

ГУАП

КАФЕДРА № 44

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

\_\_\_\_\_  
доцент, к.т.н., доцент  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
А.А.Сенцов  
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

УСЛОВНАЯ ОПТИМИЗАЦИЯ

по курсу: МЕТОДЫ ОПТИМИЗАЦИИ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4242М

\_\_\_\_\_  
подпись, дата

\_\_\_\_\_  
М.В.Климов  
инициалы, фамилия

Санкт-Петербург 2023

## 1. Цель работы и вариант

Ознакомиться с известными аналитическими и численными методами условной оптимизации, получить опыт решения условной оптимизационной задачи, приобрести навыки использования математических программных средств для решения задач на условный экстремум.

Вариант 7.

Задача Ж. Дан угол  $45^\circ$  и точка  $A$  внутри него. Как нужно провести прямую через точку  $A$ , чтобы отсечь от угла треугольник наименьшего периметра (рис. 1)? Ответ: вневписанная в треугольник окружность должна касаться  $MN$  в точке  $A$ .

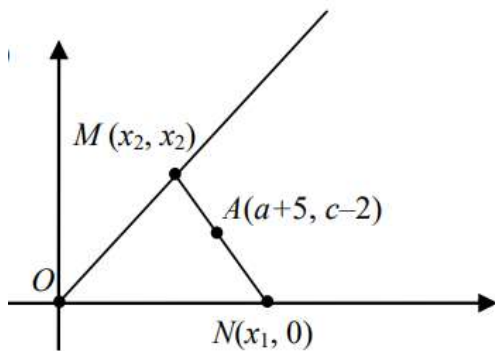


Рисунок 1. Поясняющий рисунок к задаче

Методом Штрафных функций.

Начальные данные:  $S = 7$ ,  $e = 0.001$ .

## 2. Вычисления:

Целевая функция к данной задаче имеет вид:

$$f(x_1, x_2) = x_1 + (x_2 * \sqrt{2}) + \sqrt{x_2^2 + (x_1 - x_2)^2} \rightarrow \min$$

Ограничивающая функция в данной задаче будет иметь вид:

$$q_1(x_1, x_2) = x_1 * x_2 \geq 0$$

$$g_1(x_1, x_2) = a + 5 + (c - 2) * \frac{x_1 - x_2}{x_2} - x_1 = 0$$

Метод штрафных функций так же, как и аналитические методы условной оптимизации, основан на переходе от задачи с ограничением к задаче без ограничений:

$$F(\mathbf{x}) = f(\mathbf{x}) + rP(\mathbf{x}) \rightarrow \min$$

Здесь  $r$  – некоторое число, называемое величиной штрафа.

$P(\mathbf{x})$  – функция штрафа (penalty function), удовлетворяющая условиям:

$$P(\mathbf{x}) = \begin{cases} 0, \text{ если } \mathbf{q}(\mathbf{x}) = 0, g_i(\mathbf{x}) \geq 0, i = \overline{1, s}; \\ > 0, \text{ если } \mathbf{q}(\mathbf{x}) \neq 0 \text{ или } \exists g_i(\mathbf{x}) < 0, i = \overline{1, s}. \end{cases}$$

Функция штрафа равна нулю, если все ограничения выполняются и больше нуля, если нарушено хотя бы одно из ограничений.

Наиболее простые способы формирования такой функции:

$$P(\mathbf{x}) = \sum_{j=1}^m q_j^2(\mathbf{x}) - \sum_{i=1}^s \min(0, g_i(\mathbf{x})),$$

$$P(\mathbf{x}) = \sum_{j=1}^m |q_j(\mathbf{x})| - \sum_{i=1}^s \min(0, g_i(\mathbf{x})).$$

Алгоритм поиска экстремума методом штрафных функций можно сформулировать в виде трех шагов.

Шаг 1. Выбрать произвольное начальное значение  $r$ . Обычно выбирают  $r = 1$ .

Шаг 2. Найти решение  $\mathbf{x}^*$  задачи каким-либо численным методом.

Шаг 3. Если  $P(\mathbf{x}^*) > \varepsilon$ , увеличить  $r$  и перейти к шагу 2. Иначе:  $\mathbf{x}^*$  – искомая точка минимума.

В качестве начальной точки при численном решении задачи безусловной оптимизации на шаге 2 рекомендуется выбирать точку  $\mathbf{x}^*$ , полученную на предыдущей итерации метода штрафных функций.

Программа по итогу должна проводить два алгоритма, один из которых это нахождение локального минимума при заданной начальной точке, а другой

– изменение начальной функции и перезапуск первого алгоритма при недостаточно точном выполнении заданных условий.

Как под алгоритм для метода Штрафных функций был выбран метод Градиентного спуска.

Защита от ввода некорректных данных не проводится, поскольку при некорректных данных пользователь получит лишь очень странный результат, тем не менее подходящий под введённые условия.

Код программы приведён в приложении 1, а результат выполнения представлен на рисунках 1-3.

```
Командная строка - python desktop/lab4.py
Microsoft Windows [Version 10.0.19045.2965]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\klima>python desktop/lab4.py
Введите a: -3
Введите c: 3
0)Результат функции равен 34.11718718502838 , при x1 = 9.999954 x2 = 9.989854578643763. Точность решения: 1.9985704742396404
1)Результат функции равен 34.09194078386205 , при x1 = 9.998710920217365 x2 = 9.97973509821886. Точность решения: 1.9968094847772697
2)Результат функции равен 34.06639556812045 , при x1 = 9.99751394897757 x2 = 9.96964105543431. Точность решения: 1.9947181719392315
3)Результат функции равен 34.04055057496879 , при x1 = 9.99595044944635 x2 = 9.95957195425834. Точность решения: 1.9922976331135308
4)Результат функции равен 34.01440521015182 , при x1 = 9.994021958447377 x2 = 9.949527305997186. Точность решения: 1.989549921627761
5)Результат функции равен 33.98795908704675 , при x1 = 9.991730185141611 x2 = 9.939506629370998. Точность решения: 1.986476845502899
6)Результат функции равен 33.96121206546907 , при x1 = 9.989077009569874 x2 = 9.929509450586767. Точность решения: 1.9830779668844389
7)Результат функции равен 33.93416424223399 , при x1 = 9.986064481060428 x2 = 9.919535303408265. Точность решения: 1.9793575965525108
8)Результат функции равен 33.90681594947692 , при x1 = 9.982694816503454 x2 = 9.90958372922292. Точность решения: 1.9753170003127565
9)Результат функции равен 33.87916775273679 , при x1 = 9.978970398494468 x2 = 9.899654277105606. Точность решения: 1.9709583892698337
10)Результат функции равен 33.85122044880613 , при x1 = 9.974893773348885 x2 = 9.88974650387929. Точность решения: 1.9662841219856073
11)Результат функции равен 33.82297508335248 , при x1 = 9.970467648990034 x2 = 9.879859974172506. Точность решения: 1.9612967017241427
12)Результат функции равен 33.794432848315886 , при x1 = 9.965694892713119 x2 = 9.869994260473602. Точность решения: 1.9559987743857992
13)Результат функции равен 33.76559527988766 , при x1 = 9.960578528827709 x2 = 9.860148943181755. Точность решения: 1.950393126332811
14)Результат функции равен 33.73646405147579 , при x1 = 9.955121736181486 x2 = 9.850323610654677. Точность решения: 1.9444826821088412
15)Результат функции равен 33.707041078462815 , при x1 = 9.949327845568112 x2 = 9.84051785925301. Точность решения: 1.9382705020551594
16)Результат функции равен 33.67732848676218 , при x1 = 9.943200337022178 x2 = 9.830731293381376. Точность решения: 1.931759779826141
17)Результат функции равен 33.64732861317951 , при x1 = 9.936742837004326 x2 = 9.820963525526038. Точность решения: 1.924953839806916
18)Результат функции равен 33.61704400078524 , при x1 = 9.929959115479745 x2 = 9.811214176289162. Точность решения: 1.9178561344361196
19)Результат функции равен 33.58647739490565 , при x1 = 9.922853082893353 x2 = 9.801482874419653. Точность решения: 1.9104702414367338
20)Результат функции равен 33.555631738939184 , при x1 = 9.915428787045053 x2 = 9.791769256840535. Точность решения: 1.9027998009581633
21)Результат функции равен 33.5245101700055 , при x1 = 9.907690409688591 x2 = 9.782072968672871. Точность решения: 1.8948488126327074
22)Результат функции равен 33.49311681443466 , при x1 = 9.89964226411759 x2 = 9.772393663256192. Точность решения: 1.886621032549746
23)Результат функции равен 33.46145278310428 , при x1 = 9.891288789962461 x2 = 9.762731002165438. Точность решения: 1.8781205701509656
24)Результат функции равен 33.429524166632156 , при x1 = 9.882634551501935 x2 = 9.753084655224379. Точность решения: 1.8693515850500688
25)Результат функции равен 33.39733403043279 , при x1 = 9.87368423319307 x2 = 9.743454300515529. Точность решения: 1.860318343780456
26)Результат функции равен 33.36488640964555 , при x1 = 9.864442636283627 x2 = 9.73383962438653. Точность решения: 1.8510252164744294
27)Результат функции равен 33.33218503942924 , при x1 = 9.854914674690816 x2 = 9.724240321453017. Точность решения: 1.8414766734775476
28)Результат функции равен 33.29923567222707 , при x1 = 9.8451053720104 x2 = 9.714656094597956. Точность решения: 1.8316772819017864
29)Результат функции равен 33.266041427223115 , при x1 = 9.835019856860287 x2 = 9.70508665496745. Точность решения: 1.821631702121211
30)Результат функции равен 33.23260742997764 , при x1 = 9.82466335936272 x2 = 9.69553172196304. Точность решения: 1.811344604213923
```

```

212)Результат функции равен 27.720405270182376 , при x1 = 8.006115767946032 x2 = 8.16528355622435. Точность решения: 0.025609082587832386
213)Результат функции равен 27.70256546804922 , при x1 = 8.005183927150751 x2 = 8.158327253311182. Точность решения: 0.023955339648120244
214)Результат функции равен 27.684824351359772 , при x1 = 8.004305070021676 x2 = 8.151388391359685. Точность решения: 0.02234902944015804
215)Результат функции равен 27.667180713091224 , при x1 = 8.003477948183406 x2 = 8.144466931845752. Точность решения: 0.02078896250057838
216)Результат функции равен 27.64963332795407 , при x1 = 8.002701332049059 x2 = 8.137562835467282. Точность решения: 0.01927404628119156
217)Результат функции равен 27.632180992568827 , при x1 = 8.001974010922268 x2 = 8.130676062171833. Точность решения: 0.017803205231011532
218)Результат функции равен 27.614822517627317 , при x1 = 8.001294793082764 x2 = 8.123806571184339. Точность решения: 0.016375300862405642
219)Результат функции равен 27.597556728039113 , при x1 = 8.000662505853294 x2 = 8.116954321834221. Точность решения: 0.01498953180187712
220)Результат функции равен 27.580382463063373 , при x1 = 8.00007599565013 x2 = 8.110110269582128. Точность решения: 0.013644633825986574
221)Результат функции равен 27.563298576426448 , при x1 = 7.999534128017133 x2 = 8.103381374046234. Точность решения: 0.012339679882876098
222)Результат функции равен 27.54630393642562 , при x1 = 7.999035787644006 x2 = 8.09650059102811. Точность решения: 0.01107368009992804
223)Результат функции равен 27.529397426019408 , при x1 = 7.998579878369302 x2 = 8.089716876538153. Точность решения: 0.009845661778810495
224)Результат функции равен 27.512577942904677 , при x1 = 7.998165323168705 x2 = 8.08295018602056. Точность решения: 0.008654668372829105
225)Результат функции равен 27.495844399588996 , при x1 = 7.9977918641291405 x2 = 8.076200474377849. Точность решения: 0.007499764463641787
226)Результат функции равен 27.47919572340261 , при x1 = 7.997456062409239 x2 = 8.069467695994934. Точность решения: 0.006380025711234083
227)Результат функции равен 27.4626308566184 , при x1 = 7.99715929818669 x2 = 8.062751804762716. Точность решения: 0.005294548801409782
228)Результат функции равен 27.446148756408106 , при x1 = 7.996899770593011 x2 = 8.05605275410122. Точность решения: 0.004242446331485188
229)Результат функции равен 27.429748394859324 , при x1 = 7.99667649763624 x2 = 8.049370496982268. Точность решения: 0.0032228479032365764
230)Результат функции равен 27.41342875905348 , при x1 = 7.996480516112082 x2 = 8.0427049895951664. Точность решения: 0.002234899936965462
231)Результат функции равен 27.39718885098131 , при x1 = 7.996334881503997 x2 = 8.03605617315092. Точность решения: 0.0012777652757200997
232)Результат функции равен 27.381027687568007 , при x1 = 7.9962146678727395 x2 = 8.0294240103385. Точность решения: 0.0003506236303385535
Минимум функции равен 27.381027687568007 , при x1 = 7.9962146678727395 x2 = 8.0294240103385. Точность решения: 0.0003506236303385535

```

Рисунок 1.

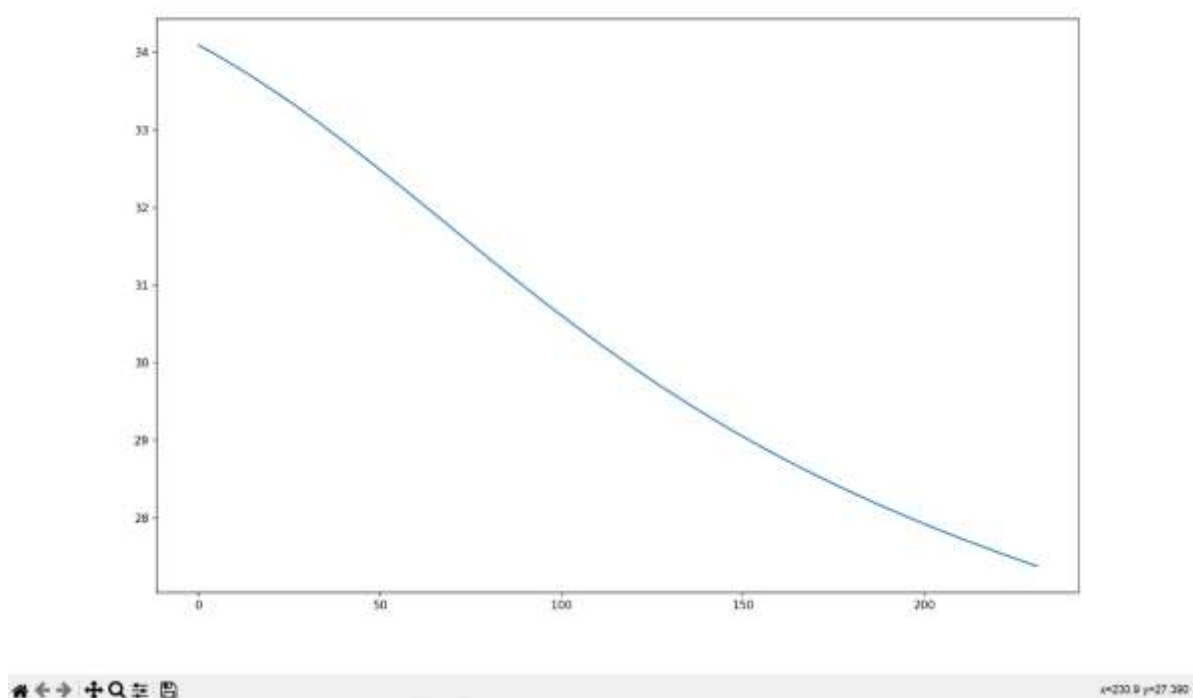


Рисунок 1-3. Результат выполнения программы в консоли.

### 3. Выводы.

В процессе выполнения лабораторной работы была решена поставленная задача по нахождению минимального периметра методом Штрафных функций. При тестировании возникали ошибки при нахождении исходной точки и начальной точки поиска в различных четвертях

координатной плоскости. В остальном, решение было получено в пределах менее 0.001 от предполагаемой точки при начальных данных  $a=-3$ ,  $c=3$  за 232 шага.

Также был проведён тест для  $a=7$ ,  $c=7$  (Рисунок 4), в котором достигнуть нужного результата удалось только более чем за 70 000 итераций основного цикла.

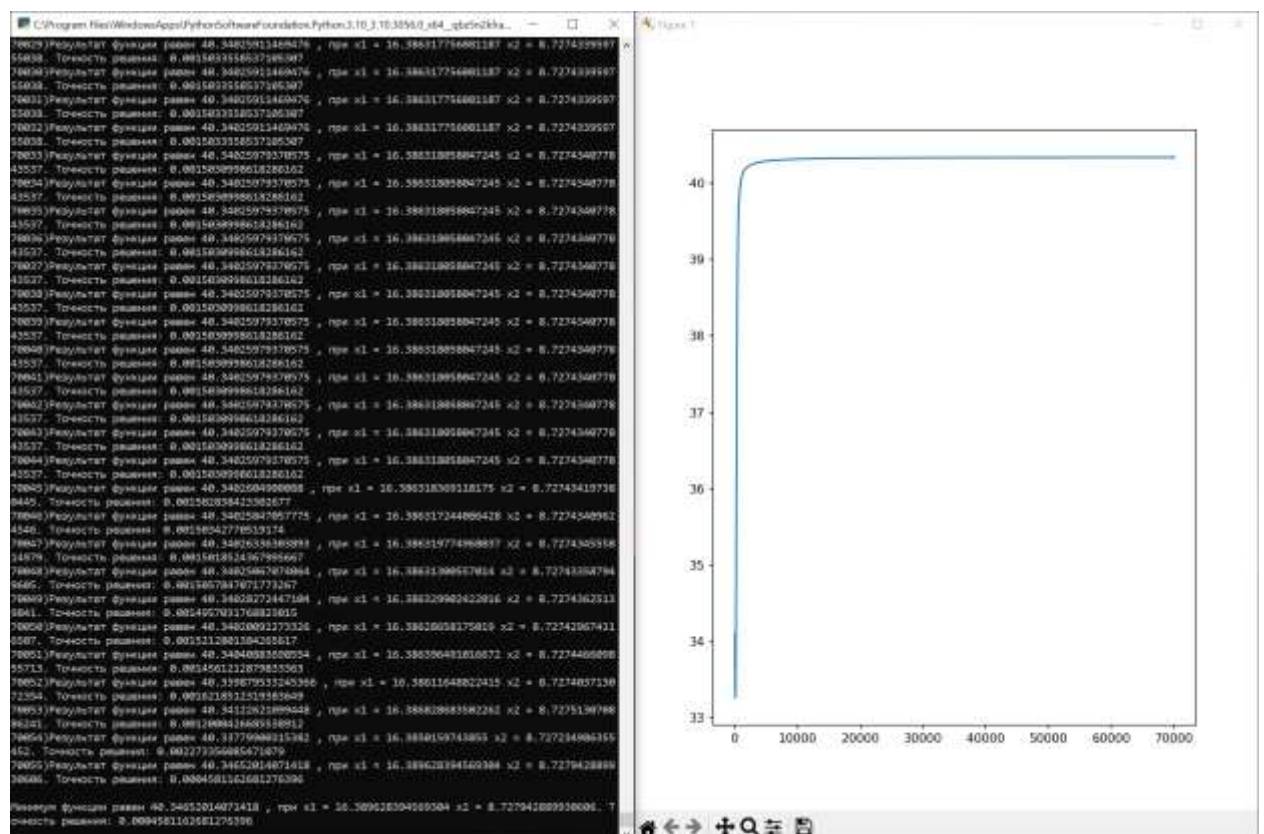


Рисунок 4 – Программное решение задачи при  $a=7$ ,  $c=7$

## ПРИЛОЖЕНИЕ 1

### Содержимое программы

```
import numpy as np
import matplotlib.pyplot as plt
import math;

def s(res, x):
    return (res[x])

def fun(x):
    return (x[0,0] + ((2 ** (1/2)) * x[1,0]) + (((x[1,0] ** 2) + ((x[0,0]
- x[1,0]) ** 2) ) ** (1/2)))

def g1(x):
    a = x[0,0]*x[1,0]
    if (a<0):
        return a
    else:
        return (0)

def g2(a, c, x):
    return ((a+5)+((c-2)*(x[0,0] - x[1,0])/x[1,0]) - x[0,0])

def Pfun(g1, g2,s,a,c,x):
    return(abs(g2(a, c, x)))

def gfun(s,a,c,r, x):
    result = []
    result.append(1 + ((x[0,0]-x[1,0])/(((x[1,0] ** 2) + ((x[0,0] -
x[1,0]) ** 2) ) ** (-1/2))) + r*( 2*((a+5)+((c-2)*(x[0,0]-
x[1,0])/x[1,0]) - x[0,0])*(((c-2)/x[1,0]) - 1) - abs(g1(x)/x[0,0])))
    result.append((2 ** (1/2)) + ((x[1,0]+x[1,0]-x[0,0])/(((x[1,0] ** 2)
+ ((x[0,0] - x[1,0]) ** 2) ) ** (-1/2))) + r*(2*((a+5)+((c-2)*(x[0,0]-
```

```

x[1,0])/x[1,0))-          x[0,0))*((c-2)/(-1*(x[1,0]
                        **2)))          -
abs(g1(x)/x[1,0]))
    return result

```

```

def grad(s,a,c,gfun, r,  x0, e):
    result = []
    la = 0.0001
    gradi = gfun(s,a,c,r,x0)
    a1 = abs(gradi[0])
    a2 = abs(gradi[1])
    while ((a1 > (e)) and (a2 > (e))):
        gradi = gfun(s,a,c,r,x0)
        x0[0,0] = x0[0,0] - (la * gradi[0])
        x0[1,0] = x0[1,0] - (la * gradi[1])
        if((a1 == abs(gradi[0])) and (a2 == abs(gradi[1]))):
            a1 = 0
            a2 = 0
            break
        else:
            a1 = abs(gradi[0])
            a2 = abs(gradi[1])

    return (x0)

```

```

a = abs(float(input('Введите a: ')))
c = abs(float(input('Введите c: ')))
e = 0.001
n = 0
S = 7
x0 = np.mat([[10.0], [10.0]])
r = 1
x = grad(S,a,c,gfun, r,  x0, e)
toc = abs(g1(x)) + abs(g2(a, c, x))

```



```

print(f'{n})Результат функции равен {fun(x)} , при  $x_1 = \{x[0,0]\}$   $x_2 = \{x[1,0]\}$ . Точность решения: {toc}')
resultfun = []

while (toc>e):
    r = r + 1
    n = n + 1
    x = grad(S,a,c,gfun, r, x0, e)
    toc = abs(g1(x)) + abs(g2(a, c, x))
    resultfun.append(fun(x))
    print(f'{n})Результат функции равен {fun(x)} , при  $x_1 = \{x[0,0]\}$   $x_2 = \{x[1,0]\}$ . Точность решения: {toc}')

print(f'')
print(f'Минимум функции равен {fun(x)} , при  $x_1 = \{x[0,0]\}$   $x_2 = \{x[1,0]\}$ . Точность решения: {toc}')

xx=np.arange(0,n,1)
plt.plot(xx, resultfun)
plt.show()

```

## ПРИЛОЖЕНИЕ 2

Вопросы:

### **1. Метод Лагранжа. Процедура сближения параметрических множителей.**

Метод, предложенный Лагранжем, может быть сформулирован в виде следующей теоремы.

Теорема о множителях Лагранжа.

Пусть  $x_0$  – решение задачи условной оптимизации

$$f(x_1, \dots, x_n) \rightarrow \min (\max) \quad (1)$$

$$q_1(x_1, \dots, x_n) = 0, \dots, q_m(x_1, \dots, x_n) = 0.$$

Тогда, если все функции в задаче (1) дифференцируемы в точке  $x_0$ , то их градиенты в этой точке линейно зависимы, то есть существуют такие не все равные нулю множители  $\lambda_0, \lambda_1, \dots, \lambda_m$ , для которых:

$$\lambda_0 \nabla f(x_0) + \lambda_1 \nabla q_1(x_0) + \dots + \lambda_m \nabla q_m(x_0) = 0. \quad (2)$$

Множители  $\lambda_0, \lambda_1, \dots, \lambda_m$  называют множителями Лагранжа.

Рассмотрим геометрическое обоснование теоремы о множителях Лагранжа на примере двумерной задачи с одним ограничением:

$$f(x, y) \rightarrow \min, q(x, y) = 0. \quad (3)$$

Функция  $f(x, y)$  представляет собой некоторую поверхность в трехмерном пространстве. Для изображения рельефа функции  $f(x, y)$  на плоскости  $xy$  будем использовать линии уровня – множества точек, с одинаковым значением функции. Таким образом, график задачи будет содержать семейство линий уровня целевой функции и кривую-ограничение (Рисунок 1). Для решения задачи требуется среди линий уровня целевой функции  $f(x, y)$  найти линию с наименьшим значением, имеющую хотя бы одну общую точку с кривой  $q(x, y) = 0$ . Из рис. 1 видно, что для любой линии уровня, пересекающей кривую ограничения, существует линия уровня с меньшим значением целевой функции, имеющая общую точку с кривой-ограничением.

Следовательно, искомая точка будет являться точкой касания линии уровня и кривой ограничения.

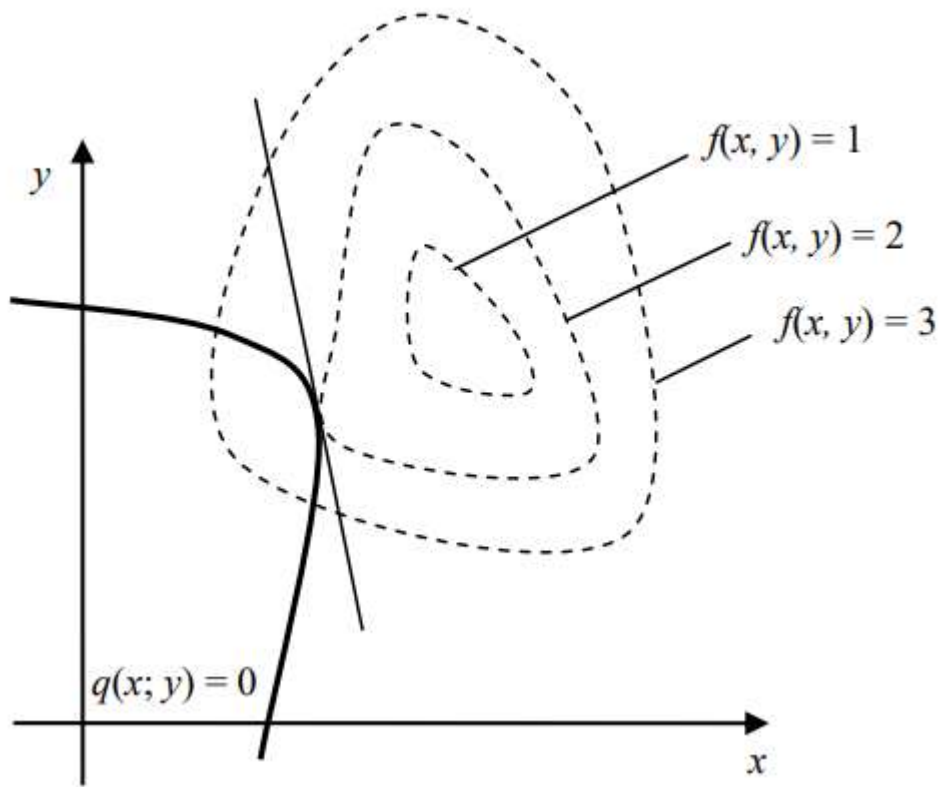


Рисунок 1. Геометрическое обоснование метода Лагранжа

Поиск точки касания опирается на одно из геометрических свойств градиента функции [4] – вектор градиента, построенный в любой точке функции, перпендикулярен касательной к функции в этой точке. Известно [4], что в точке касания двух кривых касательные к этим кривым совпадают, следовательно, векторы градиентов функций в точке касания коллинеарны. Таким образом, искомая точка минимума задачи (3.10) должна удовлетворять уравнению:

$$\lambda_0 \nabla f(x_0, y_0) = \lambda_1 \nabla q(x_0, y_0).$$

Уравнение (2) и условия задачи (1), согласно лемме Ферма, являются необходимым условием локального экстремума составной функции, называемой функцией Лагранжа:

$$L(x_1, \dots, x_n, \lambda_0, \lambda_1, \dots, \lambda_m) = \\ = \lambda_0 f(x_1, \dots, x_n) + \sum_{i=1}^m \lambda_i q_i(x_1, \dots, x_n) \rightarrow \min(\max).$$

Приравнявая, согласно лемме Ферма, частные производные функции Лагранжа нулю, получим уравнение (2) и условия задачи (1) – систему из  $n + m$  уравнений для  $n + m + 1$  неизвестных:  $x_1, \dots, x_n, \lambda_0, \lambda_1, \dots, \lambda_m$ . Несоответствие числа уравнений числу неизвестных связано с тем, что коэффициенты  $\lambda_0, \lambda_1, \dots, \lambda_m$  определены с точностью до общего множителя, так как обе части уравнения (2) могут быть умножены на любую ненулевую константу. Это означает, в частности, что если  $\lambda_0 \neq 0$ , можно разделить обе части (2) на  $\lambda_0$ , тем самым получить  $\lambda_0 = 1$  и уравнять число уравнений и число неизвестных. Таким образом, идея Лагранжа позволяет свести задачу поиска экстремума целевой функции  $n$ -го порядка с  $m$  ограничениями типа равенство (1) к задаче поиска стационарных точек составной целевой функции порядка  $n + m$  без ограничений.

## 2. Аппроксимирующие методы оптимизации. Описание и примеры.

Использование аппроксимаций (приближений) целевой функции – принципиально иной подход к решению задачи оптимизации по сравнению с

методами исключения интервалов.

Идея аппроксимации применима не только к одномерной, но и к многомерной

оптимизации и заключается в следующем итерационном алгоритме:

1. Задать целевую функцию  $f(x)$  и начальное приближение  $x_0$ .
2. Построить аппроксимирующую функцию  $\varphi(x)$ , такую, что она примерно равна  $f(x)$  в окрестности  $x_0$ .
3. Найти экстремум  $x_1$  функции  $\varphi(x)$  аналитическим методом.

4. Если желаемая точность не достигнута, то вернуться на шаг 2, используя  $x_1$  в качестве нового начального приближения. Иначе – завершить алгоритм.

В основе этого подхода лежит идея о том, что близкие функции имеют сходные свойства и их экстремумы расположены рядом. В рамках аппроксимационного подхода могут быть описаны многие численные методы, его применимость не ограничена методами оптимизации. Эффективность аппроксимационного метода определяется тем, насколько близки свойства целевой функции и её приближения.

Программное решение задачи, полученное в результате выполнения данной работы является примером использования аппроксимирующего метода.

### 3. Можно ли применить другой алгоритм, который даст решение менее, чем за 232 шага?

В данном случае метод барьерных функций плохо подходит, поскольку есть ограничения по типу  $g_n(x) = 0$ . Однако в данном методе за каждый шаг увеличивается множитель  $r$ :

```
63 ▼ while (toc>e):
64     r = r + 1
65     n = n + 1
66     x = grad(S,a,c,gfun, r, x0, e)
67     toc = gbc(g1(x)) + gbc(g2(a, c
```

Данный множитель отвечает за перевес условий над основной функцией. В данном случае он просто инкрементируется, однако если увеличивать или уменьшать прирост можно добиться меньшего или большего количество запусков метода градиентного спуска соответственно. Однако стоит уточнить, что при больших значениях  $r$  начинает дольше выполняться функция  $\text{grad}()$ , отвечающая за метод градиентного спуска, следовательно изменение шага увеличения данной переменной в некоторых примерах может быть

нецелесообразным и потому, чтобы в программе каждый раз не вычислять оптимальный прирост, было решено использовать тот, что дан на лекции(то есть инкрементирование).

При острой необходимости уменьшения количества шагов можно сильно увеличить приращение данной переменной, но уменьшать его (с откатом переменных до значений на предыдущей итерации цикла) если функция `grad()` начинает считать слишком долго (например, делает более 1000 проходов цикла. И при этом должно происходить прерывание выполнения данной функции).