newtonSolver.read_info("tests/info1.txt");
    newtonSolver.solve(2, "results/result_2.txt");

    newtonSolver.read_info("tests/info1.txt");
    newtonSolver.solve(6, "results/result_6.txt");

    cout << newtonSolver.xk[0] << " " << newtonSolver.xk[1] << endl;

    cout << "b1";
}
}
```

Файл Newton.h

```
#pragma once
#include <stdlib.h>
#include <algorithm>
#include <numeric>
#include "Tests.h"
#include "Vector.h"

typedef double real;
using namespace std;

class Newton
{
public:
    Test2_2 test; // Информация о СНУ

    int max_iter_k; // Максимальное число итераций цикла k
    int max_iter_v; // Максимальное число итераций цикла v
    real eps0 = 10; // Шаг для численного вычисления производной
};
```

```

    real eps1;                // Условие малости для выхода из цикла
v
    real eps2;                // Условие малости для выхода из цикла
k
    real norm_F0;             // Норма вектор-функции F при x=x0

    vector<vector<real>> A;    // Матрица Якоби
    vector<real> Fk;          // Значение вектор-функции F при x=xk
    vector<real> Fvk;         // Значение вектор-функции F при x=xk
                                // на v-той итерации
    vector<real> xk;          // Вектор x на k-той итерации
    vector<real> xk1;         // Вектор x на k+1-той итерации
    vector<real> dxk;         // Вектор направления поиска решения
                                // на k-той итерации

Newton(Test2_2 _test)
{
    test = _test;
    A.resize(test.n_func());

    for (int i = 0; i < test.n_func(); i++)
        A[i].resize(test.n_var());

    Fk.resize(test.n_var());
    Fvk.resize(test.n_var());

    xk.resize(test.n_var());
    xk1.resize(test.n_var());
    dxk.resize(test.n_var());
}

Newton() {};

// Считываем необходимую для решения информацию из файла file_name
void read_info(string file_name)
{
    ifstream fin;
    fin.open(file_name);

    string fict;

    fin >> fict;
    fin >> eps1;

    fin >> fict;
    fin >> eps2;

    fin >> fict;
    fin >> max_iter_k;

    fin >> fict;
    fin >> max_iter_v;

    fin >> fict;
    for(int i = 0; i < test.n_var(); i++)
        fin >> xk[i];

```

```

    get_Fk(xk, Fk);
    norm_F0 = norm(Fk);

    fin.close();
}

// Получаем значение вектор-функции F при x=xk
void get_Fk(vector<real>& xs, vector<real>& res)
{
    res = test.Fk(xs);
}

// Получаем индексы наибольших элементов матрицы Якоби на удаление
vector<int> best_indices(vector<real>& vec, const int& n)
{
    vector<int> indices(vec.size());
    iota(indices.begin(), indices.end(), 0);

    partial_sort(indices.begin(), indices.begin() + n, indices.end(),
        [&vec](int i, int j) {return abs(vec[i]) < abs(vec[j]); });

    return vector<int>(indices.begin(), indices.begin() + n);
}

// Получаем индексы наименьших элементов матрицы Якоби на удаление
vector<int> worst_indices(vector<real>& vec, const int& n)
{
    vector<int> indices(vec.size());
    iota(indices.begin(), indices.end(), 0);

    partial_sort(indices.begin(), indices.begin() + n, indices.end(),
        [&vec](int i, int j) {return abs(vec[i]) > abs(vec[j]); });

    return vector<int>(indices.begin(), indices.begin() + n);
}

// Построение матрицы Якоби согласно 1 варианту
void build_jacobi_1(vector<real>& xs)
{
    A = test.Jacobi(xs);

    get_Fk(xs, Fk);

    for(int i = 0; i < test.n_func(); i++)
        A[i].push_back(-Fk[i]);

    vector<real> max_abs(test.n_var() + 1);

    for(int i = 0; i < test.n_func(); i++)
    {
        max_abs[i] = 0;
        for(int j = 0; j < test.n_var(); j++)
            if(abs(A[i][j]) > abs(max_abs[i]))
                max_abs[i] = abs(A[i][j]);
    }
}

```

```

    vector<int> indexes = worst_indices(max_abs, test.n_func() -
test.n_var());

    sort(indexes.begin(), indexes.end());

    for(int i = indexes.size() - 1; i >= 0; i--)
        A.erase(A.begin() + indexes[i]);
}

// Построение матрицы Якоби согласно 2 варианту
void build_jacobi_2(vector<real>& xs)
{
    A = test.Jacobi(xs);

    get_Fk(xs, Fk);

    for(int i = 0; i < test.n_func(); i++)
        A[i].push_back(- Fk[i]);

    vector<int> indexes = best_indices(Fk, test.n_func() - test.n_var());

    sort(indexes.begin(), indexes.end());

    for(int i = indexes.size() - 1; i >= 0; i--)
        A.erase(A.begin() + indexes[i]);
}

// Построение матрицы Якоби согласно 6 варианту
void build_jacobi_6(vector<real>& xs)
{
    A.resize(test.n_func());

    for(int i = 0; i < test.n_func(); i++)
    {
        A[i].resize(test.n_var());
        for(int j = 0; j < test.n_var(); j++)
        {
            vector<real> eps(test.n_var());
            eps[j] = xs[j] + eps0;

            A[i][j] = test.Fk(eps)[i];

            eps[j] = xs[j] - eps0;

            A[i][j] -= test.Fk(eps)[i];
            A[i][j] /= 2 * eps0;
        }
    }

    get_Fk(xs, Fk);

    for(int i = 0; i < test.n_func(); i++)
        A[i].push_back(-Fk[i]);

    vector<int> indexes = best_indices(Fk, test.n_func() - test.n_var());

```

```

        sort(indexes.begin(), indexes.end());

        for(int i = indexes.size() - 1; i >= 0; i--)
            A.erase(A.begin() + indexes[i]);
    }

    // Прямой ход решателя методом Гаусса
    void forward_gauss(vector<vector<real>>& mat)
    {
        int n = mat.size() + 1;
        int rowWithMaxElem = 0;
        vector<real> rowAdress(n);
        real maxElem = 0;

        for (int j = 0; j < n - 2; j++)
        {
            int rowNumber = j;
            for (int currentCol = j; currentCol < n - 1; currentCol++)
            {
                if (fabs(mat[currentCol][j]) > maxElem)
                {
                    maxElem = fabs(mat[currentCol][j]);
                    rowWithMaxElem = currentCol;
                }
            }

            maxElem = 0;
            rowAdress = mat[rowWithMaxElem];
            mat[rowWithMaxElem] = mat[rowNumber];
            mat[rowNumber] = rowAdress;

            for (int k = 1 + j; k < n - 1; k++)
            {
                real factor = mat[k][rowNumber] / mat[rowNumber][rowNumber];
                if (factor != 0)
                {
                    for (int i = rowNumber; i < n; i++)
                    {
                        real tmp = mat[rowNumber][i] * factor;
                        mat[k][i] -= tmp;
                    }
                }
            }
        }
    }

    // Обратный ход решателя методом Гаусса
    void backward_gauss(vector<vector<real>>& mat, vector<real>& res)
    {
        int n = mat.size() + 1;
        res.resize(n - 1);

        for (int i = n - 2; i >= 0; i--)
        {
            real sum = 0;

```



```

        for (int j = i + 1; j < n - 1; j++)
        {
            sum += res[j] * mat[i][j];
        }

        res[i] = (mat[i][n - 1] - sum) / mat[i][i];
    }
}

// Функция решения СЛУ
void solve(int var, string file_name)
{
    ofstream fout;
    fout.open(file_name);
    fout << "k\tbeta\tx\tty\tnorm" << endl;

    for(int k = 0; k < max_iter_k && norm(Fk) / norm_F0 > eps2; k++)
    {
        switch(var)
        {
            case 1:
            {
                build_jacobi_1(xk);
                break;
            }
            case 2:
            {
                build_jacobi_2(xk);
                break;
            }
            case 6:
            {
                build_jacobi_6(xk);
                break;
            }
        }

        forward_gauss(A);
        backward_gauss(A, dxk);

        for(int i = 0; i < test.n_var(); i++)
            if(abs(dxk[i]) == INFINITY)
            {
                cout << "Cant solve!" << endl;
                return;
            }

        real beta = 1;

        for(int v = 0; v < max_iter_v; v++)
        {
            xk1 = xk + beta * dxk;
            get_Fk(xk1, Fvk);

            if(norm(Fvk) < norm(Fk) || beta < eps1)
                break;
        }
    }
}

```

```

        else
            beta /= 2;
    }
    xk = xk1;

    // Блок вывода информации о текущей итерации в консоль (для двумер-
ного случая)

    fout << k << "\t" << beta << "\t" << xk[0] << "\t" << xk[1] << "\t"
<< norm(Fk) << endl;
    }
    fout.close();
}
};

```

Файл Tests.h

```

#pragma once
#include <math.h>

using namespace std;
typedef double real;
real alpha = 1000;

// Квадрат числа
real sq(real val)
{
    return val * val;
}

// Две окружности не пересекаются
class Test1_1
{
public:
    virtual int n_func() { return 2; };
    virtual int n_var() { return 2; };

    Test1_1()
    {

    }

    real F1(vector<real>& xs) { return sq(xs[0] - 2) + sq(xs[1] - 2) - 4; }
    real F2(vector<real>& xs) { return sq(xs[0] + 2) + sq(xs[1] + 2) - 4; }
    real F1dx(vector<real>& xs) { return 2 * xs[0] - 4; }
    real F1dy(vector<real>& xs) { return 2 * xs[1] - 4; }
    real F2dx(vector<real>& xs) { return 2 * xs[0] + 4; }
    real F2dy(vector<real>& xs) { return 2 * xs[1] + 4; }
}

```

```

    vector<vector<real>> Jacobi(vector<real>& xs) { return { { F1dx(xs),
F1dy(xs) }, { F2dx(xs), F2dy(xs) } }; }

    vector<real> Fk(vector<real>& xs) { return { F1(xs), F2(xs) }; }
};

// Две окружности пересекаются в 1 точке
class Test1_2
{
public:
    virtual int n_func() { return 2; };
    virtual int n_var() { return 2; };

    Test1_2()
    {

    }

    real F1(vector<real>& xs) { return sq(xs[0] - 2) + sq(xs[1] - 4) - 4; }
    real F2(vector<real>& xs) { return sq(xs[0] + 2) + sq(xs[1] - 4) - 4; }
    real F1dx(vector<real>& xs) { return 2 * xs[0] - 4; }
    real F1dy(vector<real>& xs) { return 2 * xs[1] - 8; }
    real F2dx(vector<real>& xs) { return 2 * xs[0] + 4; }
    real F2dy(vector<real>& xs) { return 2 * xs[1] - 8; }

    vector<vector<real>> Jacobi(vector<real>& xs) { return { { F1dx(xs),
F1dy(xs) }, { F2dx(xs), F2dy(xs) } }; }

    vector<real> Fk(vector<real>& xs) { return { F1(xs), F2(xs) }; }
};

// Две окружности пересекаются в 2 точках
class Test1_3
{
public:
    virtual int n_func() { return 2; };
    virtual int n_var() { return 2; };

    Test1_3()
    {

    }

    real F1(vector<real>& xs) { return sq(xs[0] - 2) + sq(xs[1] - 4) - 9; }
    real F2(vector<real>& xs) { return sq(xs[0] + 2) + sq(xs[1] - 4) - 9; }
    real F1dx(vector<real>& xs) { return 2 * xs[0] - 4; }
    real F1dy(vector<real>& xs) { return 2 * xs[1] - 8; }

```

```

    real F2dx(vector<real>& xs) { return 2 * xs[0] + 4; }

    real F2dy(vector<real>& xs) { return 2 * xs[1] - 8; }

    vector<vector<real>> Jacobi(vector<real>& xs) { return { { F1dx(xs),
F1dy(xs) }, { F2dx(xs), F2dy(xs) } }; }

    vector<real> Fk(vector<real>& xs) { return { F1(xs), F2(xs) }; }
};

// Две окружности и прямая не пересекаются
class Test2_1
{
public:
    virtual int n_func() { return 3; };
    virtual int n_var() { return 2; };

    Test2_1()
    {

    }

    real F1(vector<real>& xs) { return sq(xs[0] - 2) + sq(xs[1] - 2) - 4; }
    real F2(vector<real>& xs) { return sq(xs[0] + 2) + sq(xs[1] + 2) - 4; }
    real F3(vector<real>& xs) { return xs[0] + xs[1]; }
    real F1dx(vector<real>& xs) { return 2 * xs[0] - 4; }
    real F1dy(vector<real>& xs) { return 2 * xs[1] - 4; }
    real F2dx(vector<real>& xs) { return 2 * xs[0] + 4; }
    real F2dy(vector<real>& xs) { return 2 * xs[1] + 4; }
    real F3dx(vector<real>& xs) { return 1; }
    real F3dy(vector<real>& xs) { return 1; }

    vector<vector<real>> Jacobi(vector<real>& xs) { return { { F1dx(xs),
F1dy(xs) }, { F2dx(xs), F2dy(xs) }, { F3dx(xs), F3dy(xs) } }; }

    vector<real> Fk(vector<real>& xs) { return { F1(xs), F2(xs), F3(xs) }; }
};

// Две окружности и прямая пересекаются
class Test2_2
{
public:
    virtual int n_func() { return 2; };
    virtual int n_var() { return 2; };

    Test2_2()
    {

```

```

}

real F1(vector<real>& xs) { return sin(xs[0]) - xs[1]; }

real F2(vector<real>& xs) { return 4*xs[0] - xs[1] - 20; }

real F1dx(vector<real>& xs) { return cos(xs[0]); }

real F1dy(vector<real>& xs) { return -1; }

real F2dx(vector<real>& xs) { return 4; }

real F2dy(vector<real>& xs) { return -1; }

vector<vector<real>> Jacobi(vector<real>& xs) { return { { F1dx(xs),
F1dy(xs) }, { F2dx(xs), F2dy(xs) } }; }

vector<real> Fk(vector<real>& xs) { return { F1(xs), F2(xs) }; }
};

```

Файл Vector.h

```

#pragma once
#include <vector>
#include <iomanip>
#include <fstream>
using namespace std;

typedef double real;

// Умножение вектора на число
vector<real> operator * (real val, const vector<real>& vec)
{
    vector<real> res(vec.size());

    for (size_t i = 0; i < vec.size(); ++i)
        res[i] = val * vec[i];
    return res;
}

// Сложение векторов
vector<real> operator + (const vector<real>& vec1, const vector<real>& vec2)
{
    if (vec1.size() != vec2.size())
        throw("a.size() != b.size()");

    vector<real> res(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
        res[i] = vec1[i] + vec2[i];
    return res;
}

```

```

// Вычитание векторов
vector<real> operator - (const vector<real>& vec1, const vector<real>& vec2)
{
    if (vec1.size() != vec2.size())
        throw("a.size() != b.size()");

    vector<real> res(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
        res[i] = vec1[i] - vec2[i];
    return res;
}

// Скалярное произведение векторов
real operator *(const vector<real>& vec1, const vector<real>& vec2)
{
    if (vec1.size() != vec2.size())
        throw("vec1.size() != vec2.size()");

    int n = vec1.size();
    real res = 0;

    for (int i = 0; i < n; i++)
        res += vec1[i] * vec2[i];

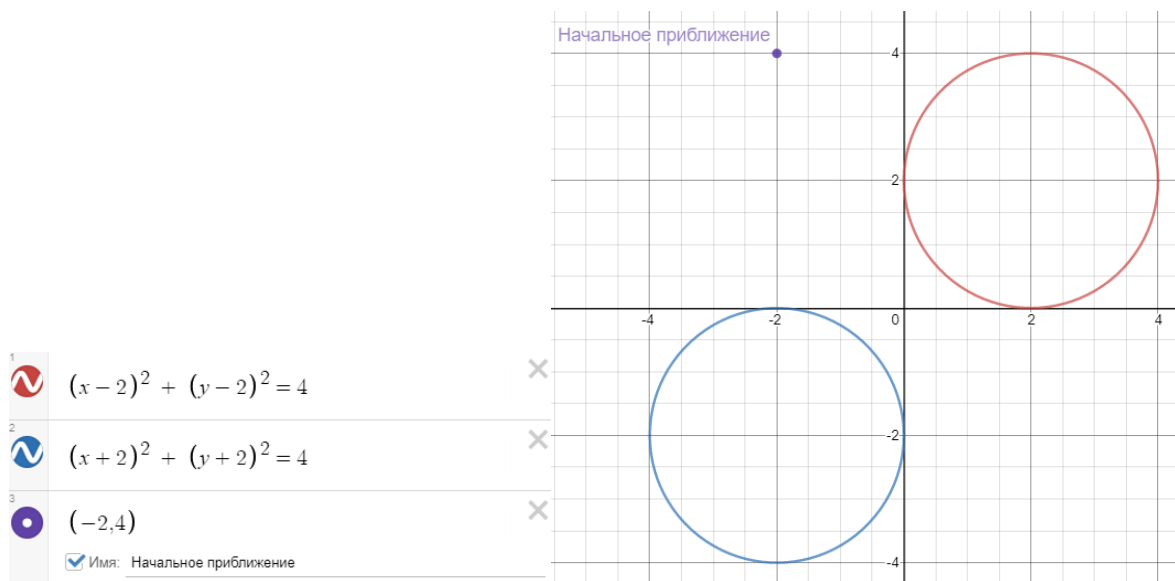
    return res;
}

// Норма вектора
real norm(const vector<real>& vec)
{
    return sqrt(vec * vec);
}

```

4. Тестирование и исследования

Окружности не пересекаются 1.



1 вариант

k	beta	x	y	norm
0	1	-1.33333	1.33333	35.7771
1	1	0.0833333	-0.0833333	10.6852
2	0.0078125	-0.0107422	0.0107422	5.6765
3	0.00012207	0.000622105	-0.000622105	5.65718
.
98	2.27374e-13	-2.31288e-07	2.31288e-07	5.65685
99	5.68434e-14	1.44817e-08	-1.44817e-08	5.65685




2 вариант

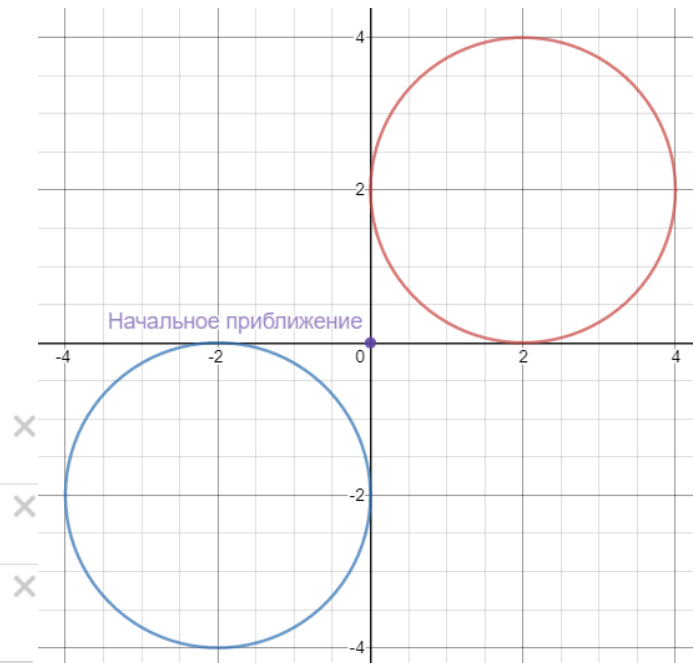
k	beta	x	y	norm
0	1	-1.33333	1.33333	35.7771
1	1	0.0833333	-0.0833333	10.6852
2	0.0078125	-0.0107422	0.0107422	5.6765
3	0.00012207	0.000622105	-0.000622105	5.65718
.
98	2.27374e-13	-2.31288e-07	2.31288e-07	5.65685
99	5.68434e-14	1.44817e-08	-1.44817e-08	5.65685

6 вариант

k	beta	x	y	norm
0	1	-1.33333	1.33333	35.7771
1	1	0.0833333	-0.0833333	10.6852
2	0.0078125	-0.0107422	0.0107422	5.6765
3	0.00012207	0.000622105	-0.000622105	5.65718
.
98	9.09495e-13	-1.83996e-07	1.83996e-07	5.65685
99	5.68434e-14	1.24942e-07	-1.24941e-07	5.65685

Окружности не пересекаются 2.

1		$(x - 2)^2 + (y - 2)^2 = 4$
2		$(x + 2)^2 + (y + 2)^2 = 4$
3		$(0, 0)$
<input checked="" type="checkbox"/> Имя: Начальное приближение		



1 вариант

Cant solve!

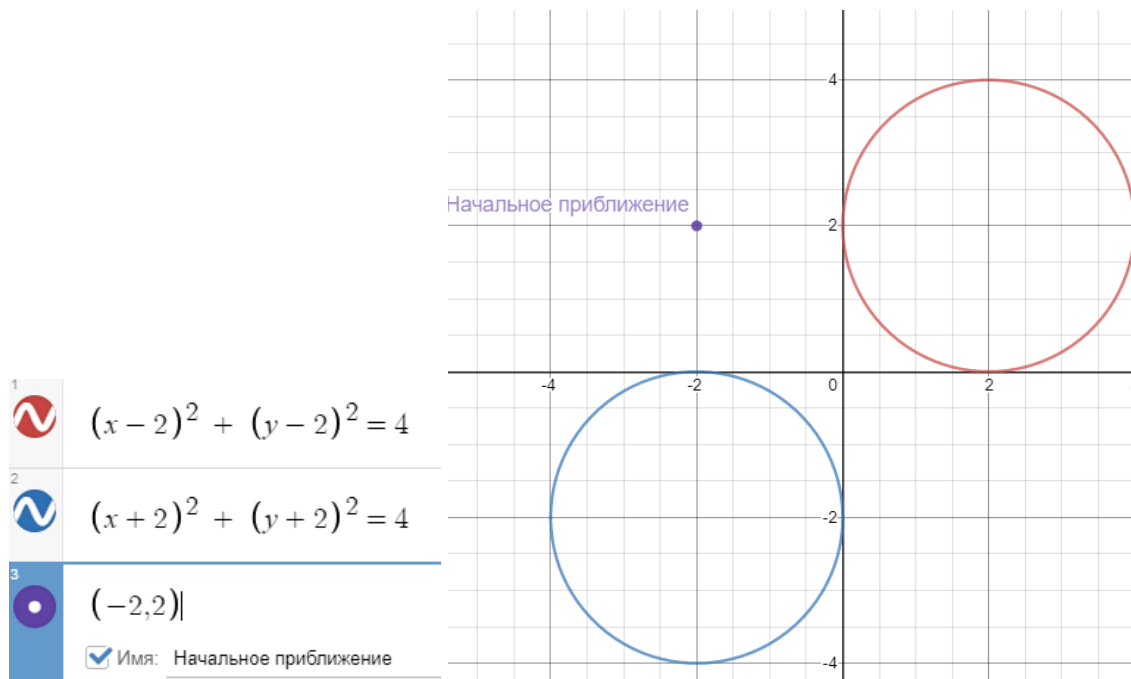
2 вариант

Cant solve!

6 вариант

Cant solve!

Окружности не пересекаются 3.



1 вариант

k	beta	x	y	norm
0	1	-0.5	0.5	16.9706
1	0.25	0.0625	-0.0625	6.36396
2	0.00390625	-0.00012207	0.00012207	5.6679
3	1.49012e-08	9.09189e-13	-9.09189e-13	5.65685
.
98	9.09495e-13	-1.04023e-07	1.04023e-07	5.65685
99	5.68434e-14	4.42426e-07	-4.42426e-07	5.65685




2 вариант

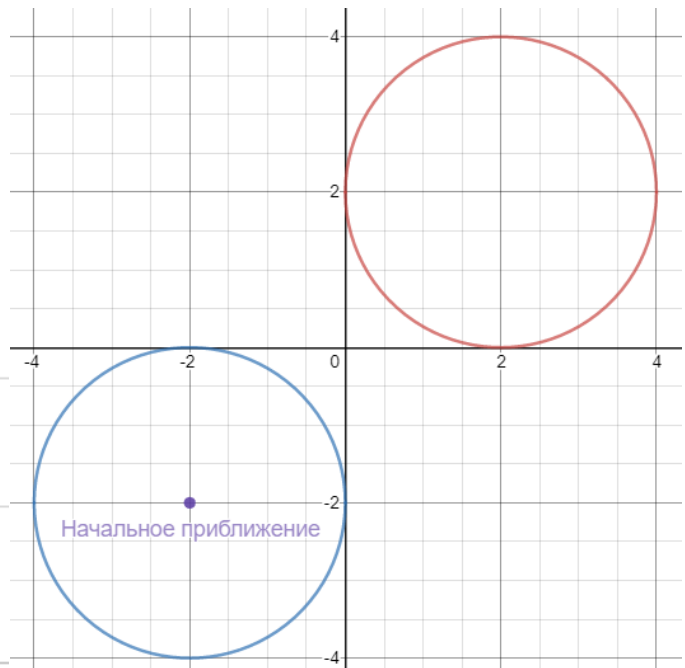
k	beta	x	y	norm
0	1	-0.5	0.5	16.9706
1	0.25	0.0625	-0.0625	6.36396
2	0.00390625	-0.00012207	0.00012207	5.6679
3	1.49012e-08	9.09189e-13	-9.09189e-13	5.65685
.
98	9.09495e-13	-1.04023e-07	1.04023e-07	5.65685
99	5.68434e-14	4.42426e-07	-4.42426e-07	5.65685

6 вариант

k	beta	x	y	norm
0	1	-0.5	0.5	16.9706
1	0.25	0.0625	-0.0625	6.36396
2	0.00390625	-0.00012207	0.00012207	5.6679
3	1.49012e-08	5.29974e-12	-5.2892e-12	5.65685
.
98	4.54747e-13	-1.27425e-07	1.27425e-07	5.65685
99	5.68434e-14	3.18669e-07	-3.18669e-07	5.65685

Окружности не пересекаются 4.

1		$(x - 2)^2 + (y - 2)^2 = 4$
2		$(x + 2)^2 + (y + 2)^2 = 4$
3		$(-2, -2)$
<input checked="" type="checkbox"/> Имя: Начальное приближение		



1 вариант

Cant solve!

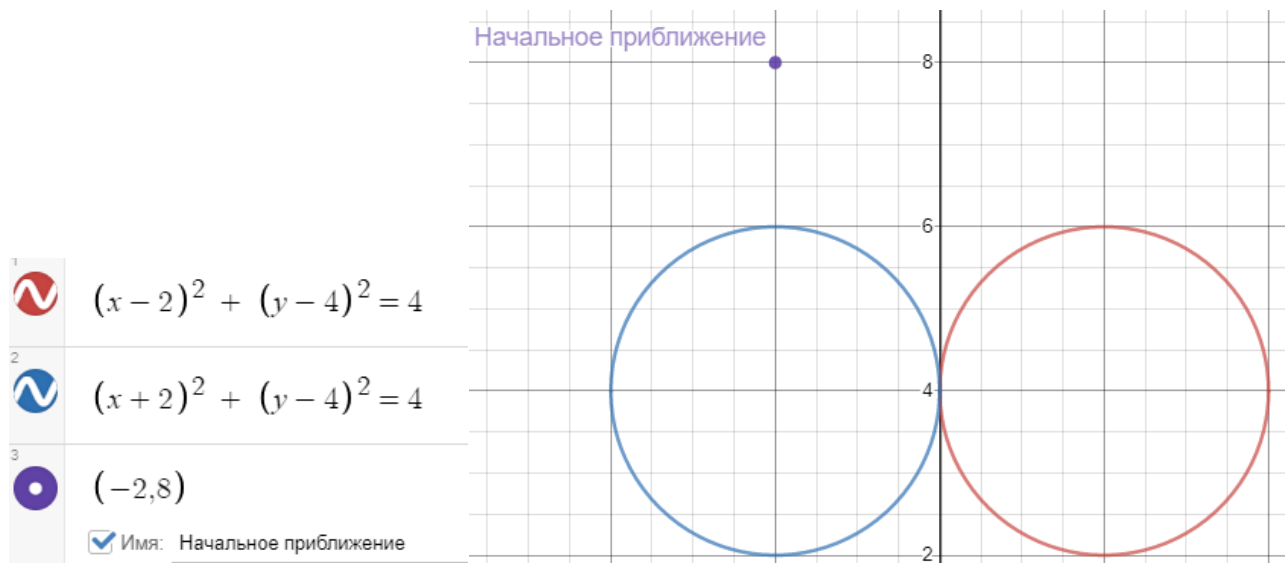
2 вариант

Cant solve!

6 вариант

Cant solve!

Окружности пересекаются в одной точке 1.



1 вариант

k	beta	x	y	norm
0	1	0	6.5	30.4631
1	1	0	5.25	8.83883
2	1	0	4.625	2.20971
3	1	0	4.3125	0.552427
.
21	1	0	4	8.03887e-12
22	1	0	4	2.00972e-12

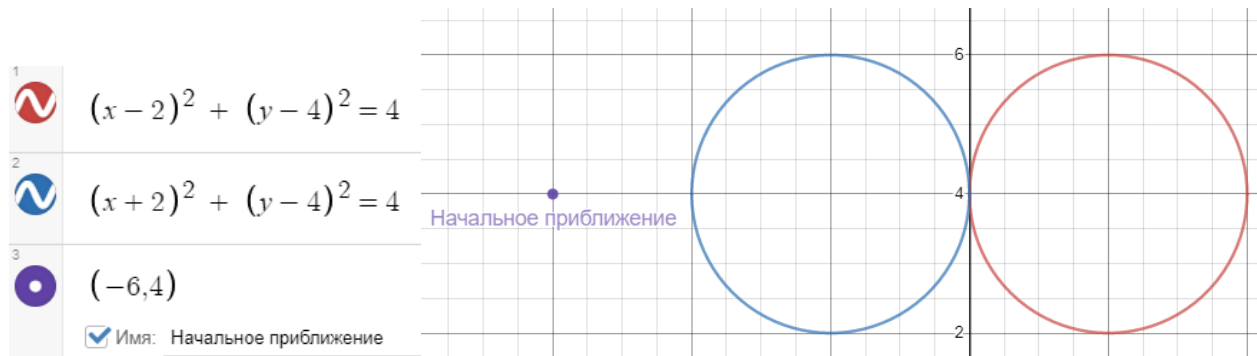
2 вариант

k	beta	x	y	norm
0	1	0	6.5	30.4631
1	1	0	5.25	8.83883
2	1	0	4.625	2.20971
3	1	0	4.3125	0.552427
.
21	1	0	4	8.03887e-12
22	1	0	4	2.00972e-12

6 вариант

k	beta	x	y	norm
0	1	1.10845e-12	6.5	30.4631
1	1	3.70171e-25	5.25	8.83883
2	1	3.70171e-25	4.625	2.20971
3	1	-1.38778e-17	4.3125	0.552427
.
21	1	-1.47451e-17	4	8.03887e-12
22	1	-1.47451e-17	4	2.00972e-12

Окружности пересекаются в одной точке 2.



1 вариант

Cant solve!

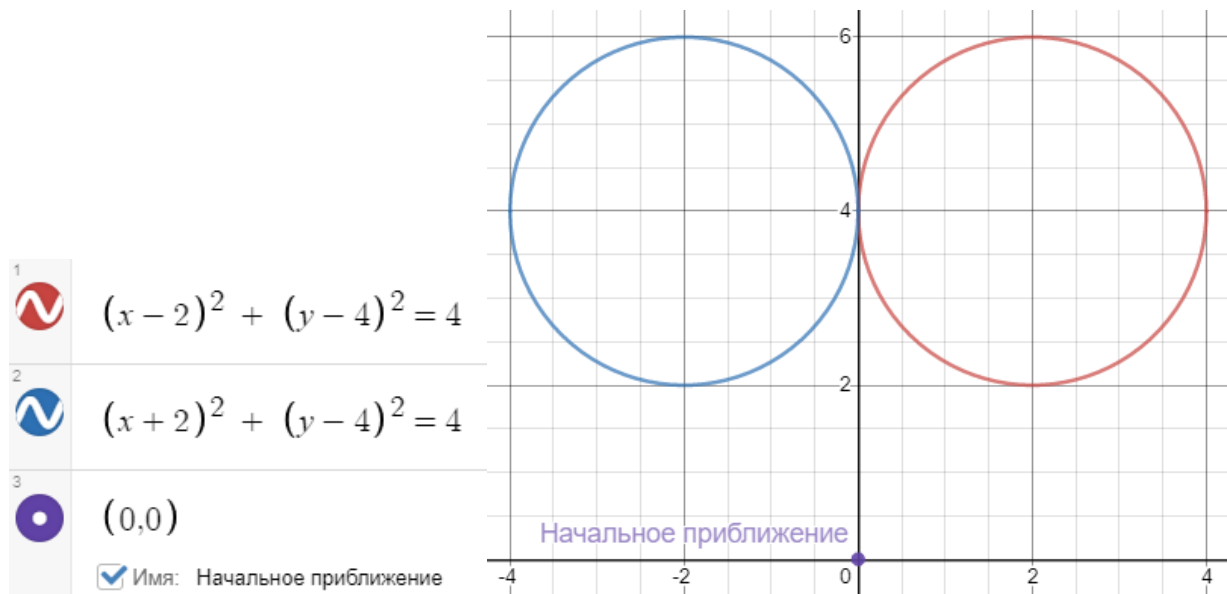
2 вариант

Cant solve!

6 вариант

Cant solve!

Окружности пересекаются в одной точке 3.



1 вариант

k	beta	x	y	norm
0	1	0	2	22.6274
1	1	0	3	5.65685
2	1	0	3.5	1.41421
3	1	0	3.75	0.353553
.
21	1	0	4	5.14488e-12
22	1	0	4	1.28622e-12

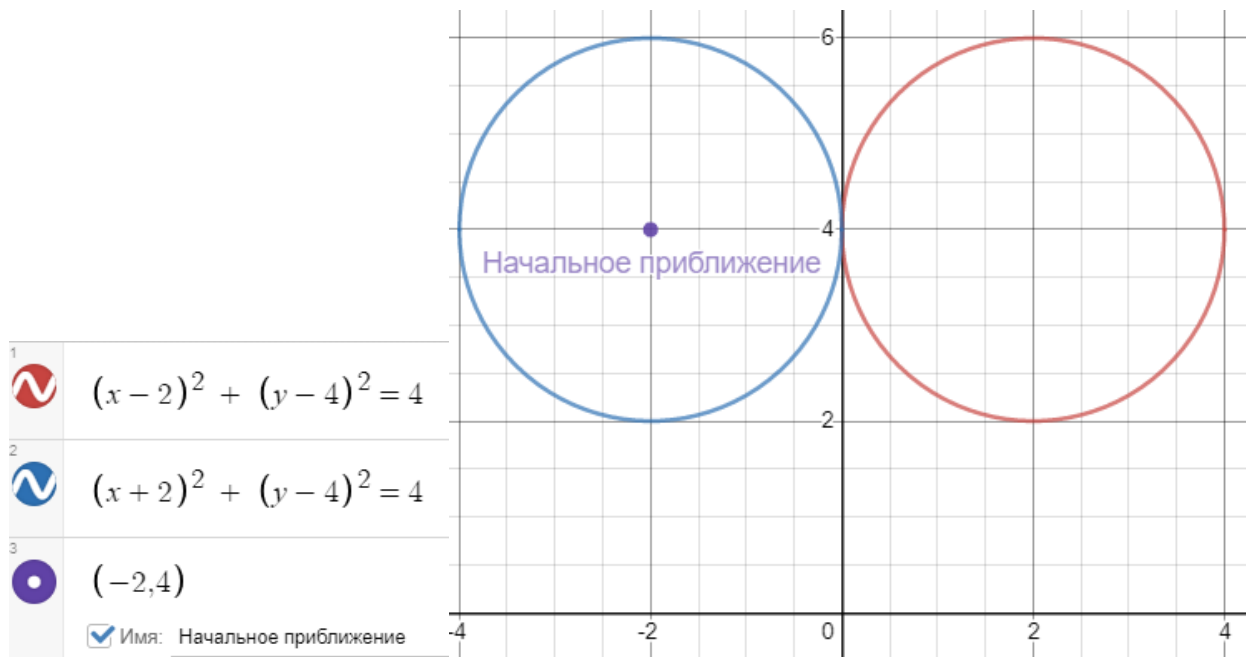
2 вариант

k	beta	x	y	norm
0	1	0	2	22.6274
1	1	0	3	5.65685
2	1	0	3.5	1.41421
3	1	0	3.75	0.353553
.
21	1	0	4	5.14488e-12
22	1	0	4	1.28622e-12

6 вариант

k	beta	x	y	norm
0	1	0	2	22.6274
1	1	0	3	5.65685
2	1	0	3.5	1.41421
3	1	0	3.75	0.353553
.
21	1	0	4	5.14488e-12
22	1	0	4	1.28622e-12

Окружности пересекаются в одной точке 4.



1 вариант

Cant solve!

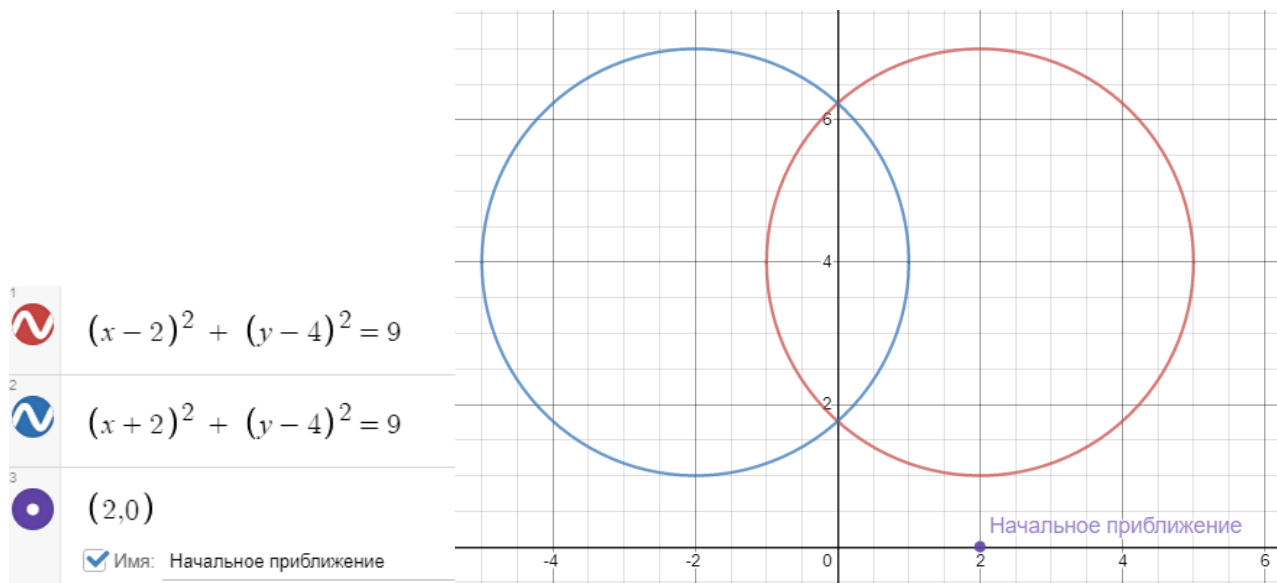
2 вариант

Cant solve!

6 вариант

Cant solve!

Окружности пересекаются в двух точках 1.



1 вариант

k	beta	x	y	norm
0	1	0	0.875	24.0416
1	1	0	1.6375	6.73961
2	1	0	1.76055	0.822233
3	1	0	1.76393	0.0214127
4	1	0	1.76393	1.61616e-05
5	1	0	1.76393	9.23466e-12
6	5.68434e-14	0	1.76393	0

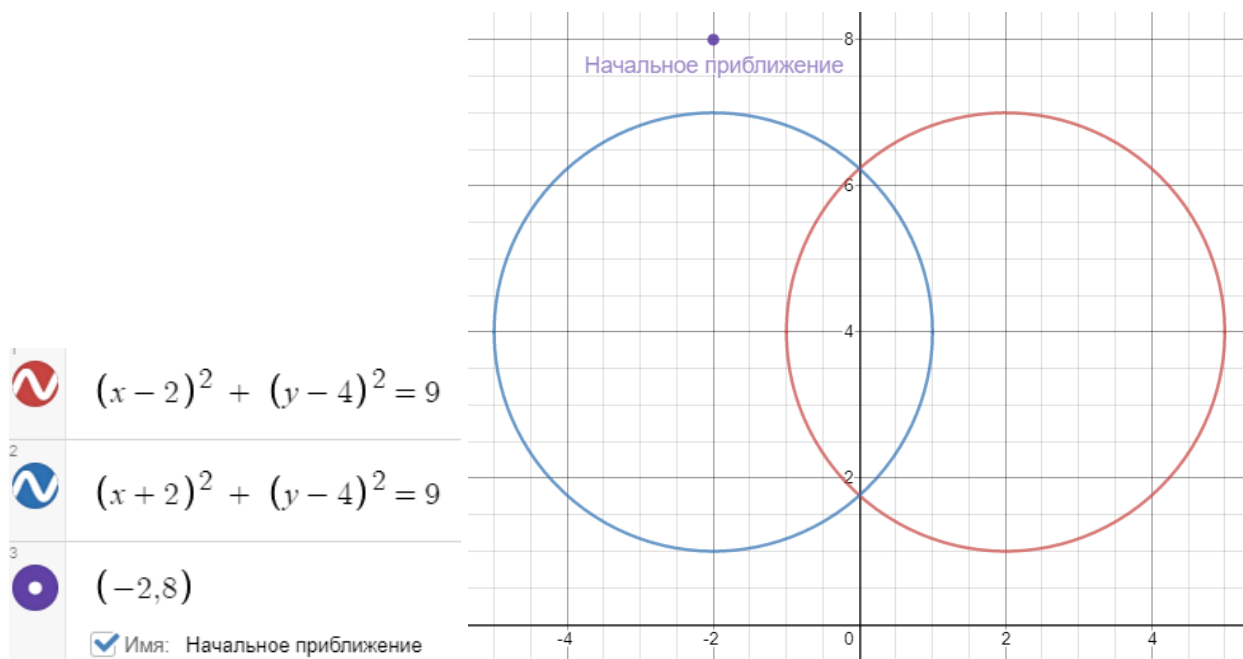
2 вариант

k	beta	x	y	norm
0	1	0	0.875	24.0416
1	1	0	1.6375	6.73961
2	1	0	1.76055	0.822233
3	1	0	1.76393	0.0214127
4	1	0	1.76393	1.61616e-05
5	1	0	1.76393	9.23466e-12
6	5.68434e-14	0	1.76393	0

6 вариант

k	beta	x	y	norm
0	1	-1.10845e-12	0.875	24.0416
1	1	-3.70171e-25	1.6375	6.73961
2	1	-3.70171e-25	1.76055	0.822233
3	1	-3.70171e-25	1.76393	0.0214127
4	1	-3.70171e-25	1.76393	1.61616e-05
5	1	-3.70171e-25	1.76393	9.23466e-12
6	5.68434e-14	-3.70171e-25	1.76393	0

Окружности пересекаются в двух точках 2.



1 вариант

k	beta	x	y	norm
0	1	0	7.125	24.0416
1	1	0	6.3625	6.73961
2	1	0	6.23945	0.822233
3	1	0	6.23607	0.0214127
4	1	0	6.23607	1.61616e-05
5	1	0	6.23607	9.23466e-12
6	5.68434e-14	0	6.23607	0

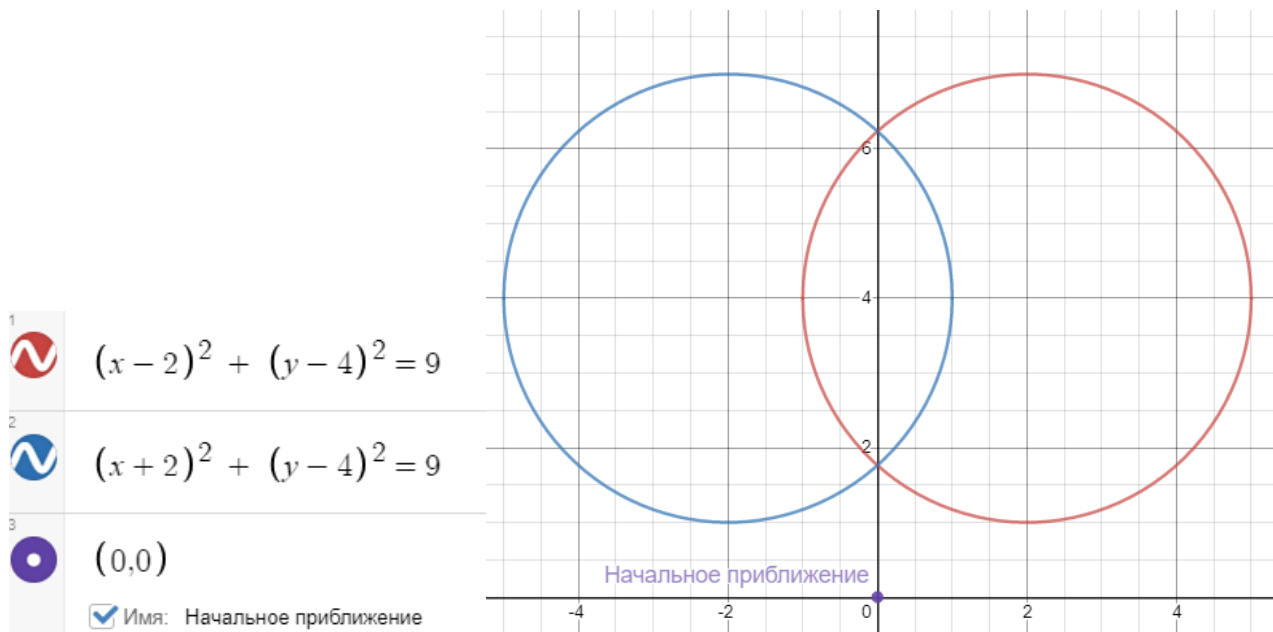
2 вариант

k	beta	x	y	norm
0	1	0	7.125	24.0416
1	1	0	6.3625	6.73961
2	1	0	6.23945	0.822233
3	1	0	6.23607	0.0214127
4	1	0	6.23607	1.61616e-05
5	1	0	6.23607	9.23466e-12
6	5.68434e-14	0	6.23607	0

6 вариант

k	beta	x	y	norm
0	1	1.10845e-12	7.125	24.0416
1	1	3.70171e-25	6.3625	6.73961
2	1	3.70171e-25	6.23945	0.822233
3	1	3.70171e-25	6.23607	0.0214127
4	1	3.70171e-25	6.23607	1.61616e-05
5	1	3.70171e-25	6.23607	9.23466e-12
6	5.68434e-14	3.70171e-25	6.23607	0

Окружности пересекаются в двух точках 3.



1 вариант

k	beta	x	y	norm
0	1	0	1.375	15.5563
1	1	0	1.73512	2.67375
2	1	0	1.76375	0.183403
3	1	-2.71051e-20	1.76393	0.00115917
4	1	-2.71034e-20	1.76393	4.74986e-08
5	5.68434e-14	-2.71034e-20	1.76393	0

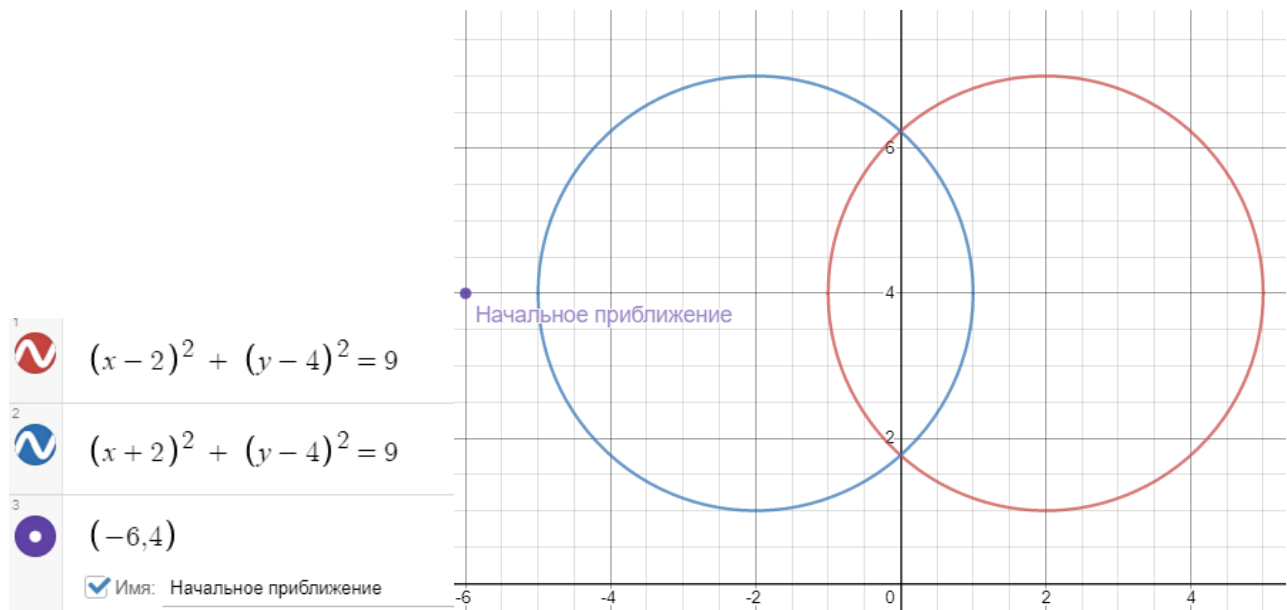
2 вариант

k	beta	x	y	norm
0	1	0	1.375	15.5563
1	1	0	1.73512	2.67375
2	1	0	1.76375	0.183403
3	1	-2.71051e-20	1.76393	0.00115917
4	1	-2.71034e-20	1.76393	4.74986e-08
5	5.68434e-14	-2.71034e-20	1.76393	0

6 вариант

k	beta	x	y	norm
0	1	0	1.375	15.5563
1	1	0	1.73512	2.67375
2	1	0	1.76375	0.183403
3	1	0	1.76393	0.00115917
4	1	1.65436e-24	1.76393	4.74986e-08
5	5.68434e-14	1.65436e-24	1.76393	0

Окружности пересекаются в двух точках 4.



1 вариант

Cant solve!

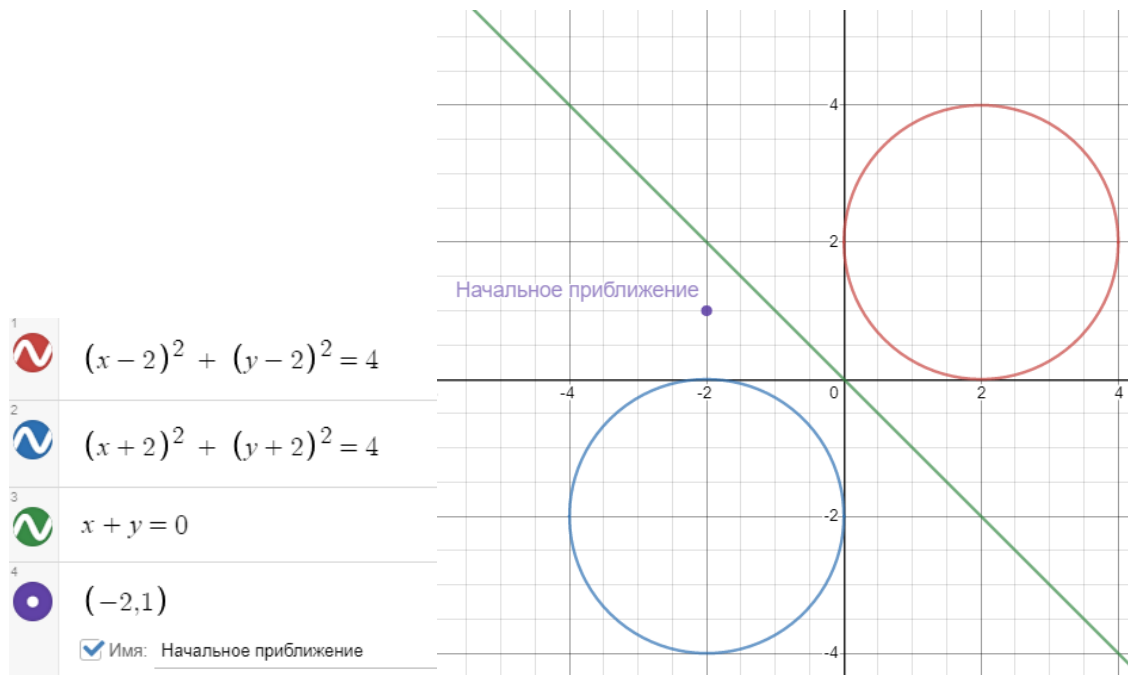
2 вариант

Cant solve!

6 вариант

Cant solve!

Две окружности и прямая 1.



1 вариант

k	beta	x	y	norm
0	1	-0.166667	0.166667	13.9642
1	0.03125	0.0234375	-0.0234375	5.73542
2	0.000976563	-0.0182406	0.0182406	5.65841
3	0.000488281	0.00853275	-0.00853275	5.6578
.
98	2.27374e-13	-1.28835e-08	1.28835e-08	5.65685
99	5.68434e-14	4.39921e-06	-4.39921e-06	5.65685





2 вариант

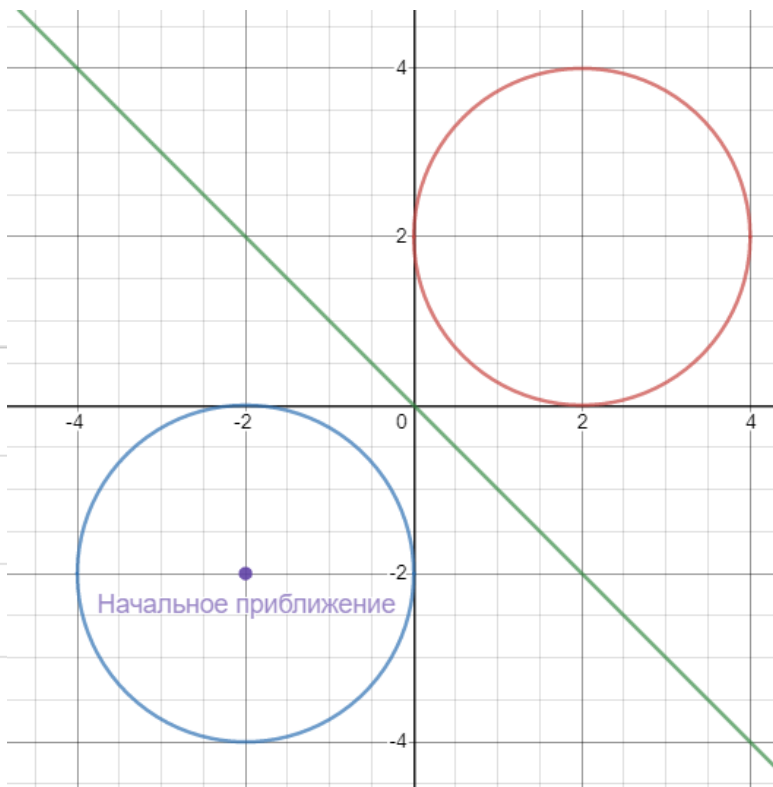
k	beta	x	y	norm
0	1	-0.166667	0.166667	13.9642
1	0.03125	0.0234375	-0.0234375	5.73542
2	0.000976563	-0.0182406	0.0182406	5.65841
3	0.000488281	0.00853275	-0.00853275	5.6578
.
98	1.13687e-13	-5.40388e-08	5.40388e-08	5.65685
99	5.68434e-14	9.97862e-07	-9.97862e-07	5.65685

6 вариант

k	beta	x	y	norm
0	1	-0.166667	0.166667	13.9642
1	0.03125	0.0234375	-0.0234375	5.73542
2	0.000976563	-0.0182406	0.0182406	5.65841
3	0.000488281	0.00853275	-0.00853275	5.6578
.
98	5.68434e-14	-1.02281e-06	1.02281e-06	5.65685
99	1.81899e-12	7.55624e-07	-7.55624e-07	5.65685

Две окружности и прямая 2.

1		$(x - 2)^2 + (y - 2)^2 = 4$
2		$(x + 2)^2 + (y + 2)^2 = 4$
3		$x + y = 0$
4		$(-2, -2)$
<input checked="" type="checkbox"/> Имя: Начальное приближение		



1 вариант

Cant solve!

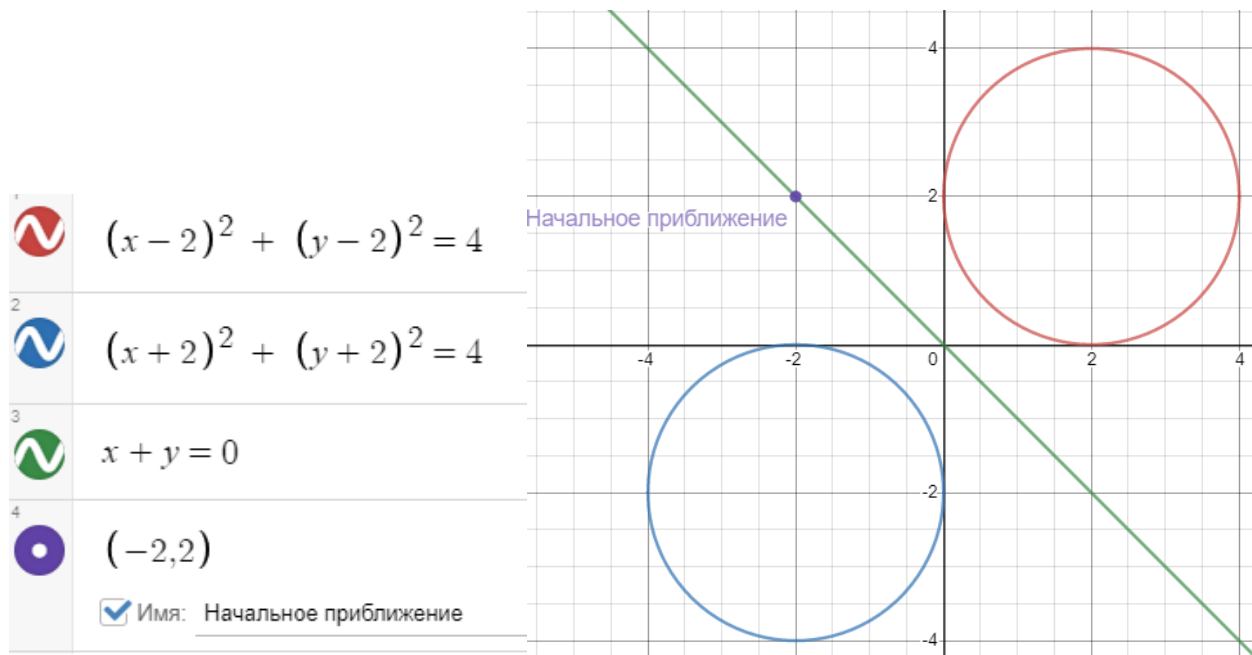
2 вариант

Cant solve!

6 вариант

Cant solve!

Две окружности и прямая 3.



1 вариант

k	beta	x	y	norm
0	1	-0.166667	0.166667	5.73542
1	0.03125	0.0234375	-0.0234375	5.65841
2	0.000976563	-0.0182406	0.0182406	5.6578
3	0.000488281	0.00853275	-0.00853275	5.65706
.
98	2.27374e-13	-1.28835e-08	1.28835e-08	5.65685
99	5.68434e-14	4.39921e-06	-4.39921e-06	5.65685

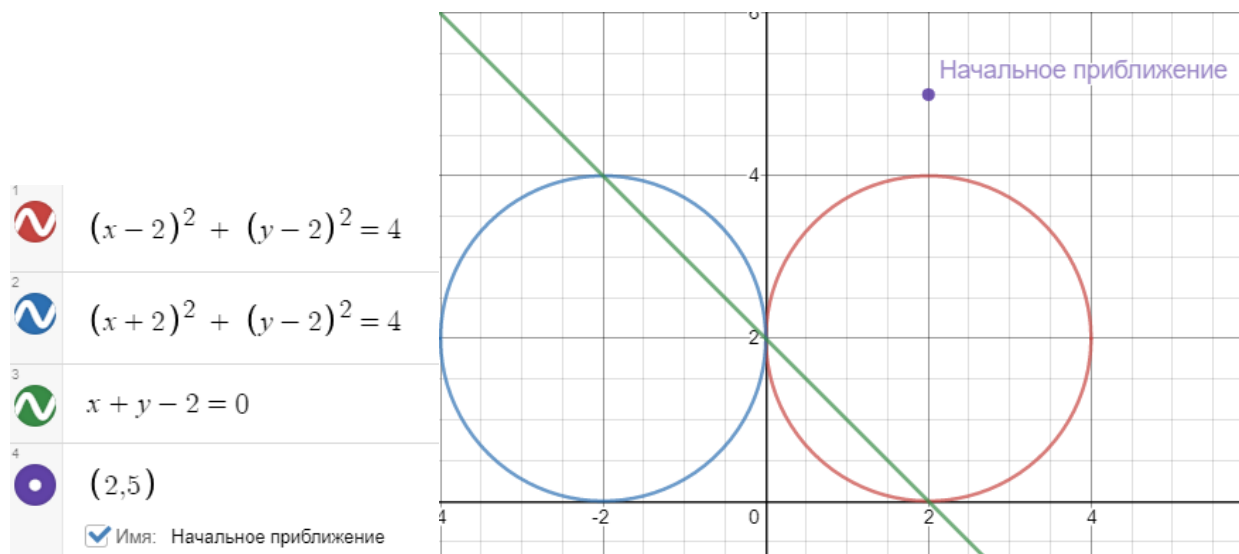
2 вариант

k	beta	x	y	norm
0	1	-0.166667	0.166667	5.73542
1	0.03125	0.0234375	-0.0234375	5.65841
2	0.000976563	-0.0182406	0.0182406	5.6578
3	0.000488281	0.00853275	-0.00853275	5.65706
.
98	1.13687e-13	-5.40388e-08	5.40388e-08	5.65685
99	5.68434e-14	9.97862e-07	-9.97862e-07	5.65685

6 вариант

k	beta	x	y	norm
0	1	-0.166667	0.166667	5.73542
1	0.03125	0.0234375	-0.0234375	5.65841
2	0.000976563	-0.0182406	0.0182406	5.6578
3	0.000488281	0.00853275	-0.00853275	5.65706
.
98	5.68434e-14	-1.02281e-06	1.02281e-06	5.65685
99	1.81899e-12	7.55624e-07	-7.55624e-07	5.65685

Две окружности и прямая 4.



1 вариант

k	beta	x	y	norm
0	1	-2.16667	4.16667	22.1585
1	1	-2.0119	4.0119	18.07
2	1	-2.00007	4.00007	16.1432
3	1	-2	4	16.0008
4	1	-2	4	16
5	5.68434e-14	-2	4	16

2 вариант

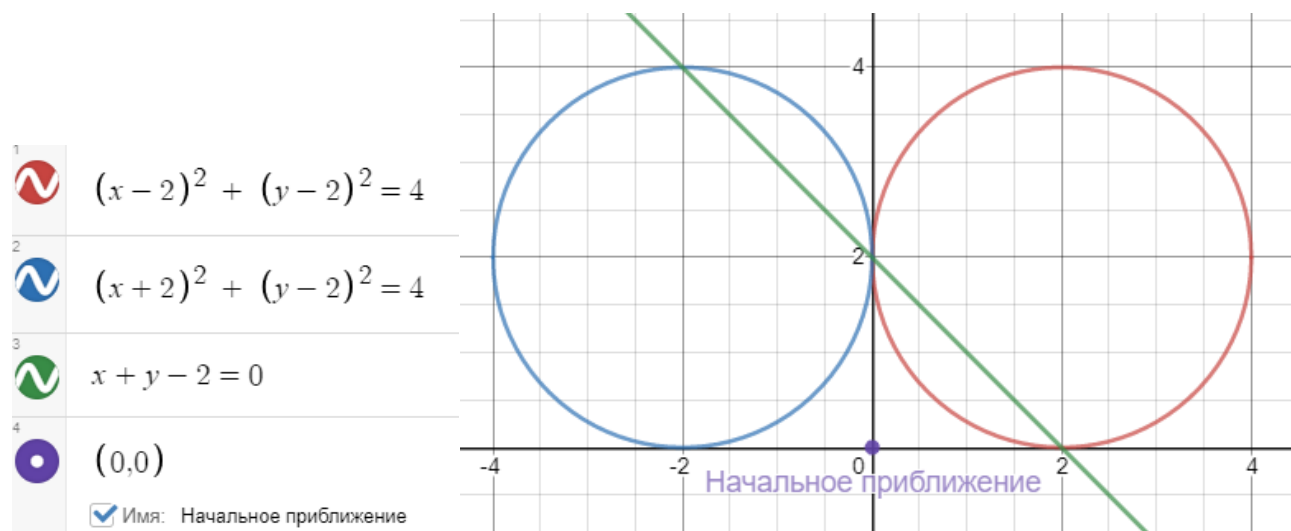
k	beta	x	y	norm
0	0.125	2.5625	3.8125	22.1585
1	1	1.79119	0.208807	20.576
2	1	2.22045e-16	0.208807	13.6021
3	1	3.33067e-16	1.1044	4.87808
4	1	0.138502	1.8615	1.44527
5	1	2.77556e-17	1.8615	0.785363
6	1	0.00448512	1.99551	0.141134
7	1	1.82146e-17	1.99551	0.0253717
8	1	5.01782e-06	1.99999	0.00448521
9	1	5.08321e-17	1.99999	2.83851e-05
10	1	6.29475e-12	2	5.01786e-06
11	5.68434e-14	6.29475e-12	2	3.56091e-11

6 вариант

k	beta	x	y	norm
0	0.125	2.5625	3.8125	22.1585
1	1	1.79119	0.208807	20.576
2	1	-4.95382e-13	0.208807	13.6021
3	1	1.11022e-16	1.1044	4.87808
4	1	0.138502	1.8615	1.44527
5	1	5.55112e-17	1.8615	0.785363
6	1	0.00448512	1.99551	0.141134

7	1	-5.03937e-16	1.99551	0.0253717
8	1	-5.04038e-06	2.00001	0.00448521
9	1	7.87571e-18	2.00001	2.85127e-05
10	1	6.35128e-12	2	5.04032e-06
11	1	-8.8273e-17	2	3.59287e-11
12	1	-8.82729e-17	2	1.28162e-11
13	5.68434e-14	-8.82729e-17	2	0

Две окружности и прямая 5.



1 вариант

k	beta	x	y	norm
0	1	0.5	1.5	6
1	1	-0.25	2.25	2.91548
2	1	0.0416667	1.95833	1.42522
3	1	-0.000905797	2.00091	0.235753
4	1	4.10606e-07	2	0.00512396
5	1	-8.41259e-14	2	2.32274e-06
6	1	2.89769e-17	2	4.75425e-13

2 вариант

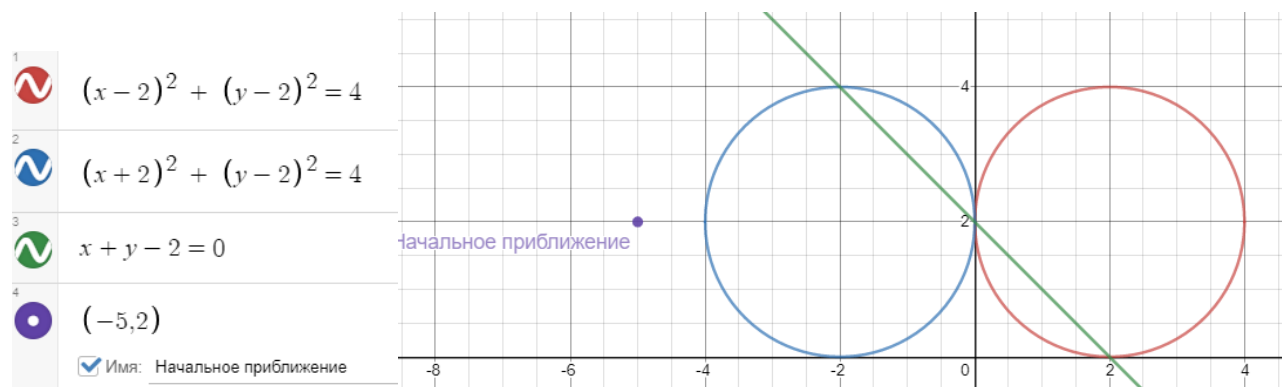
k	beta	x	y	norm
0	1	0	1	6
1	1	0.166667	1.83333	1.73205
2	1	2.77556e-17	1.83333	0.946077
3	1	0.00641026	1.99359	0.171234
4	1	1.44849e-16	1.99359	0.0362621
5	1	1.024e-05	1.99999	0.00641052
6	1	-8.89317e-17	1.99999	5.79263e-05
7	1	2.62143e-11	2	1.024e-05
8	5.68434e-14	2.62143e-11	2	1.4829e-10

6 вариант

k	beta	x	y	norm
0	1	0	1	6

1	1	0.166667	1.83333	1.73205
2	1	5.55112e-17	1.83333	0.946077
3	1	0.00641026	1.99359	0.171234
4	1	-3.2873e-16	1.99359	0.0362621
5	1	-1.03059e-05	2.00001	0.00641052
6	1	8.73443e-17	2.00001	5.82989e-05
7	1	2.65529e-11	2	1.03059e-05
8	1	-8.45827e-17	2	1.50206e-10
9	1	-8.45813e-17	2	5.31057e-11
10	5.68434e-14	-8.45813e-17	2	0

Две окружности и прямая 6.



1 вариант

k	beta	x	y	norm
0	0.5	-4.58333	4.08333	45.5522
1	1	-2.71577	4.71577	44.3107
2	1	-2.1493	4.1493	25.9073
3	1	-2.0097	4.0097	17.8477
4	1	-2.00005	4.00005	16.1166
5	1	-2	4	16.0006
6	1	-2	4	16
7	5.68434e-14	-2	4	16

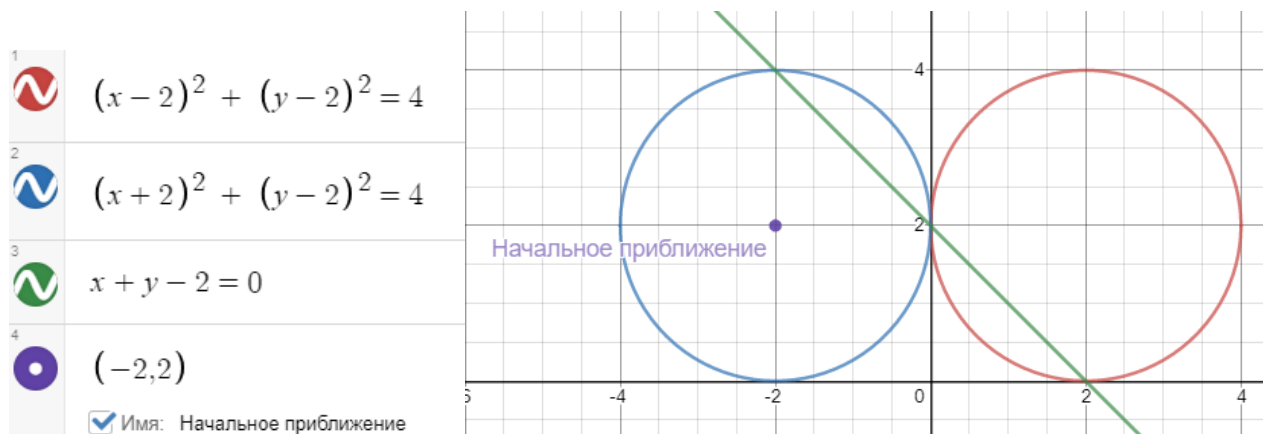
2 вариант

k	beta	x	y	norm
0	1	-1.78571	3.78571	45.5522
1	1	-2.22045e-16	3.78571	13.5421
2	1	0	2.89286	4.85029
3	0.5	0.180012	2.26642	1.43813
4	1	5.55112e-17	2.19402	1.12143
5	1	0.0104223	1.98958	0.201195
6	1	-8.67362e-18	1.98958	0.0589585
7	1	2.70155e-05	1.99997	0.0104235
8	1	8.40595e-18	1.99997	0.000152823
9	1	1.82457e-10	2	2.70155e-05
10	1	-1.03915e-16	2	1.03213e-09
11	1	-1.03849e-16	2	3.64914e-10
12	5.68434e-14	-1.03849e-16	2	0

6 вариант

k	beta	x	y	norm
0	1	-1.78571	3.78571	45.5522
1	1	9.72555e-14	3.78571	13.5421
2	1	1.088e-26	2.89286	4.85029
3	0.5	0.180012	2.26642	1.43813
4	1	-2.98928e-14	2.19402	1.12143
5	1	-0.00857899	2.00858	0.201195
6	1	-9.95731e-16	2.00858	0.0485306
7	1	-1.83212e-05	2.00002	0.00857962
8	1	3.90109e-17	2.00002	0.00010364
9	1	8.39174e-11	2	1.83212e-05
10	1	7.6334e-17	2	4.74708e-10
11	1	7.6348e-17	2	1.67724e-10
12	5.68434e-14	7.6348e-17	2	0

Две окружности и прямая 7.



1 вариант

Cant solve!




2 вариант

Cant solve!

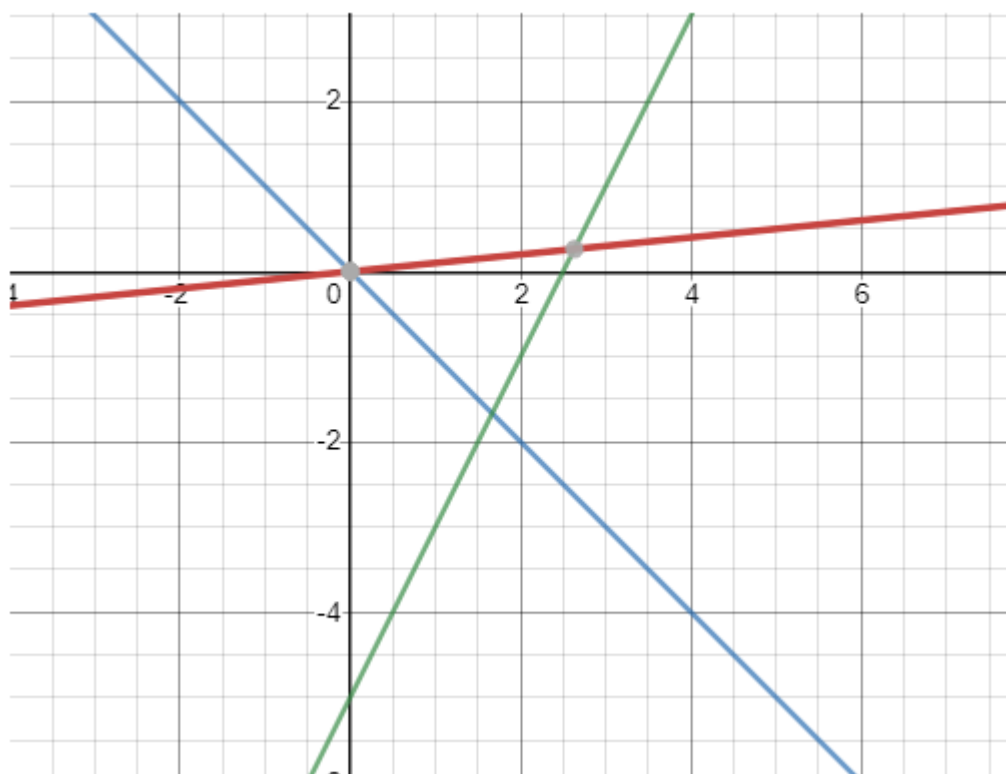
6 вариант

Cant solve!

5. Даны три попарно прямые. Исследовать сходимость метода в зависимости от начальных приближений

1		$10y = x$	×
2		$y = -x$	×
3		$y = 2x - 5$	×

График



Вариант 1

Начальное приближение (1,0)

k	beta	x	y	norm
0	5.68434e-14	1	-9.4739e-14	3.31662
1	5.68434e-14	1	-1.89478e-13	3.31662

Начальное приближение (10,0)

k	beta	x	y	norm
0	1	1.66667	-1.66667	20.6155
1	1	1.66667	-1.66667	18.3333

Начальное приближение (-1,6)

k	beta	x	y	norm
0	1	1.66667	-1.66667	62.57
1	1	1.66667	-1.66667	18.3333

Вариант 2

Начальное приближение (1,0)

k	beta	x	y	norm
0	5.68434e-14	1	-9.4739e-14	3.31662
1	1	2.63158	0.263158	3.31662
2	0.5	1.31579	0.131579	2.89474
3	0.03125	1.32675	0.0753838	2.88875

Начальное приближение (10,0)

k	beta	x	y	norm
0	1	1.66667	-1.66667	20.6155
1	1	2.63158	0.263158	18.3333
2	0.5	1.31579	0.131579	2.89474
3	0.03125	1.32675	0.0753838	2.88875

Начальное приближение (-1,6)

k	beta	x	y	norm
0	1	2.63158	0.263158	62.57
1	0.5	1.31579	0.131579	2.89474
2	0.03125	1.32675	0.0753838	2.88875

Вариант 6

Начальное приближение (1,0)

k	beta	x	y	norm
0	5.68434e-14	1	-9.4739e-14	3.31662
1	1	2.63158	0.263158	3.31662
2	0.5	1.31579	0.131579	2.89474
3	0.03125	1.32675	0.0753838	2.88875

Начальное приближение (10,0)

k	beta	x	y	norm
0	1	1.66667	-1.66667	20.6155
1	1	2.63158	0.263158	18.3333
2	0.5	1.31579	0.131579	2.89474
3	0.03125	1.32675	0.0753838	2.88875

Начальное приближение (-1,6)

k	beta	x	y	norm
0	1	2.63158	0.263158	62.57
1	0.5	1.31579	0.131579	2.89474
2	0.03125	1.32675	0.0753838	2.88875

6. Исследовать влияние взвешивания уравнений СНУ (умножения уравнений СНУ на некоторые веса).

Функции как в исследовании 5, взвешиваем «зеленую» прямую

Начальное приближение (-5, 6)

Вариант 1

Alpha	Result		Number of iteration
0.25	1.66667	-1.66667	3
2	1.66667	-1.66667	2
10	0	0	2
1000	0	0	2




Вариант 2

Alpha	Result		Number of iteration
0.25	0.66766	0.0560238	6
2	2.28276	0.170984	4
10	2.59126	0.202058	3
1000	2.60142	0.202851	3

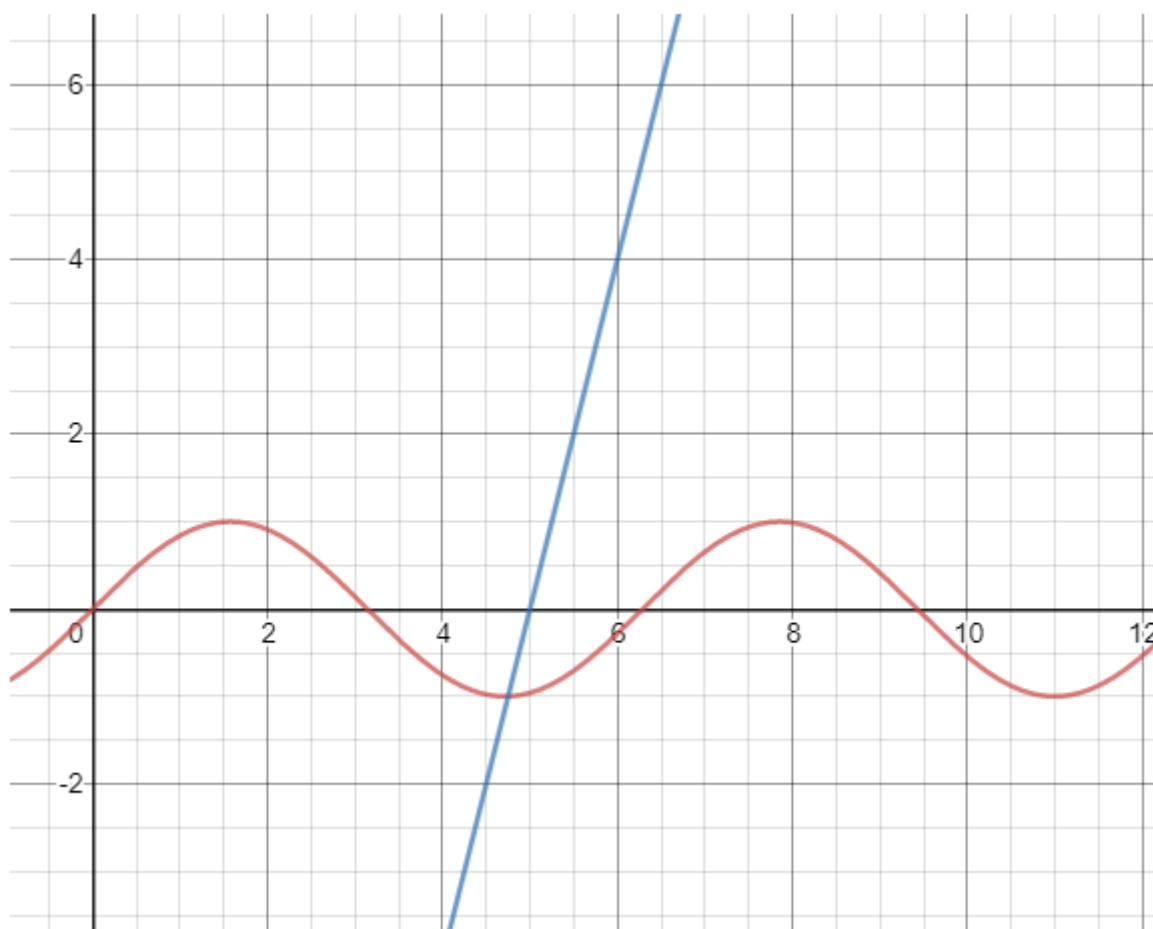
Вариант 6

Alpha	Result		Number of iteration
0.25	0.66766	0.0560238	5
2	2.28276	0.170984	4
10	2.59126	0.202058	3
1000	2.60142	0.202851	2

7. Исследовать сходимость метода Ньютона для СНУ с локальными минимумами в зависимости от начальных приближений (например, на СНУ, состоящей из синусоиды и прямой с некоторым наклоном, которая пересекает синусоиду).

1	 $y = \sin(x)$	
2	 $y = 4x - 20$	

График



Вариант 1

Начальное приближение (0,0)

k	beta	x	y	norm
0	1	1.66667	1.66667	5
1	1	1.50277	1.0111	0.671259
2	1	1.49936	0.997456	0.013409
3	1	1.49936	0.99745	5.80079e-06

4	1	1.49936	0.99745	1.08735e-12
5	5.68434e-14	1.49936	0.99745	0

Начальное приближение (1,0)

k	beta	x	y	norm
0	1	5.8679	3.47161	16.0221
1	1	4.6118	-1.55278	3.87506
2	1	4.74785	-1.00861	0.557835
3	1	4.75018	-0.999289	0.00923507
4	1	4.75018	-0.999286	2.7113e-06
5	1	4.75018	-0.999286	2.33951e-13

Начальное приближение (3*PI/2,1)

k	beta	x	y	norm
0	1	4.75	-1	2.93673
1	1	4.75018	-0.999286	0.000707211
2	1	4.75018	-0.999286	1.59164e-08
3	5.68434e-14	4.75018	-0.999286	0

Вариант 2

Начальное приближение (0,0)

k	beta	x	y	norm
0	1	1.66667	1.66667	5
1	1	1.50277	1.0111	0.671259
2	1	1.49936	0.997456	0.013409
3	1	1.49936	0.99745	5.80079e-06
4	1	1.49936	0.99745	1.08735e-12
5	5.68434e-14	1.49936	0.99745	0

Начальное приближение (1,0)

k	beta	x	y	norm
0	1	5.8679	3.47161	16.0221
1	1	4.6118	-1.55278	3.87506
2	1	4.74785	-1.00861	0.557835
3	1	4.75018	-0.999289	0.00923507
4	1	4.75018	-0.999286	2.7113e-06
5	1	4.75018	-0.999286	2.33951e-13

Начальное приближение (3*PI/2,1)

k	beta	x	y	norm
0	1	4.75	-1	2.93673
1	1	4.75018	-0.999286	0.000707211
2	1	4.75018	-0.999286	1.59164e-08
3	5.68434e-14	4.75018	-0.999286	0

Вариант 6

Начальное приближение (0,0)

k	beta	x	y	norm
0	1	1.66667	1.66667	5
1	1	1.50277	1.0111	0.671258
2	1	1.49936	0.997456	0.0134089
3	1	1.49936	0.99745	5.80081e-06
4	1	1.49936	0.99745	1.10489e-12
5	5.68434e-14	1.49936	0.99745	0

Начальное приближение (1,0)

k	beta	x	y	norm
0	1	5.8679	3.47161	16.0221
1	1	4.6118	-1.55278	3.87506
2	1	4.74785	-1.00861	0.557835
3	1	4.75018	-0.999289	0.00923506
4	1	4.75018	-0.999286	2.71131e-06
5	1	4.75018	-0.999286	2.3828e-13

Начальное приближение (3*PI/2,1)

k	beta	x	y	norm
0	1	4.75	-1	2.93673
1	1	4.75018	-0.999286	0.000707211
2	1	4.75018	-0.999286	1.59175e-08
3	5.68434e-14	4.75018	-0.999286	0

8. Исследовать влияние размера шага при численном вычислении производных на сходимость метода Ньютона. Используется следующая разностная схема:

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h}$$

Функция как в исследовании 7

Начальное приближение (0,1)

Eps(шаг)	Iterations	Result
1e-13	9	(4.75018; -0.999286)
1e-11	7	(4.75018; -0.999286)
1e-9	6	(4.75018; -0.999286)
.	.	.
0,001	6	(4.75018; -0.999286)
0,1	7	(4.75018; -0.999286)
1	8	(4.75018; -0.999286)
10	9	(4.75018; -0.999286)

9. Выводы

Артур: при расположении начального приближения на прямой, соединяющей центры окружностей в ходе прямого хода методом гаусса последняя строка матрицы состоит целиком из нулей, в результате чего при обратном ходе получаются бесконечные значения компонент искомого вектора, что приводит к аварийному выходу программы с сообщением “Cant solve!”.

При нахождении начального приближения на оси, перпендикулярной, оси соединяющей центры окружностей и пересекающей ее на равных расстояниях от центров окружностей метод не может сойтись и совершает колебания равной амплитуды на этой оси.

Для окружностей с двумя пересечениями видно как находится ближайшее к начальному приближению решение.

Стоит отметить, что на тестах, для которых программа может найти решение, первый шаг всегда приводит к линии симметрии.

При взвешивании прямых метод не только начинает сходиться к одному из пересечений прямых, но еще и делает это за меньшее число итераций.

При размещении начального приближения у локального минимума, рядом с которым находится решение, решение находится за меньшее число итераций.

Увеличение шага при численном вычислении производной положительно сказывается на количестве итераций, но только до определённого момента.

Анатолий: Сходимость метода Ньютона зависит от начального приближения.

Если задать начальное приближение на прямой центров окружностей, то получаем вырожденную СЛАУ и решение найти невозможно. Взвешивание уравнений заставляет решение сходиться к уравнению с большим весом, причём чем больше вес, тем быстрее.

Сходимость в системе из трёх прямых не зависит от начального приближения и сходится к некоторой точке.

В СЛУ с локальными минимумами решение может не сходиться к ближайшему пересечению графиков функций.

При численном вычислении элементов матрицы Якоби с уменьшением шага уменьшается количество итераций, соответственно, увеличивается скорость схождения к решению системы.