Министерство образования и науки Российской Федерации

Нижегородский Государственный Университет им. Н.И. Лобачевского

Факультет вычислительной математики и кибернетики

Кафедра численного и функционального анализа

Отчёт о лабораторной работе

«Численная реализация метода штрафов»

Выполнил: студент 4 курса

Группы 8409 Новиков Е.А.

Проверил: преподаватель, к.ф.-м.н.

Губина Е.В.

Содержание

|  |  |
| --- | --- |
| Постановка задачи …………………………………………………………… | 3 стр. |
| Метод внешнего штрафа …………………………………………………….. | 4 стр. |
| Алгоритм метода штрафов …………………………………………………... | 6 стр. |
| Реализация алгоритма ………………………………………………………... | 7 стр. |
| Руководство пользователя …………………………………………………... | 10 стр. |

# Постановка задачи

Содержание математического программирования составляют теория и методы решения задач о нахождении экстремумов функций на множествах, определяемых линейными и нелинейными ограничениями (равенствами и неравенствами).

В общем случае задача математического программирования представима в виде:



****(1)**

где *E* - множество простой геометрической структуры, например, *E* = *Rn* или

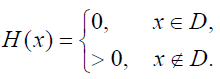
**

Метод внешнего штрафа же строит последовательность вспомогательных оптимизационных задач вида **(1)**. Минимизируемые функции этих задач зависят от настраиваемых параметров, а решения исходной задачи можно получить как предельные точки последовательностей решения вспомогательных задач.

# Необходимо написать программу, реализующую алгоритм метода внешнего штрафа, выбор настраиваемых параметров и выводящую наглядные результаты на экран.

# Метод внешнего штрафа

Функция *H*(*x*), определенная на множестве *x*∈*E* , называется функцией штрафа в задаче **(1)**, если



Одним из конкретных видов функции штрафа является степенной штраф:



где *c*1,…, *cN* ; *C*1,…,*CM* - положительные нормировочные коэффициенты, а

показатель степени *p* > 0 .

Заметим, что при достаточной гладкости функций *gi* и *hj* порядок гладкости функции штрафа определяется значением *p* следующим образом:

1. при 0 < *p* ≤ 1 негладкий штраф;
2. при 1 < *p* ≤ 2 гладкость до первого порядка;
3. при 2 < *p* ≤ 3 гладкость до второго порядка.

Построим вспомогательную функцию для задачи со штрафом

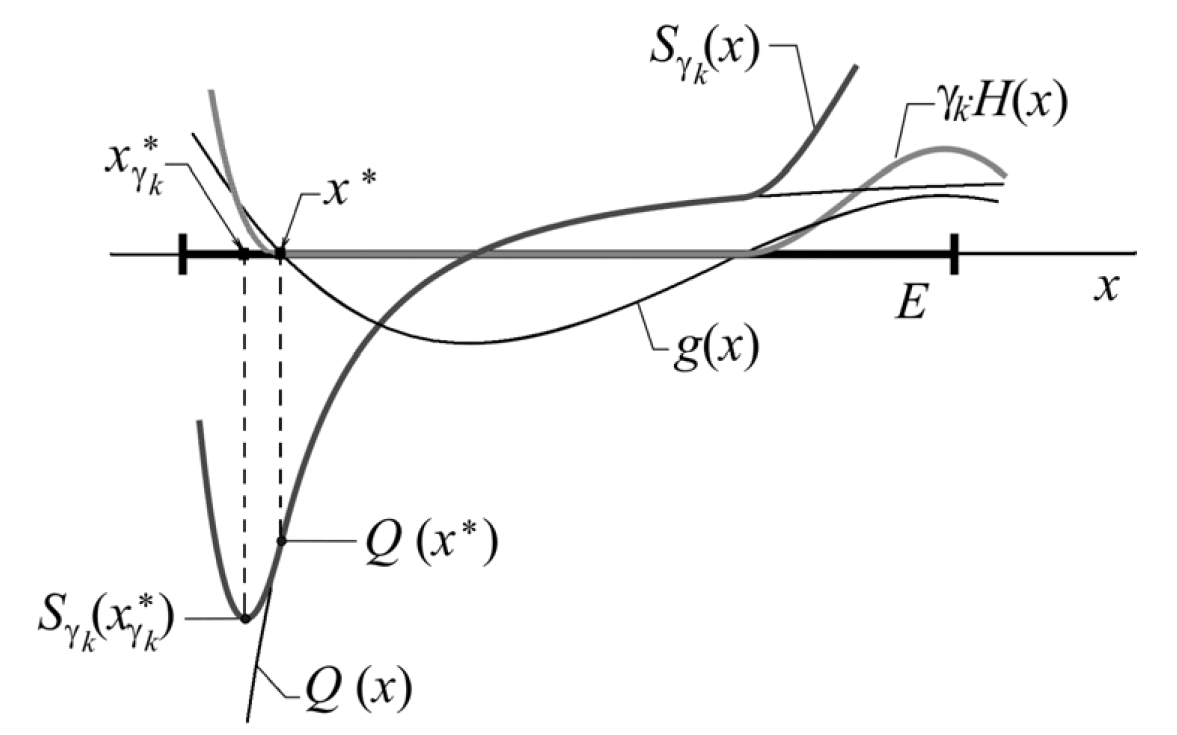
**

где γ > 0 – коэффициент штрафа. Заметим, что в допустимой области *x*∈*D* функция *S*γ(*x*) = *Q*(*x*) , а вне этой области *S*γ(*x*) > *Q*(*x*) за счет присутствия положительной штрафной добавки за нарушение ограничений.

Вспомогательные задачи со штрафом будут иметь вид:



где выбирается γ*k* →∞ при *k* →∞. Предполагается, что при больших γk решения *x*\*γ*k* этих задач существуют и не покидают некоторой ограниченной и замкнутой области. Если *E* - компактно, то последнее предположение будет всегда выполнено. Можно доказать, что при непрерывности *Q* и *H* все предельные точки последовательности *x*\*γ*k* будут являться решениями исходной задачи **(1)**. Структуру задачи со штрафом при некотором коэффициенте штрафа γ*k* иллюстрирует рис. 1.



*Рис. 1. Пример поведения функции задачи со штрафом при гладкой функции штрафа*

Заметим, что при гладкости штрафа в случае положения *x*\* (решения исходной задачи) на границе допустимой области *D* сходящаяся к *x*\* подпосследовательность *x*\*γ*kS* решений вспомогательных задач со штрафом в общем случае будет приближаться к *x*\*, оставаясь вне области *D*. Сходимость обеспечивается лишь при γ*k* →∞.

**Алгоритм метода штрафов**

1. Задать δ > 0 – параметр останова, γ0 – достаточно большое начальное значение коэффициента штрафа; выбрать стратегию построения последовательности γ*k*, обеспечивающую стремление γ*k* к бесконечности при *k* →∞. Выбрать степень в штрафа *p* , *p* >1, и принять *k* = 0 .
2. Найти решение *x*\*γ*k* в задаче со штрафом **(1)**, используя один из методов решения задач без функциональных ограничений с использованием информации о решении *x*\*γ*k-1* предыдущей задачи.
3. Проверить критерий останова по малости невязки **

Если *G*(*x*\*γ*k*) ≤ δ, остановить вычисления и принять *x*\*γ*k* в качестве оценки решения; иначе, при *G*(*x*\*γ*k*) > δ, перейти на пункт (d).

1. Выбрать, согласно принятой стратегии, обеспечивающей неограниченное увеличение γ*k*, значение γ*k+1* > γ*k*, положить *k* := *k* +1 и вернуться к пункту (b).

# Реализация алгоритма

В качестве метода безусловной локальной оптимизации мной был выбран метод Хука-Дживса.

Программа написана с использованием технологии Windows Presentation Foundation, платформы .NET Framework 4.0 на языке C#.

Для поиска решения методом штрафов был создан класс PenaltyMethod:

public class PenaltyMethod

{

private GeneralFunction GeneralFunction;

public PenaltyMethod(GeneralFunction generalFunction)

{

this.GeneralFunction = generalFunction;

}

/// <summary> Выполнить поиск </summary>

public List<Point> Perform(Point startPoint, double Sigma\_Discrepancy, double Sigma\_HookJeevs, double h0, double Gamma\_0)

{

List<Point> AlgorithmSteps = new List<Point>();

List<Point> Point\_list = new List<Point>();

Point z;

int k = 1;

int i = 0;

double h = h0;

for (; true; k++)

{

h = h0;

i = 0;

double Gamma\_k = Gamma\_0 \* k;

AlgorithmSteps.Clear();

Point\_list.Clear();

Point\_list.Add(z = startPoint);

while (true)

{

Point\_list.Add(GetConfiguration(z, h, Gamma\_k));

if (this.GeneralFunction.Value(Point\_list[i + 1], Gamma\_k)

< this.GeneralFunction.Value(Point\_list[i], Gamma\_k))

{

i++;

z = new Point()

{

X = Point\_list[i].X + 2 \* (Point\_list[i].X –

 Point\_list[i - 1].X),

Y = Point\_list[i].Y + 2 \* (Point\_list[i].Y –

 Point\_list[i - 1].Y)

};

}

else if (h < Sigma\_HookJeevs)

break;

else

{

if (i == 0)

h /= 2;

Point lastPoint = Point\_list[i];

AlgorithmSteps.AddRange(Point\_list.Take(i));

Point\_list.Clear();

Point\_list.Add(lastPoint);

i = 0;

z = lastPoint;

}

}

AlgorithmSteps.Add(Point\_list.Last());

if (this.Discrepancy(Point\_list[i]) < Sigma\_Discrepancy) break;

}

return AlgorithmSteps;

}

/// <summary> Возвращает конфигурацию </summary>

private Point GetConfiguration(Point z, double h, double Gamma\_k)

{

Point Result = z;

Point a = new Point(Result.X + h, Result.Y);

Point b = new Point(Result.X - h, Result.Y);

if (this.GeneralFunction.Value(a, Gamma\_k) < this.GeneralFunction.Value(Result, Gamma\_k))

Result = a;

else if (this.GeneralFunction.Value(b, Gamma\_k) < this.GeneralFunction.Value(Result, Gamma\_k))

Result = b;

Point c = new Point(Result.X, Result.Y + h);

Point d = new Point(Result.X, Result.Y - h);

if (this.GeneralFunction.Value(c, Gamma\_k) < this.GeneralFunction.Value(Result, Gamma\_k))

Result = c;

else if (this.GeneralFunction.Value(d, Gamma\_k)

< this.GeneralFunction.Value(Result, Gamma\_k))

Result = d;

return Result;

}

/// <summary> Невязка </summary>

private double Discrepancy(Point p)

{

double Result = 0.0;

foreach (ConditionFunction cond in this.GeneralFunction.GetConditionFunction())

Result = Math.Max(Result, cond.Value(p));

return Result;

}

}

Были также созданы классы Function, ConditionFunction, ObjectiveFunction, OwnFunction, PenaltyFunction, GeneralFunction со следующей схемой наследования:

Function

(абстрактный класс для представления функции)

ObjectiveFunction

(класс для представления целевой функции)

ConditionFunction

(класс для представления ограничений)

(наследуемые от ObjectiveFunction классы, представляющие стандартные функции)

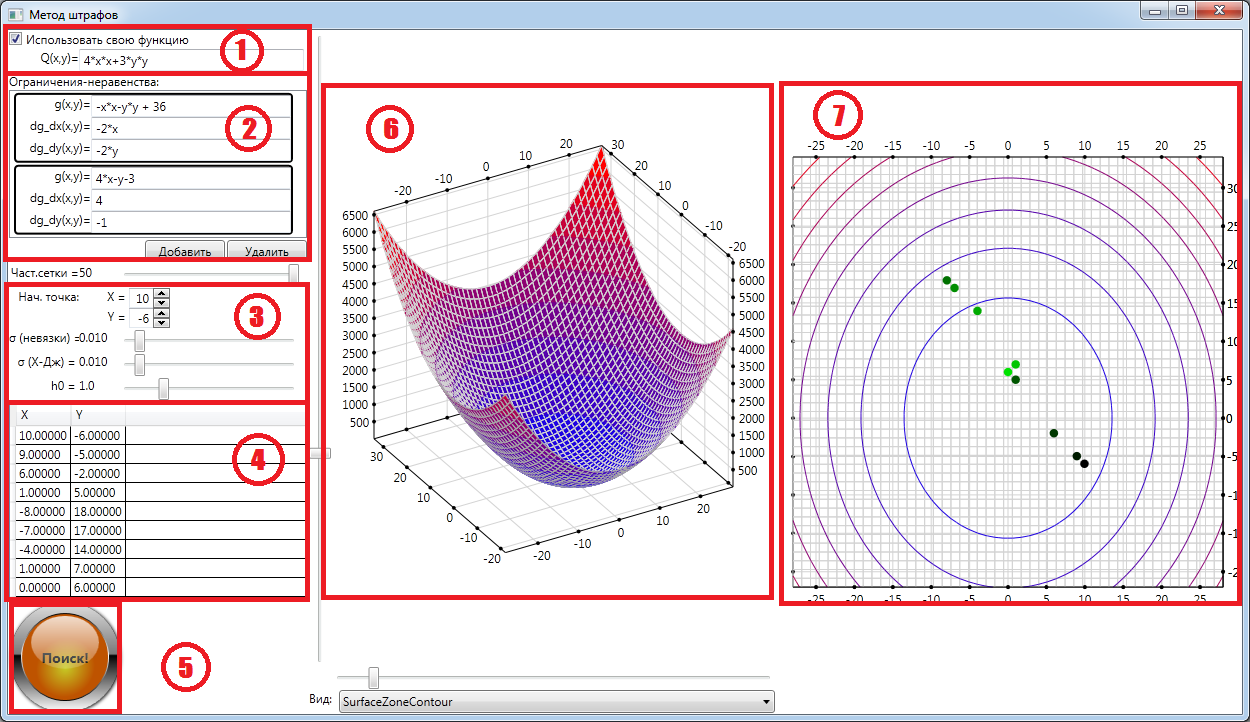
OwnFunction

(класс для представления пользовательской целевой функции)

А также класс GeneralFunction содержит поля типа ObjectiveFunction и PenaltyFunction (функция штрафа).

# Руководство пользователя

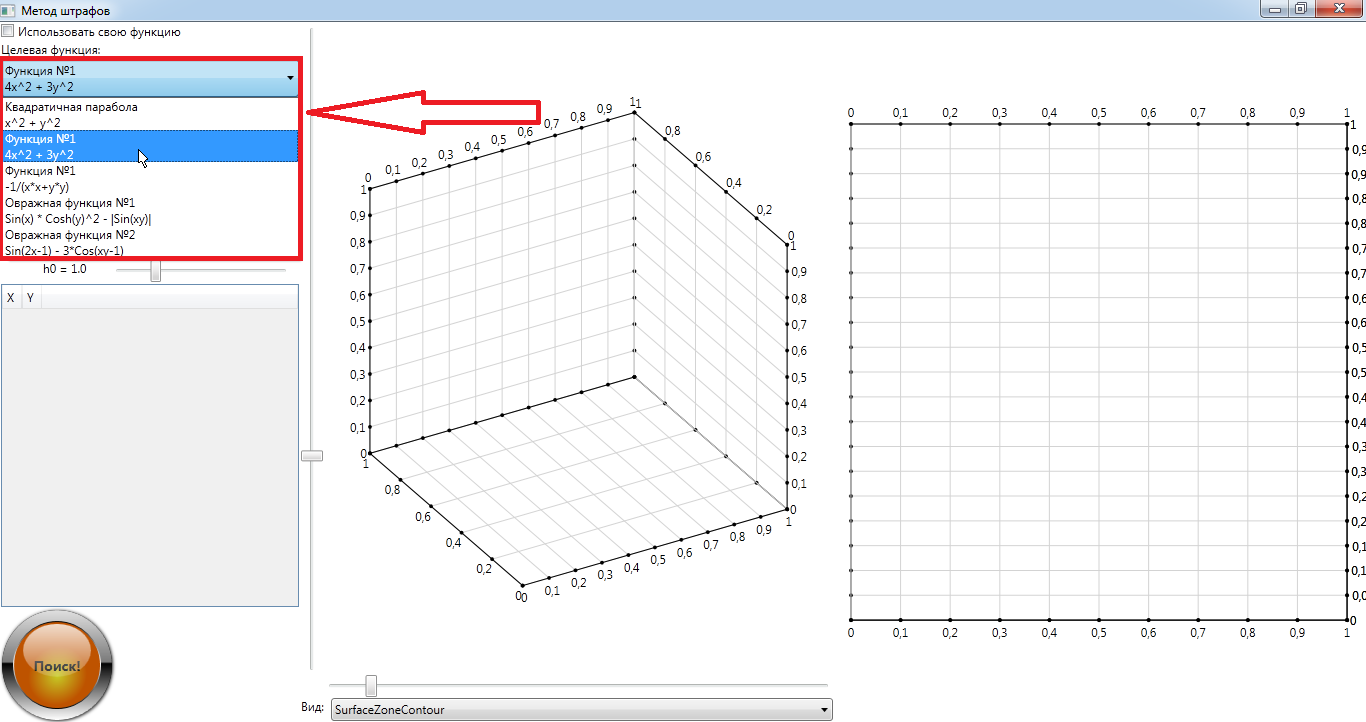
Интерфейс программы состоит из следующих частей:



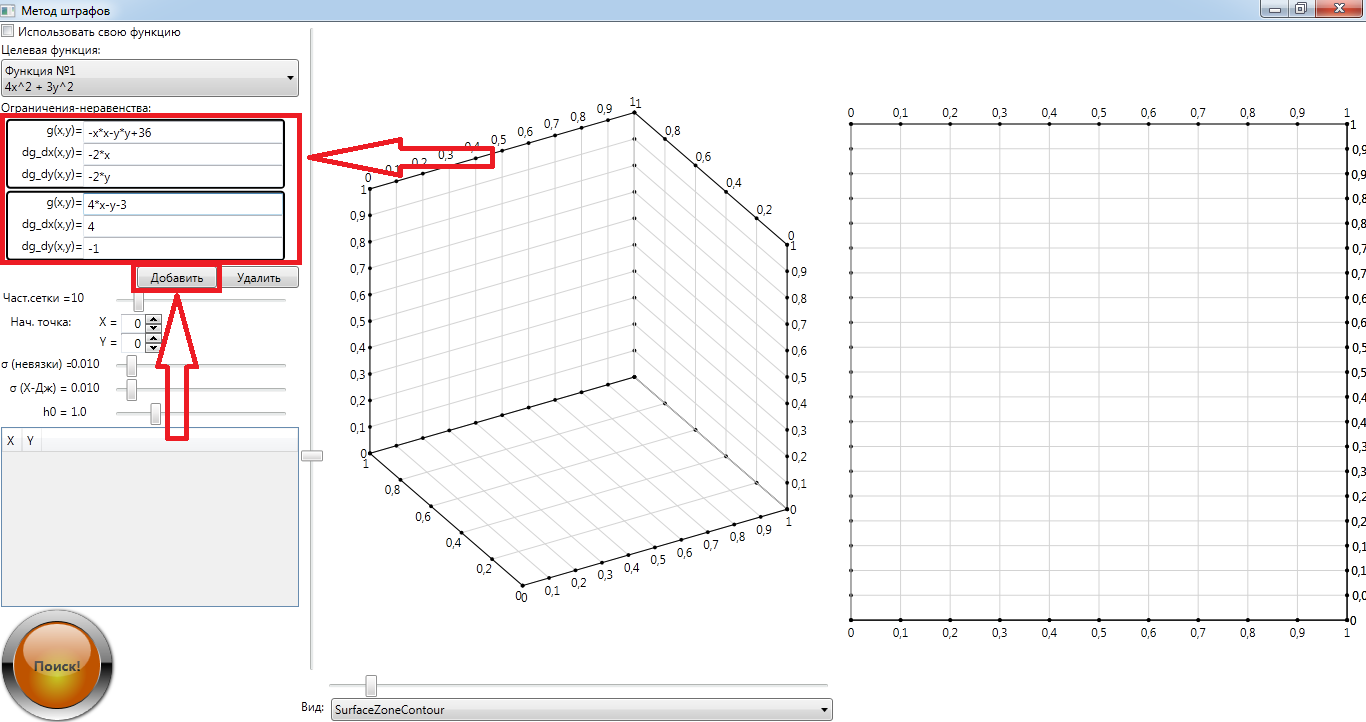
1. Поле для ввода функции
2. Список для ввода ограничений
3. Выбор условий поиска
4. Список точек, просмотренных на каждом шаге
5. Кнопка «Поиск!»
6. Трёхмерное изображение целевой функции
7. Двумерное изображение целевой функции с визуализацией шагов поиска

Для того, чтобы произвести поиск минимума, необходимо произвести следующие действия:

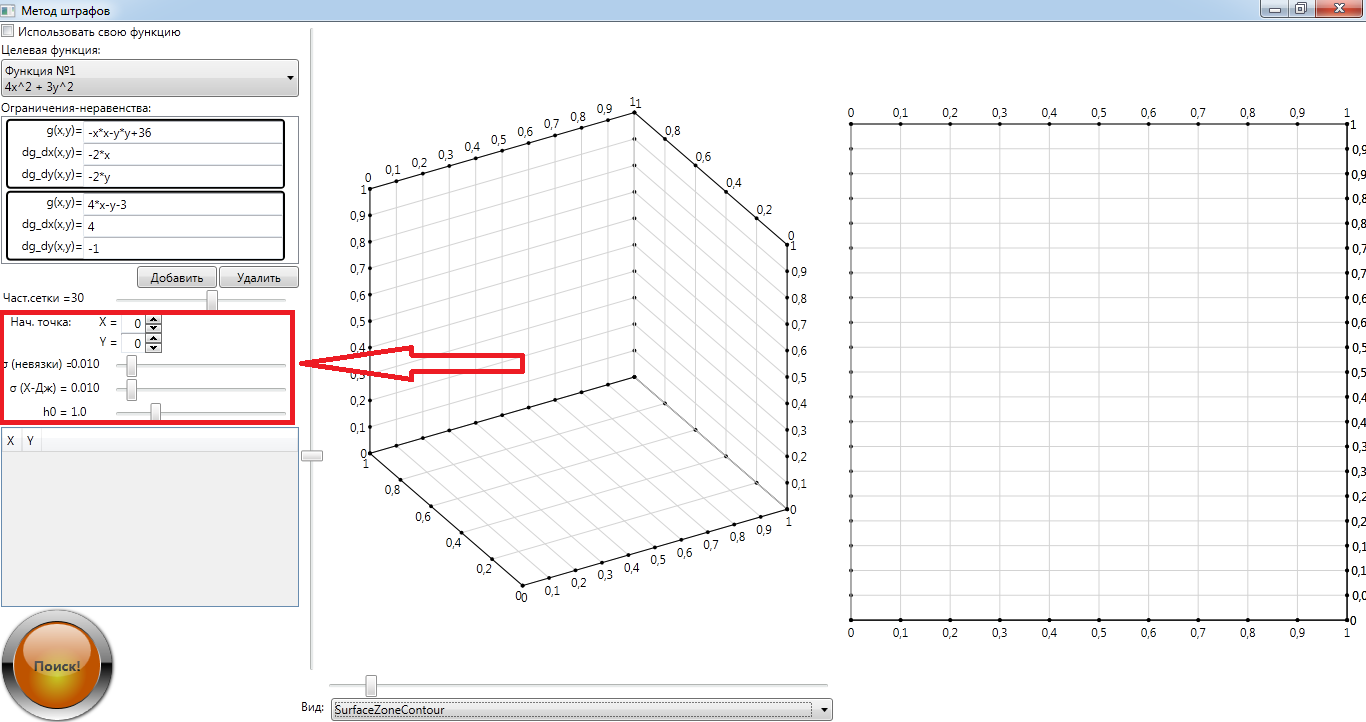
1. Ввести свою целевую функцию или выбрать функцию из списка:



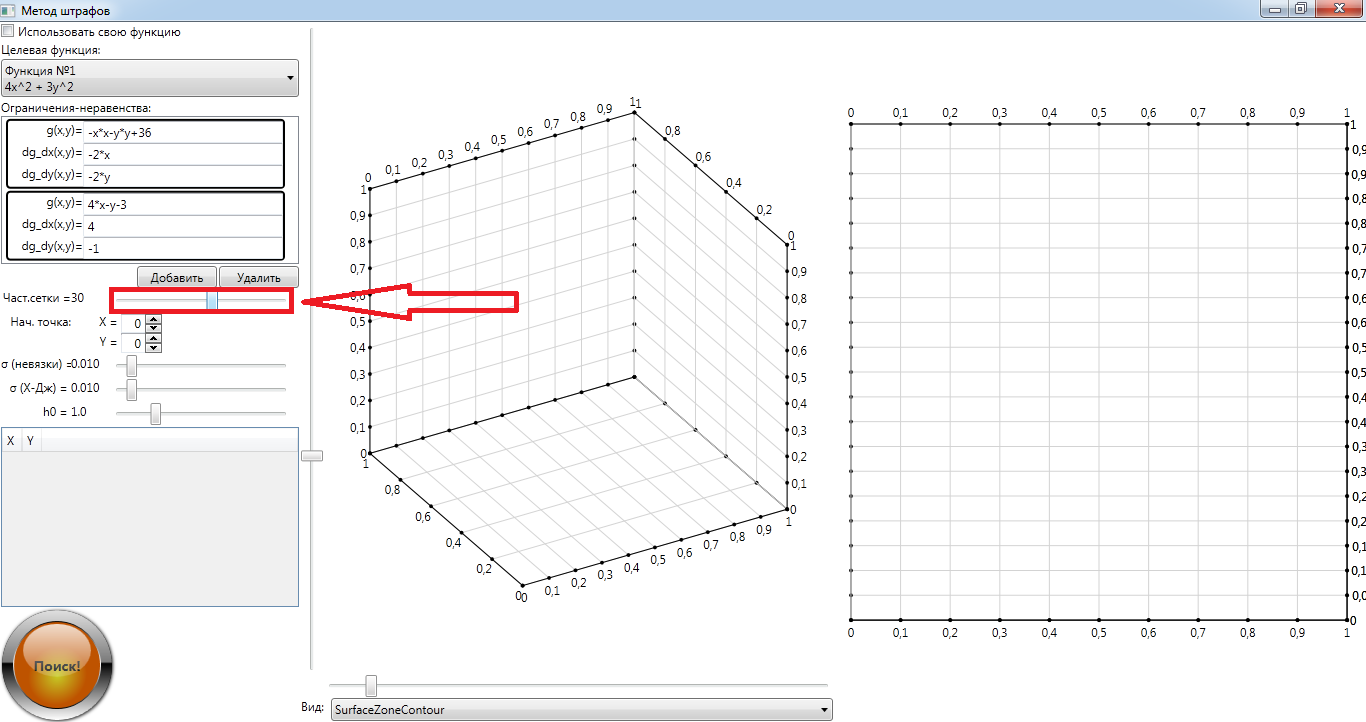
1. Затем нужно задать в списке необходимое количество ограничений:



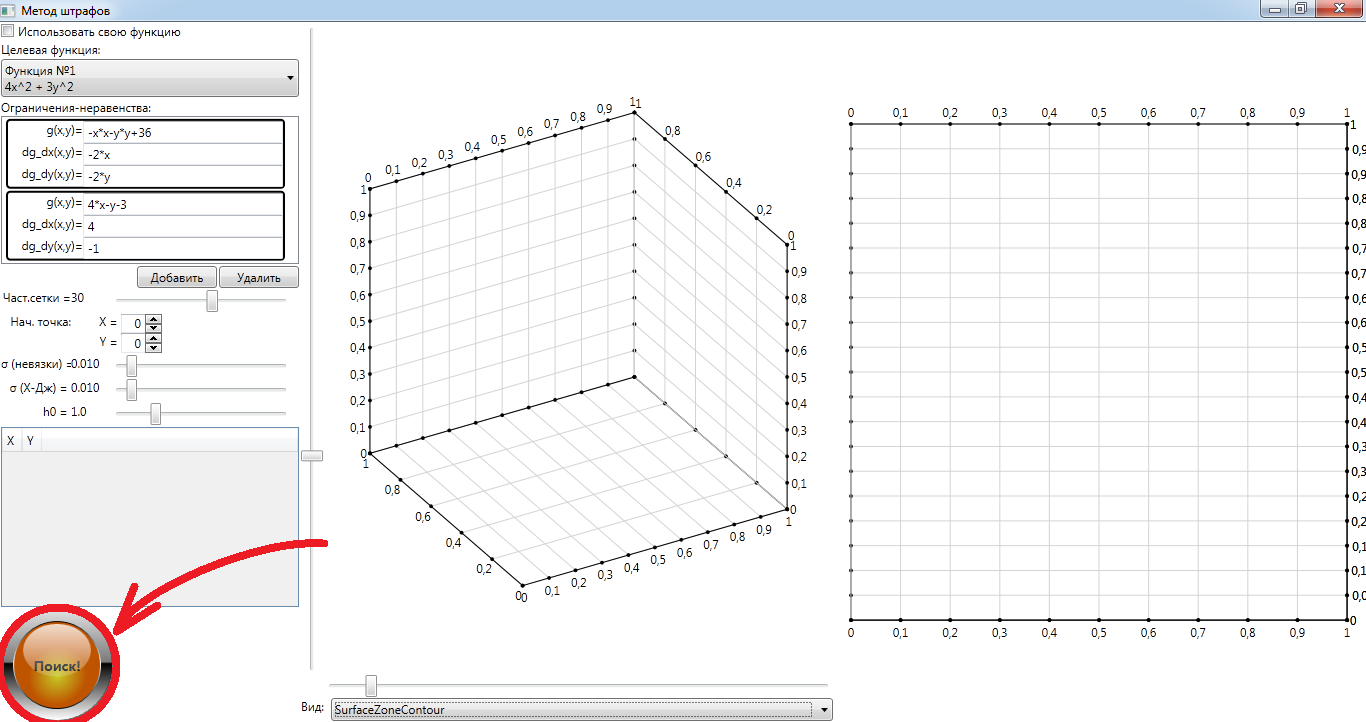
1. Далее необходимо выбрать условия поиска: начальную точку, критерий останова метода штрафов (σ-невязки), критерий останова метода Хука-Дживса (σ Х-Дж), а также начальную длину шага.



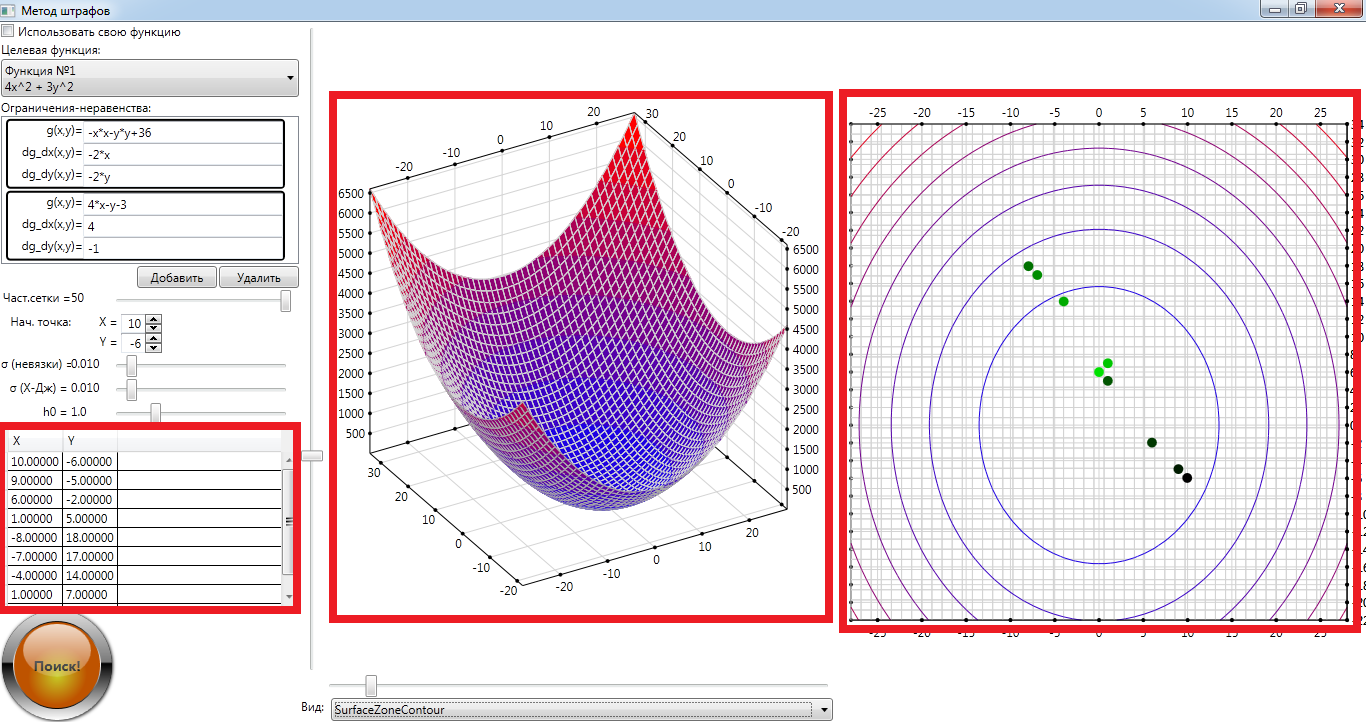
1. Затем необходимо задать частоту сетки графиков. От этого будет зависеть, насколько точно отобразится целевая функция. Но следует помнить, что, задавая слишком большое значение частоты сетки, программа будет «думать» значительное время!



1. Для начала вычисления нажимаем кнопку «Поиск!»



1. Далее программа отображает результаты поиска. Слева можно увидеть список точек, полученных на каждом шаге методом Хука-Дживса. По центру визуализируется целевая функция. А справа – она же, но с шагами поиска, отмеченных точками. Стартовая точка поиска отображается черным цветом. С каждым следующим шагом точка становится более зелёной. Точка минимума целевой функции с ограничениями отображается в центре правого графика зелёным цветом.



Трёхмерную визуализацию также можно поворачивать для более подробного её изучения.