Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта
Высшая школа автоматизации и робототехники

Отчёт

по лабораторной работе №1

Дисциплина: Методы и теория оптимизации

Студент гр. 3341506/10401 Якименко Г.К.
Преподаватель Третьяков Д.А.
« » _____ 2022 г.

Санкт-Петербург
2022

Введение

Работа была выполнена с помощью интерактивной среды исполнения кода языке python – Jupyter Notebook. Далее будут представлены оформленные страницы Jupyter Notebook с выполненным кодом в формате PDF.

Исходный код проекта: https://github.com/ImAllergicToFish/Optimization-methods

Задание №1

Найти аналитические стационарные точки проверить их на экстремальность, а также определить все локальные и глобальные минимумы и максимумы в приведенных ниже примерах и подтвердить их характер графиками функции:

8) *f(x1,x2) = x1^4 + x2^4 - (x1 + x2)^4*

Найдем производные функции

In [1]:
```
#sympy - библиотека для символьных вычислений
from sympy import *
from IPython.display import display

x1, x2 = symbols('x1 x2')
f = x1**4 + x2**4 - (x1 + x2)**4

dif_x1 = diff(f, x1)
dif_x2 = diff(f, x2)

print("Функция f:")
display(f)
print('Производная f по x1:')
display(dif_x1)
print('Производная f по x2:')
display(dif_x2)
```

```
Функция f:
```
$$x_1^4 + x_2^4 - (x_1 + x_2)^4$$
```
Производная f по x1:
```
$$4x_1^3 - 4(x_1 + x_2)^3$$
```
Производная f по x2:
```
$$4x_2^3 - 4(x_1 + x_2)^3$$

Решим систему уравнений для поиска стационарных точек

In [2]:
```
eq1 = Eq(dif_x1, 0)
eq2 = Eq(dif_x2, 0)
eqs = [eq1, eq2]

stat_p = solve(eqs, (x1, x2))

print('Стационарная точка: (x1, x2) =', stat_p[0])
```

```
Стационарная точка: (x1, x2) = (0, 0)
```

Проверим стационарную точку на экстремальность, для чего возьмем 2ю производную по каждой переменной и проверим их на достаточное условие экстремума функции двух переменных

In [3]:
```
dif_x1x1 = diff(dif_x1, x1)
dif_x1x2 = diff(dif_x1, x2)
dif_x2x2 = diff(dif_x2, x2)

print("Вторые производные функции f:")
```

```
print('-----------------------------------------------------')
display(dif_x1x1)
display(dif_x1x2)
display(dif_x2x2)
print('-----------------------------------------------------')

A = dif_x1x1.subs([(x1, stat_p[0][0]), (x2, stat_p[0][1])])
B = dif_x1x2.subs([(x1, stat_p[0][0]), (x2, stat_p[0][1])])
C = dif_x2x2.subs([(x1, stat_p[0][0]), (x2, stat_p[0][1])])

extreme_indicator = A*C - B**2

if(extreme_indicator > 0):
    if(A > 0):
        print('Локальный минимум в точке: (x1, x2) =', stat_p[0])
    if(A < 0):
        print('Локальный минимум в точке: (x1, x2) =', stat_p[0])
elif(extreme_indicator < 0):
    print('В точке (x1, x2) =', stat_p, 'экстремум отсутствует')
else:
    print('Четкий экстремум отсутствует, нужны дополнительные исследования')
    print('Вероятно, точка является седловой')
```

```
Вторые производные функции f:
-----------------------------------------------------
```

$$12x_1^2 - 12(x_1 + x_2)^2$$

$$-12(x_1 + x_2)^2$$

$$12x_2^2 - 12(x_1 + x_2)^2$$

```
-----------------------------------------------------
Четкий экстремум отсутствует, нужны дополнительные исследования
Вероятно, точка является седловой
```

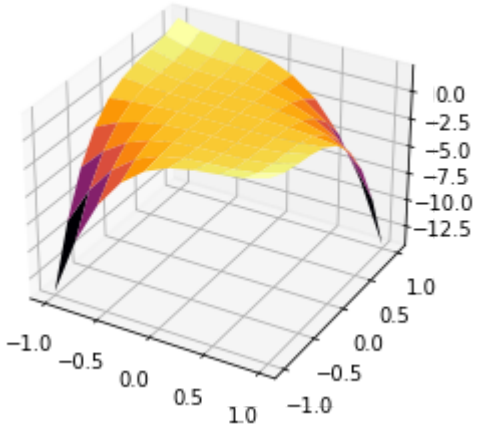
Подтвердим характер статической точки, построив график функции

In [4]:

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

#Инициализируем систему координат
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, projection = "3d")

#Задаем количество точек на ось границы
vals_quantity = 10
vals_range = np.linspace(-1, 1, vals_quantity)

#Генерируем значения сетки для x1 и x2
x1_vals, x2_vals = np.meshgrid(vals_range, vals_range)

#Инициализация пустого двумерного массива значений функции
f_vals = np.empty((vals_quantity, vals_quantity), dtype="float32")

#Подстановка значений x1, x2 в функцию f
i = 0
while(i < vals_quantity ):
    j = 0

    while(j < vals_quantity):
        f_vals[i][j] = f.subs([(x1, x1_vals[i][j]), (x2, x2_vals[i][j])])
        j = j + 1

    i = i + 1
```

```
#Отрисовка графика по точкам
ax.plot_surface(x1_vals, x2_vals, f_vals, cmap = 'inferno')
```

Out[4]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x7f36ca8f3ba8>



Задание №2

Вычислить **максимальное** значение функции f(x) на отрезке [a,b],
используя следующие методы первого порядка:

- метод установления границ интервала (метод Свенна);
- метод половинного деления;
- метод золотого сечения;
- метод равномерного поиска;

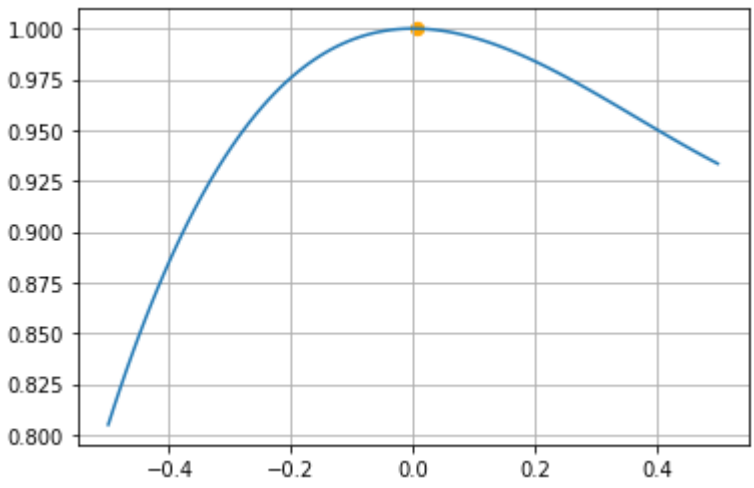
Точку оптимума x' определить с точностью не менее 10^-4.
Построить график функции *f(x)* на отрезке. Указать число шагов,
при котором достигается требуемая точность.

8) *f(x)* = *1/3* x^3 - (1 + x)(ln(1 + x) - 1)*, на интервале: [-0,5; 0,5]

Построим график функции

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return (1/3) * x**3 - (1 + x) * (np.log(1 + x) - 1)

a, b = [-0.5, 0.5]

x = np.linspace(a, b, 100)
f_x = f(x)

fig = plt.subplot()
fig.plot(x, f_x)
fig.grid()

i_max = np.argmax(f_x)
f_max = f_x[i_max]
x_max = x[i_max]

fig.scatter(x_max, f_max, color='orange', s=40, marker='o')
```

Out[2]:
```
<matplotlib.collections.PathCollection at 0x7f1838c5cb00>
```



Найдем максимум функции методом Свенна

In [3]:
```
eps = 10**(-5)

x0 = a
x1 = b

sven_counter = 0
while(np.abs(x1 - x0) > eps):
    xn = (x0 + x1)/2
    d = np.abs(x1 - x0)/10

    is_d_negative = False

    f_l = f(xn - d)
    f_xn = f(xn)
    f_r = f(xn + d)

    if(f_l <= f_xn and f_xn >= f_r):
        x0 = xn - d
        x1 = xn + d
    else:
        if(f_l <= f_xn and f_xn <= f_r):
            x0 = xn
        elif(f_l >= f_xn and f_xn >= f_r):
            d = -d
            is_d_negative = True
            x1 = xn

    while(True):
        sven_counter += 1
        xn1 = xn + d
        if(f(xn1) < f(xn)):
            if (is_d_negative):
                x0 = xn1
                x1 = xn
            else:
                x1 = xn1
            break
        else:
            if(not is_d_negative):
                x0 = xn
            d *= 2


res = (x0 + x1) / 2
f_res = f(res)

#Функция для вывода digits чисел после запятой
def toFixed(numObj, digits=0):
    return f"{numObj:.{digits}f}"

print('Максимум f(x) =', toFixed(f_res, 5), ' при x =', toFixed(res, 5))
print('Количество итераций:', sven_counter)
```

```
Максимум f(x) = 1.00000  при x = 0.00000
Количество итераций: 8
```

Найдем максимум методом половинного деления

In [4]:
```
x0 = a
x1 = b

mid = (x0 + x1)/2
f_mid = f(mid)

bin_counter = 0
while(np.abs(x1 - x0) > eps):
```

```
    mid = (x0 + x1)/2
    f_mid = f(mid)

    n = (x0 + mid)/2
    m = (mid + x1)/2

    f_n = f(n)
    f_m = f(m)

    if(f_n > f_mid):
        x1 = mid
    elif(f_m > f_mid):
        x0 = mid
    elif(f_n <= f_mid and f_m <= f_mid):
        x0 = n
        x1 = m
    bin_counter += 1

print('Максимум f(x) =', toFixed(f_mid, 5), ' при x =', toFixed(mid, 5))
print('Количество итераций:', bin_counter)
```

```
Максимум f(x) = 1.00000  при x = 0.00000
Количество итераций: 17
```

Найдем максимум методом золотого сечения

In [5]:

```
x0 = a
x1 = b

phi = (1 + np.sqrt(5))/2
gold_counter = 0

while(np.abs(x1 - x0) > eps):
    n = x1 - (x1 - x0)/phi
    m = x0 + (x1 - x0)/phi

    f_n = f(n)
    f_m = f(m)

    if(f_n <= f_m):
        x0 = n
    else:
        x1 = m

    gold_counter += 1

res = (x0 + x1)/2
f_res = f(res)

print('Максимум f(x) =', toFixed(f_res, 5), ' при x =', toFixed(res, 5))
print('Количество итераций:', gold_counter)
```

```
Максимум f(x) = 1.00000  при x = 0.00000
Количество итераций: 24
```

Найдем максимум методом равномерного поиска

In [6]:

```
x0 = a
f_x0 = f(x0)

uniform_counter = 0
i = 0
while (True):
    x_i = x0 + eps
```

```
    f_i = f(x_i)

    if(x_i == b):
        break

    if(f_i > f_x0):
        x0  = x_i
        f_x0 = f_i

    if(f_i < f_x0):
        break

    uniform_counter += 1

print('Максимум f(x) =', toFixed(f_x0, 5), ' при x =', toFixed(x0, 5))
print('Количество итераций:', uniform_counter)
```

```
Максимум f(x) = 1.00000  при x = 0.00000
Количество итераций: 50000
```

Составим таблицу количества итераций, сделанных каждым методом для поиска максимума

In [7]:

```
from prettytable import PrettyTable

counter_names = ['Метод', 'Свенна', 'Половинного деления',
    'Золотого сечения', 'Равномерного поиска']

counter_vals = ['Кол-во \n итераций', sven_counter, bin_counter,
    gold_counter, uniform_counter]

table = PrettyTable(counter_names)
table.add_row(counter_vals)
print(table)
```

```
+-----------+--------+----------------------+------------------+----------------------+
|   Метод   | Свенна | Половинного деления | Золотого сечения | Равномерного поиска |
+-----------+--------+----------------------+------------------+----------------------+
| Кол-во    |   8    |          17          |        24        |        50000         |
| итераций  |        |                      |                  |                      |
+-----------+--------+----------------------+------------------+----------------------+
```