

Цель работы

Знакомство с задачей минимизации функций многих переменных методами.

Задания

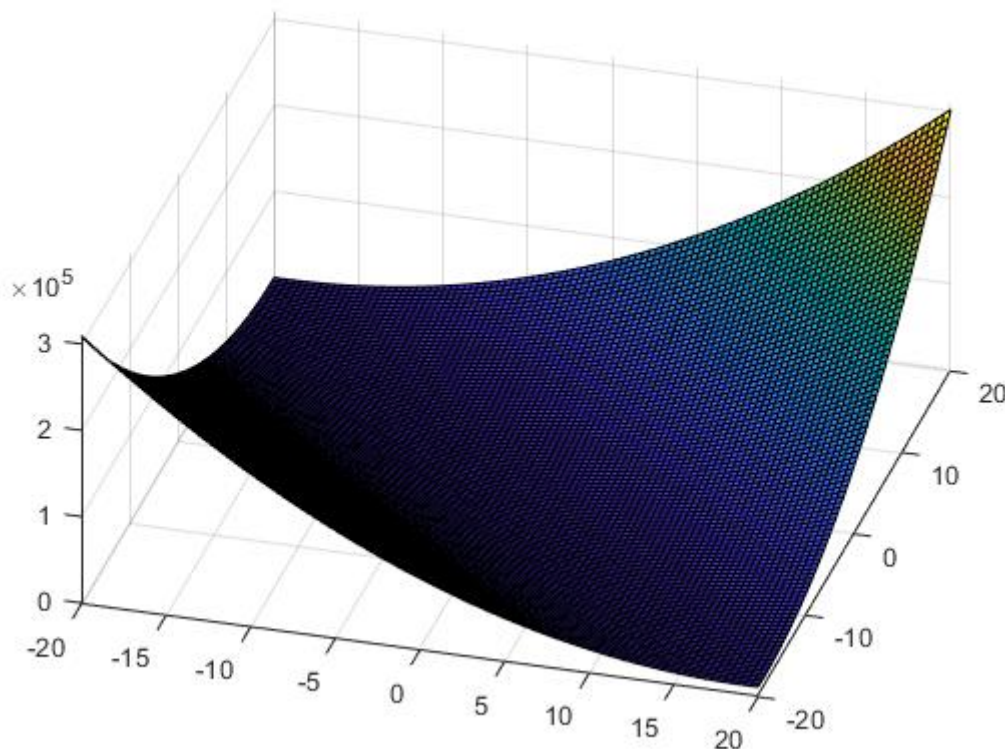
Реализовать алгоритмы одномерной минимизации функции:

- Метод градиентного спуска;
- Метод наискорейшего спуска;
- Метод сопряженных градиентов;
- Метод Нелдера-Мида

Индивидуальный вариант для заданий:

$$f(x) = 194x_1^2 + 376x_1x_2 + 194x_2^2 + 31x_1 - 229x_2 + 4.$$

График функции:



Выполнение заданий

Для нахождения минимума исходной функции необходимо составить систему

$$\begin{cases} df/dx_1 = 0; \\ df/dx_2 = 0. \end{cases}$$

и решить её.

$$\begin{cases} df/dx_1 = 0; \\ df/dx_2 = 0. \end{cases} = \begin{cases} 388x_1 + 376x_2 + 31 = 0; \\ 376x_1 + 388x_2 - 229 = 0. \end{cases}$$

Решаем систему с помощью метода Крамера. Получаем $x_1 \approx -10,704$, $x_2 \approx 10,963$.

Проверим достаточные условия:

$$H(x) = \begin{pmatrix} 388 & 376 \\ 376 & 388 \end{pmatrix}$$

Угловой минор первого порядка = 388;

Угловой минор второго порядка = 9168.

Следовательно, точка x является минимумом.

$f(x) \approx -1417,161$.

Метод градиентного спуска

Реализация кода в C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

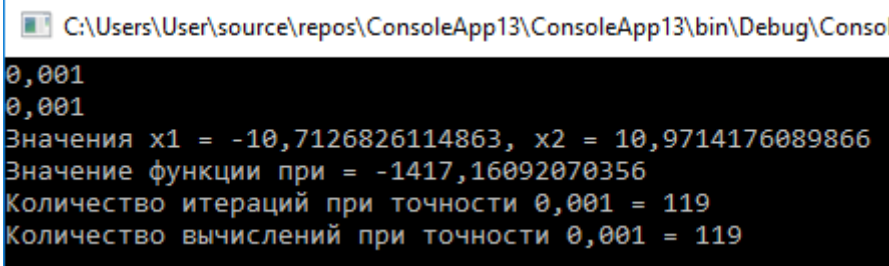
namespace ConsoleApp12
{
    class Program
    {
        static double f(double X1, double X2)
        {
            double fx;
            fx = 194 * Math.Pow(X1, 2) + 376 * X1 * X2 + 194 * Math.Pow(X2, 2) + 31 * X1 - 229 * X2 + 4;
            return fx;
        }
        static double grad1(double X1, double X2)
        {
            double gradX1;
            gradX1 = 388 * X1 + 376 * X2 + 31;
            return gradX1;
        }
        static double grad2(double X1, double X2)
        {
            double gradX2;
            gradX2 = 376 * X1 + 388 * X2 - 229;
            return gradX2;
        }
        static void Main(string[] args)
        {
            double x1, x2;
```

```

x1 = 0;
x2 = 0;
double x1pred, x2pred;
double M = 400;
double eps1 = Convert.ToDouble(Console.ReadLine());
double eps2 = Convert.ToDouble(Console.ReadLine());
int k = 0;
double tk = 0.01;
while(true)
{
    if (Math.Sqrt(Math.Pow(grad1(x1, x2), 2) + Math.Pow(grad2(x1, x2), 2)) < eps1)
    {
        Console.WriteLine($"{x1}, {x2}, {f(x1, x2)}, {k}");
        break;
    }
    if (k >= M)
    {
        Console.WriteLine($"{x1}, {x2}, {f(x1, x2)}, {k}");
        break;
    }
    x1pred = x1;
    x2pred = x2;
    x1 = x1 - tk * grad1(x1, x2);
    x2 = x2 - tk * grad2(x1, x2);
    if (f(x1, x2) - f(x1pred, x2pred) < 0)
    {
        if (Math.Sqrt(Math.Pow((x1 - x1pred), 2) + Math.Pow((x2 - x2pred), 2)) < eps1 &&
Math.Abs(f(x1, x2) - f(x1pred, x2pred)) < eps2)
        {
            Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
            Console.WriteLine($"Значение функции при = {f(x1, x2)}");
            Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
            Console.WriteLine($"Количество вычислений при точности {eps1} = {k}");
            break;
        }
    }
    else
    {
        tk = tk / 2;
    }
    k++;
}
Console.ReadKey(true);
Console.ReadKey();
Console.ReadLine();
}
}
}

```

Полученное решение имеет следующий вид:



```

C:\Users\User\source\repos\ConsoleApp13\ConsoleApp13\bin\Debug\ConsoleApp13.exe
0,001
0,001
Значения x1 = -10,7126826114863, x2 = 10,9714176089866
Значение функции при = -1417,16092070356
Количество итераций при точности 0,001 = 119
Количество вычислений при точности 0,001 = 119

```

Метод сопряженных градиентов

Реализация кода в C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp15

```

```

{
    class Program
    {
        static double f(double X1, double X2)
        {
            double fx;
            fx = 194 * Math.Pow(X1, 2) + 376 * X1 * X2 + 194 * Math.Pow(X2, 2) + 31 * X1 - 229 * X2 + 4;
            return fx;
        }
        static double grad1(double X1, double X2)
        {
            double gradX1;
            gradX1 = 388 * X1 + 376 * X2 + 31;
            return gradX1;
        }
        static double grad2(double X1, double X2)
        {
            double gradX2;
            gradX2 = 376 * X1 + 388 * X2 - 229;
            return gradX2;
        }
        static void Main(string[] args)
        {
            double t = 0;
            double x1, x2;
            x1 = 0;
            x2 = 0;
            double x1pred, x2pred;
            int M = 10000;
            int j = 0;
            int n = 2;
            double eps1 = Convert.ToDouble(Console.ReadLine());
            double eps2 = Convert.ToDouble(Console.ReadLine());
            int k = 0;
            double tk = 0.01;
            while (true)
            {
                if (j >= M)
                {
                    Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
                    Console.WriteLine($"Значение функции при = {f(x1, x2)}");
                    Console.WriteLine($"Количество итераций при точности {eps1} = {t}");
                    Console.WriteLine($"Количество вычислений при точности {eps1} = {j+t}");
                    break;
                }
                else
                {
                    k = 0;
                }
                if(k<=n-1)
                {
                    if (Math.Sqrt(Math.Pow(grad1(x1, x2), 2) + Math.Pow(grad2(x1, x2), 2)) < eps1)
                    {
                        Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
                        Console.WriteLine($"Значение функции при = {f(x1, x2)}");
                        Console.WriteLine($"Количество итераций при точности {eps1} = {t}");
                        Console.WriteLine($"Количество вычислений при точности {eps1} = {j + t}");
                        break;
                    }
                    else
                    {
                        x1pred = x1;
                        x2pred = x2;
                        x1 = x1 - tk * grad1(x1, x2);
                        x2 = x2 - tk * grad2(x1, x2);
                        if (f(x1, x2) - f(x1pred, x2pred) < 0)
                        {
                            if (Math.Sqrt(Math.Pow((x1 - x1pred), 2) + Math.Pow((x2 - x2pred), 2)) < eps1
                                && Math.Abs(f(x1, x2) - f(x1pred, x2pred)) < eps2)
                            {
                                Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
                                Console.WriteLine($"Значение функции при = {f(x1, x2)}");
                                Console.WriteLine($"Количество итераций при точности {eps1} = {t}");
                                Console.WriteLine($"Количество вычислений при точности {eps1} = {j + t}");
                                break;
                            }
                        }
                    }
                }
                else
            }
        }
    }
}

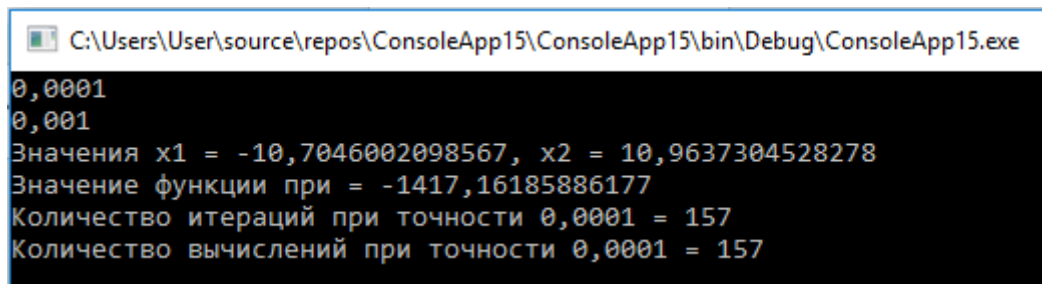
```

```

        {
            k++;
        }
    }
    else
    {
        tk = tk / 2;
    }
}
else if(k == n)
{
    j++;
}
t++;
}
Console.ReadKey(true);
Console.ReadKey();
Console.ReadLine();
}
}
}

```

Полученное решение имеет следующий вид:



```

C:\Users\User\source\repos\ConsoleApp15\ConsoleApp15\bin\Debug\ConsoleApp15.exe
0,0001
0,001
Значения x1 = -10,7046002098567, x2 = 10,9637304528278
Значение функции при = -1417,16185886177
Количество итераций при точности 0,0001 = 157
Количество вычислений при точности 0,0001 = 157

```

Метод наискорейшего спуска

Реализация кода в C#

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp16
{
    class Program
    {
        static double f(double X1, double X2)
        {
            double fx;
            fx = 194 * Math.Pow(X1, 2) + 376 * X1 * X2 + 194 * Math.Pow(X2, 2) + 31 * X1 - 229 * X2 + 4;
            return fx;
        }
        static double grad1(double X1, double X2)
        {
            double gradX1;
            gradX1 = 388 * X1 + 376 * X2 + 31;
            return gradX1;
        }
        static double grad2(double X1, double X2)
        {
            double gradX2;
            gradX2 = 376 * X1 + 388 * X2 - 229;
            return gradX2;
        }
        static double ff(double tk, double x1, double x2)
        {
            return f(x1 - tk * grad1(x1, x2), x2 - tk * grad2(x1, x2));
        }
        static double min(double eps, double a, double b, double x1, double x2)
        {

```

```

double x = 0;
double y1, y2, x11, x22;
while (Math.Abs(b - a) > eps)
{
    x11 = b - (b - a) / 1.618;
    x22 = a + (b - a) / 1.618;
    y1 = ff(x1, x2, x11);
    y2 = ff(x1, x2, x22);
    if (y1 >= y2)
    {
        a = x11;
    }
    else
    {
        b = x22;
    }
    x = (a + b) / 2;
}
return x;
}
static void Main(string[] args)
{
    double a = 0.2;
    double b = 0;
    double x1, x2;
    x1 = 0;
    x2 = 0;
    double x1pred, x2pred;
    double M = 20000;
    double eps1 = Convert.ToDouble(Console.ReadLine());
    double eps2 = Convert.ToDouble(Console.ReadLine());
    int k = 0;
    double tk = 0.1;
    while (true)
    {
        if (Math.Sqrt(Math.Pow(grad1(x1, x2), 2) + Math.Pow(grad2(x1, x2), 2)) < eps1)
        {
            Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
            Console.WriteLine($"Значение функции при = {f(x1, x2)}");
            Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
            Console.WriteLine($"Количество вычислений при точности {eps1} = {k}");
            break;
        }
        if (k >= M)
        {
            Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
            Console.WriteLine($"Значение функции при = {f(x1, x2)}");
            Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
            Console.WriteLine($"Количество вычислений при точности {eps1} = {k}");
            break;
        }
        tk = min(eps1, a, b, x1, x2);
        x1pred = x1;
        x2pred = x2;
        x1 = x1 - tk * grad1(x1, x2);
        x2 = x2 - tk * grad2(x1, x2);
        if (f(x1, x2) - f(x1pred, x2pred) < 0)
        {
            if (Math.Sqrt(Math.Pow((x1 - x1pred), 2) + Math.Pow((x2 - x2pred), 2)) < eps1 &&
Math.Abs(f(x1, x2) - f(x1pred, x2pred)) < eps2)
            {
                Console.WriteLine($"Значения x1 = {x1}, x2 = {x2}");
                Console.WriteLine($"Значение функции при = {f(x1, x2)}");
                Console.WriteLine($"Количество итераций при точности {eps1} = {k}");
                Console.WriteLine($"Количество вычислений при точности {eps1} = {k}");
                break;
            }
        }
        k++;
    }
    Console.ReadKey(true);
    Console.ReadKey();
    Console.ReadLine();
}
}

```

```
}
```

Полученное решение имеет следующий вид:

```
C:\Users\User\source\repos\ConsoleApp16\ConsoleApp16\bin\Debug\ConsoleApp16.exe
0,01
0,01
Значения x1 = -10,7532400945095, x2 = 11,0092072285323
Значение функции при = -1417,13239562251
Количество итераций при точности 0,01 = 73
Количество вычислений при точности 0,01 = 73
```

Метод Нелдера-Мида

Реализация кода в C#

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp17
{
    class Program
    {
        static double f(double X1, double X2)
        {
            double fx;
            fx = 194 * Math.Pow(X1, 2) + 376 * X1 * X2 + 194 * Math.Pow(X2, 2) + 31 * X1 - 229 * X2 + 4;
            return fx;
        }
        static void Main(string[] args)
        {
            double [] x1 = new double[3];
            double [] x2 = new double[3];
            double midx1, midx2;
            double x1r, x2r, x1e, x2e, x1s, x2s;
            x1[0] = -10;
            x2[0] = 10;
            x1[2] = 20;
            x2[2] = 20;
            x1[1] = 30;
            x2[1] = 30;
            int koorbest = 0;
            int koorworst = 0;
            int koorgood = 0;
            double alpha, beta, gamma;
            alpha = 1;
            beta = 0.5;
            gamma = 2;
            double eps = Convert.ToDouble(Console.ReadLine());
            double k = 0;
            double tempx1;
            double tempx2;
            double best, worst, good;
            best = 0;
            worst = 0;
            while (true)
            {
                if (f(x1[0], x2[0]) < f(x1[1], x2[1]))
                {
                    if (f(x1[0], x2[0]) < f(x1[2], x2[2]))
                    {
                        best = f(x1[0], x2[0]);
                        koorbest = 0;
                    }
                    else
                    {
                        best = f(x1[2], x2[2]);

```

```

        koorbest = 2;
    }
}
else
{
    if (f(x1[1], x2[1]) < f(x1[2], x2[2]))
    {
        best = f(x1[1], x2[1]);
        koorbest = 1;
    }
    else
    {
        best = f(x1[2], x2[2]);
        koorbest = 2;
    }
}

if (f(x1[0], x2[0]) > f(x1[1], x2[1]))
{
    if (f(x1[0], x2[0]) > f(x1[2], x2[2]))
    {
        worst = f(x1[0], x2[0]);
        koorworst = 0;
    }
    else
    {
        worst = f(x1[2], x2[2]);
        koorworst = 2;
    }
}
else
{
    if (f(x1[1], x2[1]) > f(x1[2], x2[2]))
    {
        worst = f(x1[1], x2[1]);
        koorworst = 1;
    }
    else
    {
        worst = f(x1[2], x2[2]);
        koorworst = 2;
    }
}

for (int i = 0; i < 3; i++)
{
    if (i != koorbest && i != koorworst)
    {
        good = f(x1[i], x2[i]);
        koorgood = i;
    }
}

midx1 = (x1[koorgood] + x1[koorbest]) / 2;
midx2 = (x2[koorgood] + x2[koorbest]) / 2;
x1r = (1 + alpha) * midx1 - alpha * x1[koorworst];
x2r = (1 + alpha) * midx2 - alpha * x2[koorworst];
if (f(x1r, x2r) < f(x1[koorbest], x2[koorbest]))
{
    x1e = (1 - gamma) * midx1 + gamma * x1r;
    x2e = (1 - gamma) * midx2 + gamma * x2r;
    if (f(x1e, x2e) < f(x1r, x2r))
    {
        x1[koorworst] = x1e;
        x2[koorworst] = x2e;
    }
    else
    {
        x1[koorworst] = x1r;
        x2[koorworst] = x2r;
    }
}
if (f(x1[koorbest], x2[koorbest]) < f(x1r, x2r) && f(x1r, x2r) < f(x1[koorgood],
x2[koorgood]))
{
    x1[koorworst] = x1r;
    x2[koorworst] = x2r;
}

```

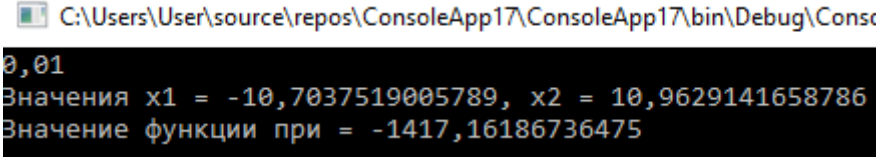


```

    }
    if (f(x1[koorgood], x2[koorgood]) < f(x1r, x2r) && f(x1r, x2r) <
f(x1[koorworst], x2[koorworst]))
    {
        temp1 = x1r;
        temp2 = x2r;
        x1r = x1[koorworst];
        x2r = x2[koorworst];
        x1[koorworst] = temp1;
        x2[koorworst] = temp2;
    }
    x1s = (1 - beta) * x1[koorworst] + beta * midx1;
    x2s = (1 - beta) * x2[koorworst] + beta * midx2;
    if (f(x1s, x2s) < f(x1[koorworst], x2[koorworst]))
    {
        x1[koorworst] = x1s;
        x2[koorworst] = x2s;
    }
    else
    {
        for (int i = 0; i < 3; i++)
        {
            x1[i] = x1[koorbest] + (x1[i] - x1[koorbest])/2;
            x2[i] = x2[koorbest] + (x2[i] - x2[koorbest])/2;
        }
    }
    if (Math.Round(Math.Sqrt(Math.Pow((x1[koorbest] - x1[koorgood]), 2) + Math.Pow((x2[koorbest] -
x2[koorgood]), 2)), 10, MidpointRounding.ToEven) <= eps && Math.Round(Math.Abs(f(x1[koorbest],
x2[koorbest]) - f(x1[koorgood], x2[koorgood])), 10, MidpointRounding.ToEven) < eps &&
Math.Round(Math.Sqrt(Math.Pow((x1[koorgood] - x1[koorworst]), 2) + Math.Pow((x2[koorgood] -
x2[koorworst]), 2)), 10, MidpointRounding.ToEven) <= eps && Math.Round(Math.Sqrt(Math.Pow((x1[koorbest] -
x1[koorworst]), 2) + Math.Pow((x2[koorbest] - x2[koorworst]), 2)), 10, MidpointRounding.ToEven) <= eps)
    {
        Console.WriteLine($"Значения x1 = {x1[koorbest]}, x2 = {x2[koorbest]}");
        Console.WriteLine($"Значение функции при = {best}");
        Console.WriteLine($"Значение функции при = {k}");
        Console.WriteLine($"Количество итераций при точности {eps} = {k}");
        Console.WriteLine($"Количество вычислений при точности {eps} = {3*k}");
        break;
    }
    k++;
}
Console.ReadKey(true);
Console.ReadKey();
Console.ReadLine();
}
}
}

```

Полученное решение имеет следующий вид:



```

C:\Users\User\source\repos\ConsoleApp17\ConsoleApp17\bin\Debug\Consc
0,01
Значения x1 = -10,7037519005789, x2 = 10,9629141658786
Значение функции при = -1417,16186736475

```

Сравнение методов

Проведём сравнение методов. Для этого найдём зависимость количества вычислений функции от точности решения.

```

x = -6:0.00001:-4;
y = x.*sin(x)+2*cos(x);
min(y)
MGS = [43, 81, 119, 157];
MH = [53, 90, 130, 170];
MNEWT = [152, 73, 198, 1451];
eps = [0.1, 0.01, 0.001, 0.0001];
hold on

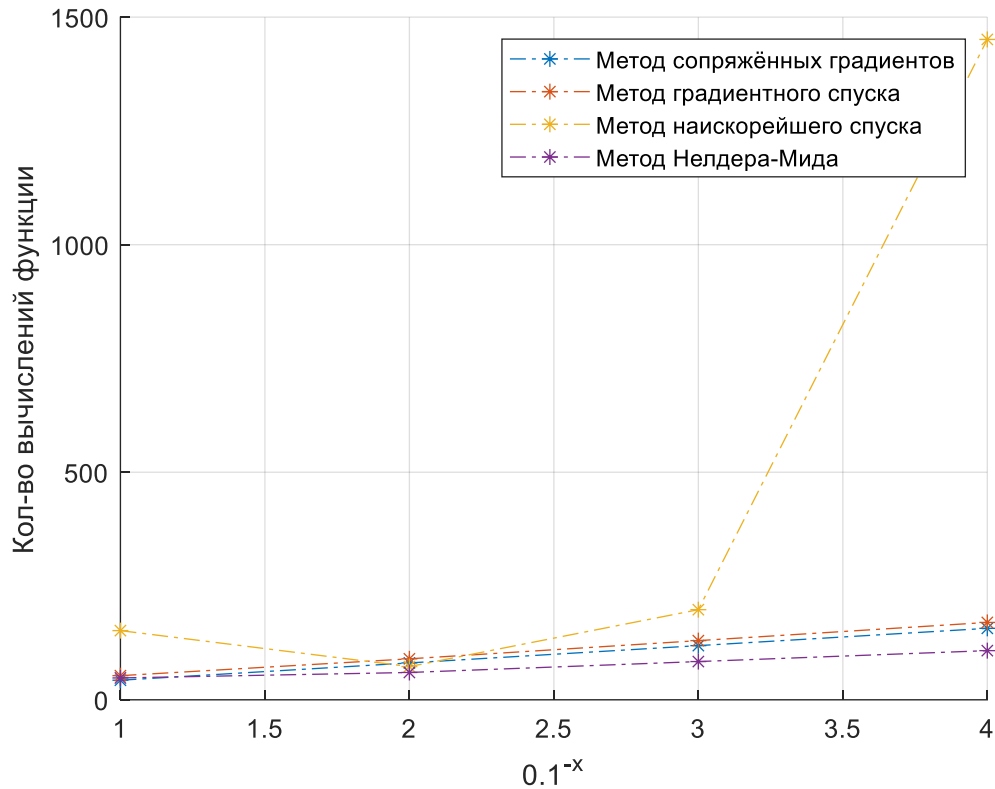
```

```

grid on
plot(abs(log10(eps)), MDP, '-.*');
plot(abs(log10(eps)), MH, '-.*');
plot(abs(log10(eps)), MNEWT, '-.*');
xlabel('0.1^-x');
ylabel('Кол-во вычислений функции');
legend({'Метод сопряжённых градиентов', 'Метод градиентного спуска', 'Метод наискорейшего спуска'});
hold off

```

Построим графики зависимостей



Сравнение методов

Из рисунка видно, что метод сопряжённых градиентов даёт приемлемый результат при меньшем количестве вычислений функции.