МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчет по ЛР №7**

**по дисциплине**

**«Численные методы»**

Выполнил: студент гр. БЭИ2202

Тогузов

Вариант 24.

Проверил: доц. каф. «Информатика»

Мацкевич А. Г.

Москва, 2023 г.

1. **Задание для решения задачи многомерной оптимизации:**

* функция – Метод оптимизации для «ручного расчета» - значение параметра **p=3**;
* метод оптимизации для расчета на ПК – значение параметра t=1.

1. **Проверка существования минимума функции**

Известно, что всякий глобальный минимум выпуклой функции является одновременно и локальным.

Проверим, что функция является выпуклой на множестве R.

**Матрица Гессе** для функции :

а угловые миноры:

.

Таким образом, функция  - выпуклая на множестве R**.**

1. **Решение задачи многомерной оптимизации аналитическим методом**

Необходимые условия существования точки экстремума:



Откуда

1. **Начальная точка итерационного процесса численного решения задачи многомерной оптимизации**

Выберем начальную точку спуска - .

1. **Метод наискорейшего спуска (НСА):**

**Пример выполнения 3-х итерации по методу НСА:**

*Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание*

После 3-х итераций: f= -0.3725**.**

def f\_lamda(x, y):

  return (2\*x\*\*2 - 2\*x + 8\*y\*\*2 - 4\*y + 1)/(4\*x\*\*2 - 4\*x + 32\*y\*\*2 - 16\*y + 3)

def f\_lamda(x, y):

  return (60\*x - 48\*y + 144\*y\*\*2 + 100\*x\*\*2 + 13) / (-576\*y + 138 + 1728\*y\*\*2 + 1000\*x\*\*2 + 600\*x)

x = Symbol('x')

y = Symbol('y')

f\_ = x\*\*2 + 2 \* y\*\*2 - x - y

dfdx = f\_.diff(x)

dfdy = f\_.diff(y)

dfdx = lambdify(x, dfdx)

dfdy = lambdify(y, dfdy)

n = 3

x0, y0 = 1, 1

df\_NSA = pd.DataFrame(columns=['x', 'y', 'f', 'g1', 'g2', 'lamda'])

for i in range(n):

  fxy = f(x0, y0)

  lamda = f\_lamda(x0, y0)

  g1 = dfdx(x0)

  g2 = dfdy(y0)

  df\_NSA.loc[len(df\_NSA.index)] = [x0, y0, fxy, g1, g2, lamda]

  x0 = x0 - lamda \* g1

  y0 = y0 - lamda \* g2

df\_NSA.loc[len(df\_NSA.index)] = [x0, y0, fxy, np.nan, np.nan, np.nan]

1. **Метод градиентного дробления шага (ГДШ):**

**Пример выполнения 3-х итерации по методу ГДШ:**

*Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание*

После 3-х итераций: f= -3.365**.**

x0, y0 = 1, 1

df\_GDS = pd.DataFrame(columns=['x', 'y', 'f', 'g1', 'g2', 'lamda'])

for \_ in range(n):

  lamda = 0.618

  fxy = f(x0, y0)

  g1 = dfdx(x0)

  g2 = dfdy(y0)

  x1 = x0 - lamda \* g1

  y1 = y0 - lamda \* g2

  while fxy - f(x1, y1) < lamda / 2 \* (g1\*\*2 + g2\*\*2):

    lamda /= 2

    x1 = x0 - lamda \* g1

    y1 = y0 - lamda \* g2

  df\_GDS.loc[len(df\_GDS.index)] = [x0, y0, fxy, g1, g2, lamda]

  x0, y0 = x1, y1

df\_GDS

1. **Построение траектории поиска минимума методами НСА и ГДШ.**

fig = go.Figure()

fig.add\_trace(go.Scatter(x=df\_NSA.x, y=df\_NSA.y, name='НСА'))

fig.add\_trace(go.Scatter(x=df\_GDS.x, y=df\_GDS.y, name='ГДШ'))

fig.update\_layout(xaxis\_title="X",

                  yaxis\_title="Y",

                  xaxis=dict(scaleanchor="y", scaleratio=1))

*Изображение выглядит как снимок экрана, График, линия, диаграмма

Автоматически созданное описание*

Траектории методов НСА и ГДШ. Оба метода начинают в (1;1)

Изображение выглядит как линия, График, снимок экрана, диаграмма

Автоматически созданное описание

Оба метода начинают в (3;-1)

Изображение выглядит как снимок экрана, линия, График, Прямоугольник

Автоматически созданное описание

НСА – (2;0), ГДШ – (0;2)