МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский Нижегородский государственный университет им. Н. И. Лобачевского»

Институт информационных технологий математики и механики

Кафедра теоретической, компьютерной и экспериментальной механики

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) программы бакалавриата: Математическое моделирование и вычислительная математика

ОТЧЕТ по теме: «Методы оптимизации»

Вариант 7

Выполнил:

студент группы 3821Б1ПМмм1

Чувашев Анатолий Александрович

Проверил: Чернов Андрей Владимирович

Нижний Новгород, 2024 г

Постановка задачи.

С помощью методов: Ньютона, Ньютона-Рафсона, Полака-Рибьера, ФлетчераРивса, DFP, BFGS нужно найти минимум функции с заданной точностью.

Функция имеет вид:

$$\boldsymbol{f(x,y) = 10^2 * (y^2 - e^{2x})^2 + (x + y - 1)^2 \rightarrow min,} \qquad (x, y) \in \mathbb{R}^2$$

Реализация методов.

Вспомогательные функции:

1) Отрисовка графика:

```
record ContourConfig(double Min, double Max, ScottPlot.Color LineColor);

void GenerateCoordinates(Coordinates3d[,] array, Func<Coord, double> func, ContourConfig
configs)
{
    for (int y = 0; y < array.GetLength(0); y++)
    {
        for (int x = 0; x < array.GetLength(1); x++)
        {
            double xCoord = -4.0 + x * (8.0 / (array.GetLength(1) - 1));
            double yCoord = -4.0 + y * (8.0 / (array.GetLength(0) - 1));
            double z = func(new Coord(xCoord, yCoord));

            if (z >= configs.Min && z <= configs.Max)
            {
                array[y, x] = new Coordinates3d(xCoord, yCoord, z);
            }
            else
                array[y, x] = new Coordinates3d(xCoord, yCoord, 0);
        }
    }
}

void AddContours(Coordinates3d[,] array, ContourConfig config)
{
    var contour = this.WpfPlot1.Plot.Add.ContourLines(array, 5);
    contour.LabelStyle.FontSize = 0;
    contour.LineColor = config.LineColor;
}

// Метод для создания графика
private void PlotGraph()
{
    this.WpfPlot1.Plot.Clear();

    Coordinates3d[,] cs = new Coordinates3d[1000, 1000];
    Coordinates3d[,] cs1 = new Coordinates3d[1000, 1000];
    Coordinates3d[,] cs2 = new Coordinates3d[1000, 1000];
    Coordinates3d[,] cs3 = new Coordinates3d[1000, 1000];
    Coordinates3d[,] cs4 = new Coordinates3d[1000, 1000];
    Coordinates3d[,] cs5 = new Coordinates3d[1000, 1000];

    var configs = new[]
    {
        new ContourConfig(0, 10, ScottPlot.Colors.Blue),    // Синий для минимального
диапазона
        new ContourConfig(10, 50, ScottPlot.Colors.Cyan),   // Переход от синего к
голубому
        new ContourConfig(50, 200, ScottPlot.Colors.Yellow), // Желтый
        new ContourConfig(200, 400, ScottPlot.Colors.Orange), // Оранжевый
        new ContourConfig(400, 600, ScottPlot.Colors.Red),    // Красный
        new ContourConfig(600, 1000, ScottPlot.Colors.Red)    // Красный
    };

    GenerateCoordinates(cs, Func,  configs[0]);
    GenerateCoordinates(cs1, Func, configs[1]);
    GenerateCoordinates(cs2, Func, configs[2]);
    GenerateCoordinates(cs3, Func, configs[3]);
    GenerateCoordinates(cs4, Func, configs[4]);
    GenerateCoordinates(cs5, Func, configs[5]);

    AddContours(cs, configs[0]);
```

```
    AddContours(cs1, configs[1]);
    AddContours(cs2, configs[2]);
    AddContours(cs3, configs[3]);
    AddContours(cs4, configs[4]);
    AddContours(cs5, configs[5]);

    List<double> x0_array = new List<double>();
    List<double> y0_array = new List<double>();

    for (int i = 0; i < coords.Count; ++i)
    {
        x0_array.Add(coords[i].m_x);
        y0_array.Add(coords[i].m_y);
    }

    // Добавление точек траектории
    this.WpfPlot1.Plot.Add.Scatter(x0_array.ToArray(), y0_array.ToArray(),
ScottPlot.Colors.Purple);

    // Настройка графика
    this.WpfPlot1.Plot.Title("Contour Plot with Optimization Path");
    this.WpfPlot1.Plot.XLabel("X");
    this.WpfPlot1.Plot.YLabel("Y");
    this.WpfPlot1.Plot.Axes.AutoScale();
    this.WpfPlot1.Refresh();
}
```

2) Проверка Гессиана на положительную определенность:

```
bool CheckHessian(Matrix<double> hessian)
{
    if (hessian.RowCount != 2 || hessian.ColumnCount != 2)
        throw new ArgumentException("Этот метод предназначен только для 2x2
матриц.");

    double det = hessian[0, 0] * hessian[1, 1] - hessian[0, 1] * hessian[1, 0];
    Console.WriteLine($"Hessian[0,0] = {hessian[0, 0]}, det(H) = {det}");

    if (hessian[0, 0] > 0 && det > 0)
    {
        Console.WriteLine("Hessian is positive definite.");
        return true;
    }
    else
    {
        Console.WriteLine("Hessian is NOT positive definite.");
        return false;
    }

}
```

3) Реализация функции, градиента и гессиана:

```
public double Func(Coord coord)
{
    return 100 * (Math.Pow(coord.m_y, 2) - Math.Exp(2 * coord.m_x)) *
(Math.Pow(coord.m_y, 2) - Math.Exp(2 * coord.m_x)) +
        (coord.m_x + coord.m_y - 1) * (coord.m_x + coord.m_y - 1);
}
public Coord Gradient(Coord coord)
{
    double df_dx = -400 * (Math.Pow(coord.m_y, 2) - Math.Exp(2 * coord.m_x)) *
Math.Exp(2 * coord.m_x) +
        2 * (coord.m_x + coord.m_y - 1);

    double df_dy = 400 * coord.m_y * (Math.Pow(coord.m_y, 2) - Math.Exp(2 *
coord.m_x)) + 2 * (coord.m_x + coord.m_y - 1);
```

```
    return new Coord(df_dx, df_dy);
}
public Matrix<double> Hessian(Coord coord)
{

    double d2f_dx2 = -800 * ((Math.Pow(coord.m_y, 2) - Math.Exp(2 * coord.m_x)) *
Math.Exp(2 * coord.m_x) -
       Math.Exp(4 * coord.m_x)) + 2;
    double d2f_dy2 = 1200 * Math.Pow(coord.m_y, 2) - 400 * Math.Exp(2 * coord.m_x) +
2;
    double d2f_dxdy = -800 * coord.m_y * Math.Exp(2 * coord.m_x) + 2;
    double d2f_dydx = d2f_dxdy;

    Matrix<double> matrix = Matrix<double>.Build.DenseOfArray(new double[,]
    {
        { d2f_dx2, d2f_dxdy },
        { d2f_dydx, d2f_dy2 }
    });


    return matrix;
}
```

4) Метод Ньютона:

```
public void Newton(Coord startCoord, int maxIters, double eps)
{
    coords.Clear();

    Coord currentCoord = startCoord;

    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    int iters = 0;

    for (int i = 0; i < maxIters; ++i)
    {
        Coord coordsOfgradient = Gradient(currentCoord);

        Matrix<double> hessian = Hessian(currentCoord);
        Vector<double> gradient = Vector<double>.Build.DenseOfArray(new double[] {
coordsOfgradient.m_x, coordsOfgradient.m_y });

        if (CheckHessian(hessian))
        {
            var step = hessian.Inverse() * gradient;

            // Обновление координат
            currentCoord.m_x -= step[0];
            currentCoord.m_y -= step[1];

            iters += 1;

            coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

            coordsOfgradient = Gradient(currentCoord);
            gradient = Vector<double>.Build.DenseOfArray(new double[] {
coordsOfgradient.m_x, coordsOfgradient.m_y });

            if (gradient.L2Norm() < eps)
            {
                Console.WriteLine($"Метод сходится к точке ({currentCoord.m_x},
{currentCoord.m_y}) за {iters} итераций.");
                break;
```

```
            }
        }
        else
        {
            Console.WriteLine("МЕТОД НЬЮТОНА НЕ ПРИМЕНИМ");
            break;
        }

    }

    Console.WriteLine($"Результат: x = {currentCoord.m_x}, y = {currentCoord.m_y}");
}
```

5) Метод Ньютона-Рафсона:

```
public void NewtonRaphson(Coord startCoord, int maxIters, double eps)
{
    coords.Clear(); // очищаем предыдущие точки

    Coord currentCoord = startCoord;

    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    int iters = 0;

    for (int i = 0; i < maxIters; ++i)
    {
        Matrix<double> hessian = Hessian(currentCoord);

        if (CheckHessian(hessian))
        {
            Coord coordsOfgradient = Gradient(currentCoord);
            Vector<double> gradient = Vector<double>.Build.DenseOfArray(new double[]
{ coordsOfgradient.m_x, coordsOfgradient.m_y });

            var pk = -1 * (hessian.Inverse() * gradient);

            double alpha = FindAlpha(pk, currentCoord);

            currentCoord.m_x += alpha * pk[0];
            currentCoord.m_y += alpha * pk[1];

            iters += 1;

            coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

            coordsOfgradient = Gradient(currentCoord);
            gradient = Vector<double>.Build.DenseOfArray(new double[] {
coordsOfgradient.m_x, coordsOfgradient.m_y });

            if (gradient.L2Norm() < eps)
            {
                Console.WriteLine($"Метод сходится к точке ({currentCoord.m_x},
{currentCoord.m_y}) за {iters} итераций.");
                break;
            }
        }
        else
        {
            Console.WriteLine("МЕТОД НЬЮТОНА-РАПСОНА НЕ ПРИМЕНИМ");
            break;
        }
    }
}
```

6) Метод Полака-Рибьера:

```
public void PolakaRibiera(Coord startCoord, int maxIters, double eps)
{
    coords.Clear();

    Coord currentCoord = startCoord;
    Coord saveCoord = new Coord(-2, -2);

    Vector<double> pk = Vector<double>.Build.DenseOfArray(new[] { 0.0, 0.0 });
    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    int iters = 0;

    for (int i = 0; i < maxIters; ++i)
    {
        double betta = 0;

        if (i == 0)
            betta = 0;
        else
            betta = GetBettaForPR(currentCoord, saveCoord);

        Coord coordsOfgradient = Gradient(currentCoord);
        Vector<double> gradient = Vector<double>.Build.DenseOfArray(new double[] {
coordsOfgradient.m_x, coordsOfgradient.m_y });

        pk = -gradient + betta * pk;

        double alpha = FindAlpha(pk, currentCoord);

        Console.WriteLine($"Alpha: {alpha}");

        saveCoord.m_x = currentCoord.m_x;
        saveCoord.m_y = currentCoord.m_y;

        currentCoord.m_x += alpha * pk[0];
        currentCoord.m_y += alpha * pk[1];

        iters += 1;

        coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

        if (gradient.L2Norm() < eps)
        {
            Console.WriteLine($"Метод сходится к точке ({currentCoord.m_x},
{currentCoord.m_y}) за {iters} итераций.");
            break;
        }
    }

    if (iters == maxIters)
    {
        Console.WriteLine("Метод не сходится за максимальное количество итераций.");
    }
}
```

7) Метод Флетчера-Ривса:

```
public void FletcherReeves(Coord startCoord, int maxIters, double eps)
{
    coords.Clear();

    Coord currentCoord = startCoord;
    Coord saveCoord = new Coord(currentCoord.m_x, currentCoord.m_y);
```

```
    Vector<double> pk = Vector<double>.Build.DenseOfArray(new[] { 0.0, 0.0 });
    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    int iters = 0;

    for (int i = 0; i < maxIters; ++i)
    {
        Console.WriteLine($"Итерация {i + 1}: x = {currentCoord.m_x}, y =
{currentCoord.m_y}, f(x, y) = {Func(currentCoord)}");

        double betta = 0;

        if (i == 0)
            betta = 0;
        else
            betta = GetBettaForFR(currentCoord, saveCoord);

        Coord coordsOfgradient = Gradient(currentCoord);
        Vector<double> gradient = Vector<double>.Build.DenseOfArray(new double[] {
coordsOfgradient.m_x, coordsOfgradient.m_y });

        // Обновление направления pk
        pk = -gradient + betta * pk;

        double alpha = FindAlpha(pk, currentCoord);

        // Сохраняем текущую координату для следующего шага
        saveCoord.m_x = currentCoord.m_x;
        saveCoord.m_y = currentCoord.m_y;

        // Обновление текущей координаты
        currentCoord.m_x += alpha * pk[0];
        currentCoord.m_y += alpha * pk[1];

        iters += 1;

        // Добавляем текущую координату в список
        coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

        // Проверка на сходимость
        if (gradient.L2Norm() < eps)
        {
            Console.WriteLine($"Метод сходится к точке ({currentCoord.m_x},
{currentCoord.m_y}) за {iters} итераций.");
            break;
        }
    }

    // Выводим сообщение, если метод не сходится
    if (iters == maxIters)
    {
        Console.WriteLine("Метод не сходится за максимальное количество итераций.");
    }
}
```

8) Метод Давидона-Флетчера-Пауэлла (DFP):

```
public void DFP(Coord startCoord, int maxIters, double eps)
{
    coords.Clear();

    Coord currentCoord = startCoord;
    Coord saveCoord = new Coord(0, 0);
    Coord yCoord = new Coord(0, 0);
```

```
    Vector<double> pk = Vector<double>.Build.DenseOfArray(new[] { 0.0, 0.0 });
    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    double alpha = 0;

    int iters = 0;

    // Вычисление Гессиана
    Matrix<double> hessian = Hessian(currentCoord);

    for (int i = 0; i < maxIters; ++i)
    {

        Console.WriteLine($"Итерация {i + 1}: x = {currentCoord.m_x}, y =
{currentCoord.m_y}, f(x, y) = {Func(currentCoord)}");

        // Градиенты в текущей и сохраненной координатах
        Coord gradCurrent = Gradient(currentCoord);
        Coord gradSave = Gradient(saveCoord);

        // Разница градиентов
        yCoord.m_x = gradCurrent.m_x - gradSave.m_x;
        yCoord.m_y = gradCurrent.m_y - gradSave.m_y;

        Vector<double> y = Vector<double>.Build.DenseOfArray(new[] { yCoord.m_x,
yCoord.m_y });

        // Инициализация или обновление Гессиана
        if (i == 0)
            hessian = Hessian(currentCoord).Inverse();
        else
            hessian = FindH_DFP(hessian, y, pk, alpha);

        // Направление pk
        Vector<double> gradVector = Vector<double>.Build.DenseOfArray(new[] {
gradCurrent.m_x, gradCurrent.m_y });
        pk = -1 * (hessian * gradVector);

        // Шаг alpha
        alpha = FindAlpha(pk, currentCoord);

        // Сохраняем текущую координату для следующего шага
        saveCoord.m_x = currentCoord.m_x;
        saveCoord.m_y = currentCoord.m_y;

        // Обновление координат
        currentCoord.m_x += alpha * pk[0];
        currentCoord.m_y += alpha * pk[1];

        // Сохранение результатов
        coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));
        iters++;

        // Проверка условия завершения
        if (gradVector.L2Norm() < eps)
        {
            Console.WriteLine("Optimization converged");
            break;
        }
    }
}
```

9) Метод Бройдена-Флетчера-Гольдфраба-Шенно (BFGS):

```
public void BFGS(Coord startCoord, int maxIters, double eps)
{
    coords.Clear();

    Coord currentCoord = startCoord;
    Coord saveCoord = new Coord(0, 0);
    Coord yCoord = new Coord(0, 0);
    Coord dCoord = new Coord(0, 0);

    Vector<double> pk = Vector<double>.Build.DenseOfArray(new[] { 0.0, 0.0 });
    coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));

    double alpha = 0;

    int iters = 0;

    // Вычисление Гессиана
    Matrix<double> hessian = Hessian(currentCoord);

    for (int i = 0; i < maxIters; ++i)
    {

        Console.WriteLine($"Итерация {i + 1}: x = {currentCoord.m_x}, y =
{currentCoord.m_y}, f(x, y) = {Func(currentCoord)}");

        // Градиенты в текущей и сохраненной координатах
        Coord gradCurrent = Gradient(currentCoord);
        Coord gradSave = Gradient(saveCoord);

        // Разница градиентов
        yCoord.m_x = gradCurrent.m_x - gradSave.m_x;
        yCoord.m_y = gradCurrent.m_y - gradSave.m_y;

        dCoord.m_x = currentCoord.m_x - saveCoord.m_x;
        dCoord.m_y = currentCoord.m_y - saveCoord.m_y;

        Vector<double> y = Vector<double>.Build.DenseOfArray(new[] { yCoord.m_x,
yCoord.m_y });
        Vector<double> d = Vector<double>.Build.DenseOfArray(new[] { dCoord.m_x,
dCoord.m_y });

        // Инициализация или обновление Гессиана
        if (i == 0)
            hessian = Hessian(currentCoord);
        else
            hessian = FindH_BFGS(hessian, d, y);

        // Направление pk
        Vector<double> gradVector = Vector<double>.Build.DenseOfArray(new[] {
gradCurrent.m_x, gradCurrent.m_y });

        pk = -1 * (hessian.Inverse() * gradVector);

        // Шаг alpha
        alpha = FindAlpha(pk, currentCoord);

        // Сохраняем текущую координату для следующего шага
        saveCoord.m_x = currentCoord.m_x;
        saveCoord.m_y = currentCoord.m_y;

        // Обновление координат
        currentCoord.m_x += alpha * pk[0];
        currentCoord.m_y += alpha * pk[1];

        // Сохранение результатов
```

```
        coords.Add(new Coord(currentCoord.m_x, currentCoord.m_y));
        iters++;

        // Проверка условия завершения
        if (gradVector.L2Norm() < eps)
        {
            Console.WriteLine("Optimization converged");
            break;
        }
    }
}
```

Исследование

Для запуска какого-либо из методов необходимо вызвать его с помощью строки NameOfMethod(new Coord(start.x, start.y), countOfIters, eps) и отобразить полученный результат с помощью метода PlotGraph().

1) Рассмотрим метод Ньютона. Для этого запустим его с помощью Newton(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат.

Hessian[0,0] = -56,073304239305344, det(H) = -269833,2040167645

Hessian is NOT positive definite.

МЕТОД НЬЮТОНА НЕ ПРИМЕНИМ

Результат: x = -2, y = -2

Как мы видим, из-за того, что Гессиан в точке (-2; -2) не положительно определен, то метод Ньютона не применим. Кроме того, в точке (2; -2) метод тоже не применим.

2) Рассмотрим метод Ньютона-Рафсона. Для этого запустим его с помощью NewtonRaphson(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат.

Hessian[0,0] = -56,073304239305344, det(H) = -269833,2040167645

Hessian is NOT positive definite.

МЕТОД НЬЮТОНА-РАПСОНА НЕ ПРИМЕНИМ

Из-за того, что Гессиан не положительно определен, то мы не можем применить метод Ньютона-Рафсона.

3) Рассмотрим метод Полака-Рибьера. Для этого запустим его с помощью PolakaRibiera(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат:

Метод сходится к точке (2,926155479762096E-11, 1,0000000000293487) за 13 итераций.

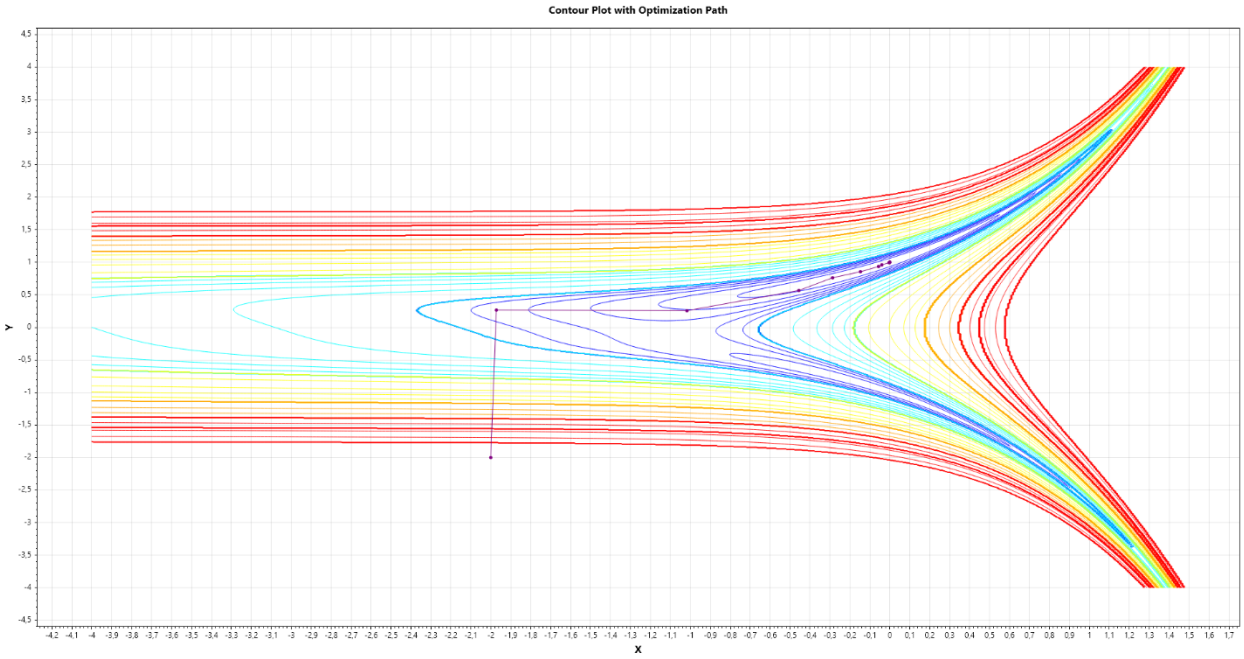

Рисунок 1. Метод Полака-Рибьера

4) Рассмотрим метод Флетчера-Ривса. Для этого запустим его с помощью FletcherReeves(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат:

Метод сходится к точке (-2,829939417048103E-08, 0,9999999718733347) за 39 итераций.

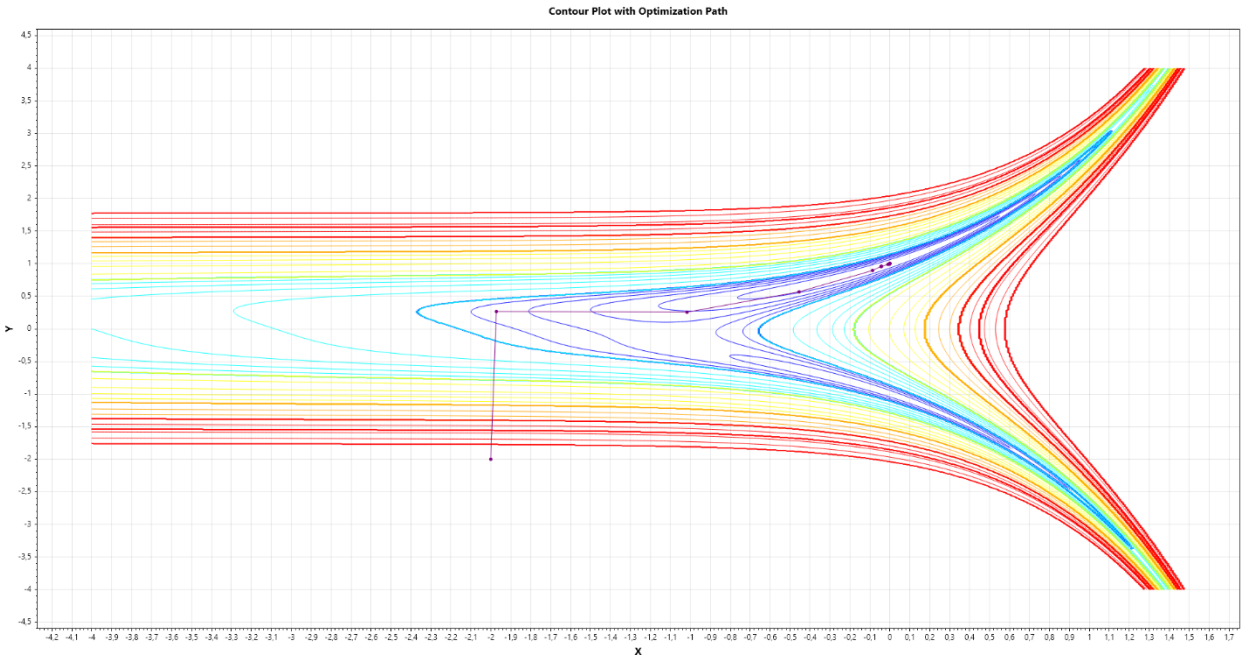

Рисунок 2. Метод Флетчера-Ривса

5) Рассмотрим метод Давидона-Флетчера-Пауэлла (DFP). Для этого запустим его с помощью DFP(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат:

Метод сходится к точке (-3,358291589930611E-11, 1,0000000000655285) за 13 итераций.

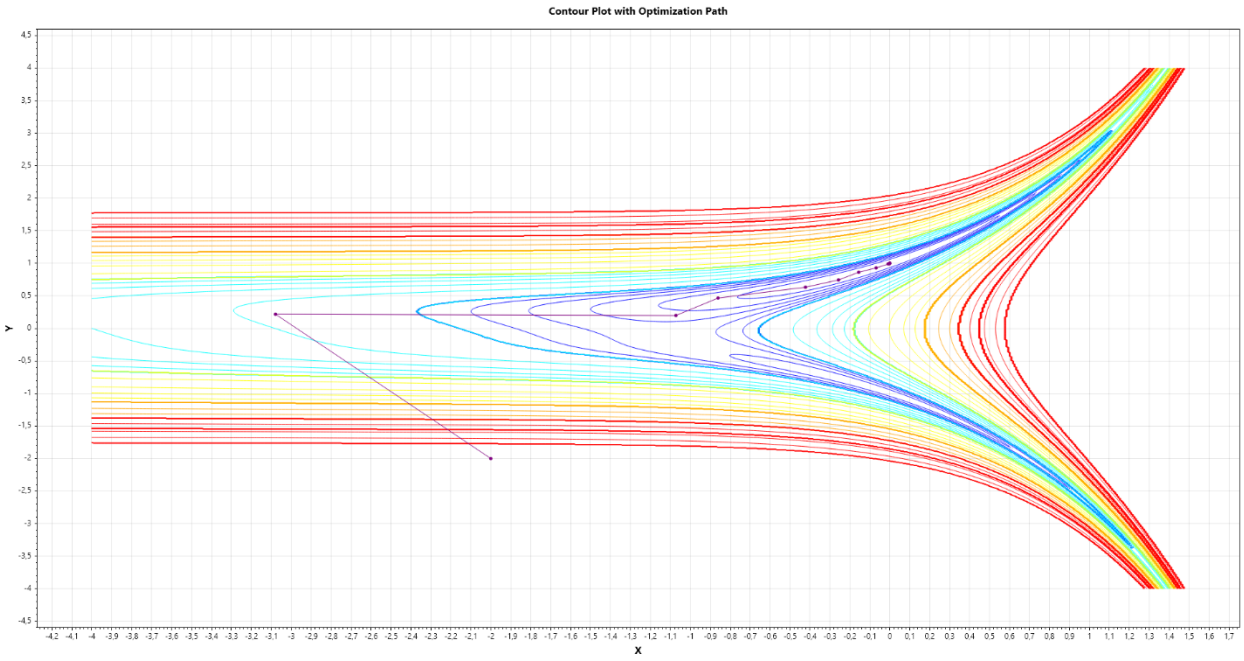

Рисунок 3. Метод Давидона-Флетчера-Пауэлла

6) Рассмотрим метод Бройдена-Флетчера-Гольдфарба-Шенн (BFGS). Для этого запустим его с помощью BFGS(new Coord(-2, -2), 100, 1e-6); и посмотрим на результат:

Метод сходится к точке (-3,4987201799324666E-11, 1,0000000000441187) за 13 итераций.

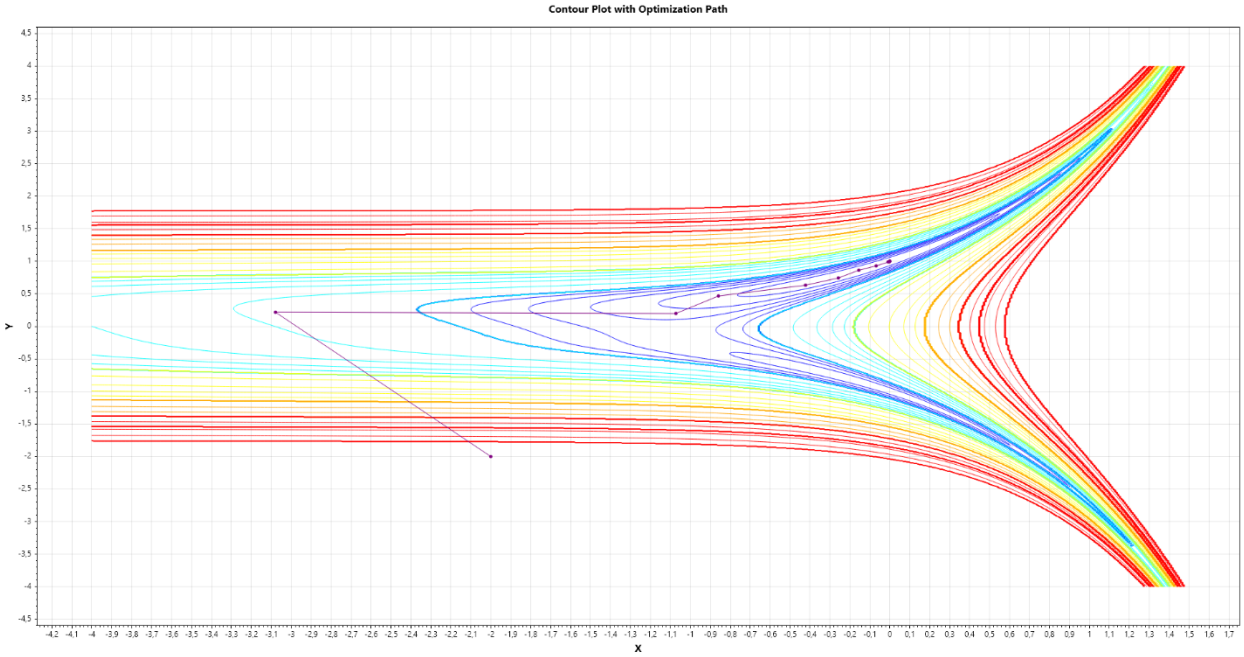

Рисунок 4. Метод Бройдена-Флетчера-Гольдфарба-Шенн

Выводы

В результате проделанной мною работы по лабораторной были реализованы и протестированы 6 методов поиска минимума заданной фунции.