МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ
НЭТИ | **Факультет прикладной математики и информатики**

Кафедра теоретической и прикладной информатики

Практическое задание №3

по дисциплине «Планирование и анализ экспериментов»

ПОСТРОЕНИЕ ДИСКРЕТНЫХ ОПТИМАЛЬНЫХ ПЛАНОВ ЭКСПЕРИМЕНТА

Группа ПМ-12	ВОСТРЕЦОВА ЕКАТЕРИНА
Вариант 4	ЗИЯНУРОВ АРТЁМ
	ХАМИТОВА ЕКАТЕРИНА

Преподаватели ПОПОВ АЛЕКСАНДР АЛЕКСАНДРОВИЧ

Новосибирск,2025

1. Задание

1. Изучить алгоритмы построения дискретных оптимальных планов.
2. Разработать программу построения дискретных оптимальных планов эксперимента, реализующую заданный алгоритм.
3. Для числа наблюдений 20, 25, 30, 35, 40 построить оптимальные планы на каждой из сеток, указанных в варианте задания. Выбрать лучшие дискретные планы для заданного числа наблюдений.
4. Оформить отчет, включающий в себя постановку задачи, результаты проведенных в п. 3 исследований, текст программы.
5. Защитить лабораторную работу.

Вариант 4

Двухфакторная квадратичная модель на квадрате со сторонами [1, +1]. Дискретное множество X – сетки 20×20 и 30×30 . Строить D- оптимальные планы. Последовательный алгоритм достраивания. Повторные наблюдения допускаются.

Постановка задачи

$$y = f^T(x)\theta + e = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + e,$$

У – значение зависимой переменной,

e – ошибка,

$f^T(x) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2)$ – заданная вектор функция, от независимой переменной x,

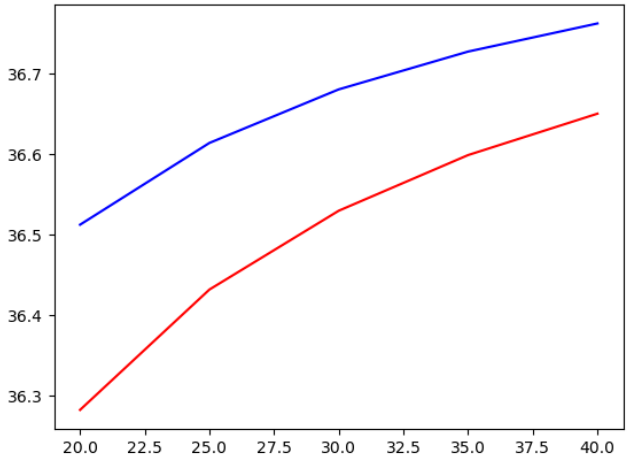
$\theta = (\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5)$ – вектор неизвестных параметров

2. Ход работы

Значение определителя для каждого плана:

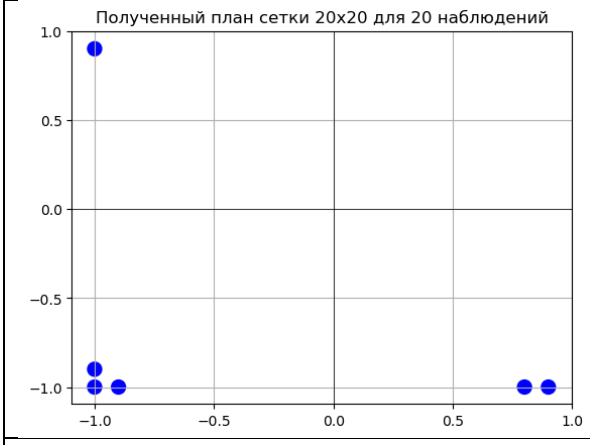
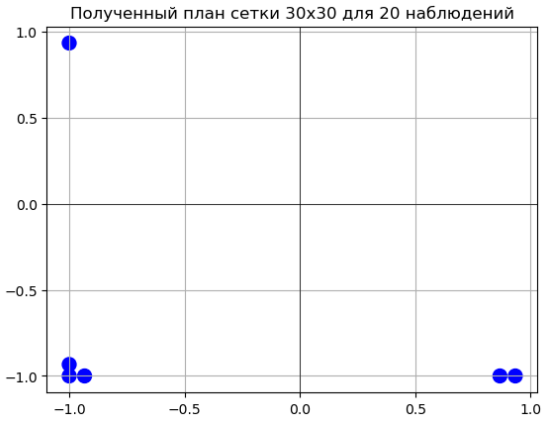
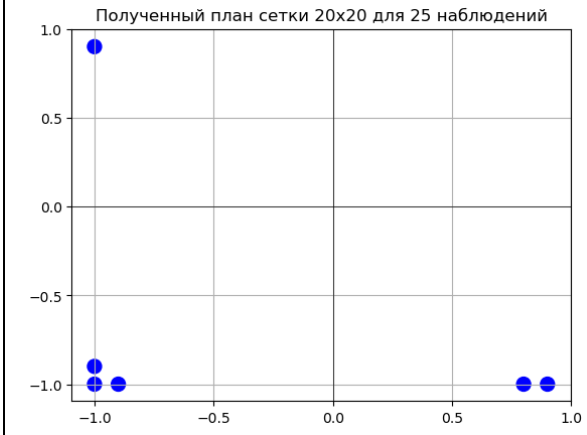
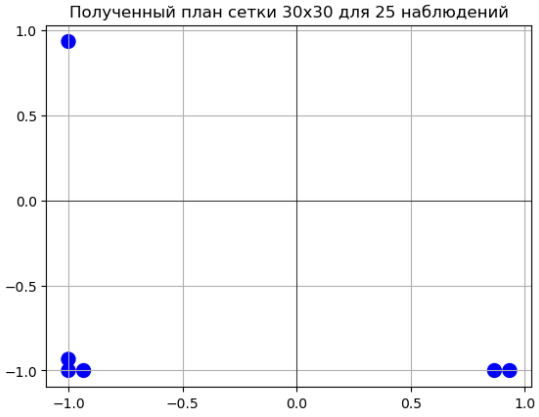
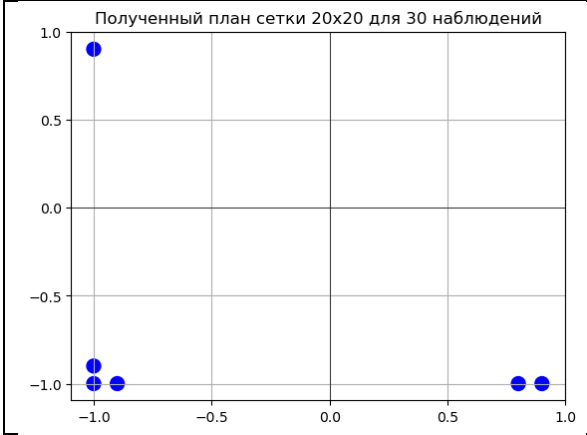
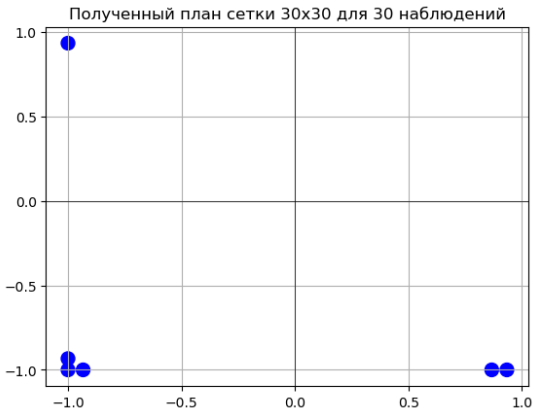
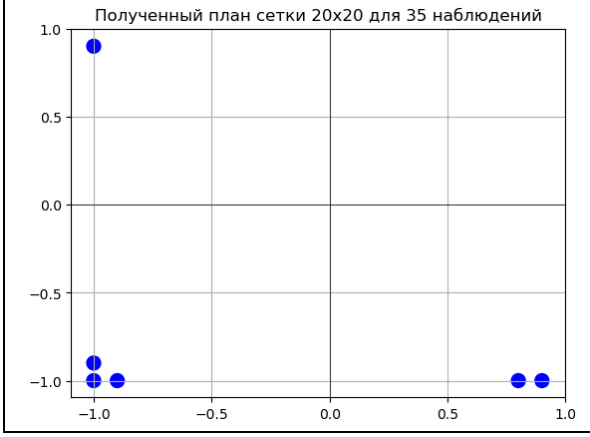
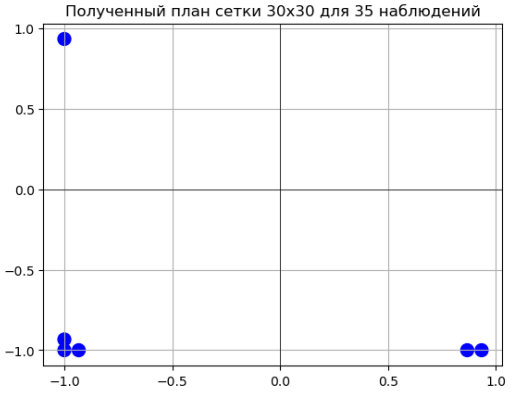
Число набл.\ размер сетки	20x20	30x30
20	36.2818897790436	36.512139707146844
25	36.431496075076176	36.613777268157904
30	36.529513993166454	36.68036739433756
35	36.59870311181848	36.72737218928792
40	36.650151430816116	36.76232447271257

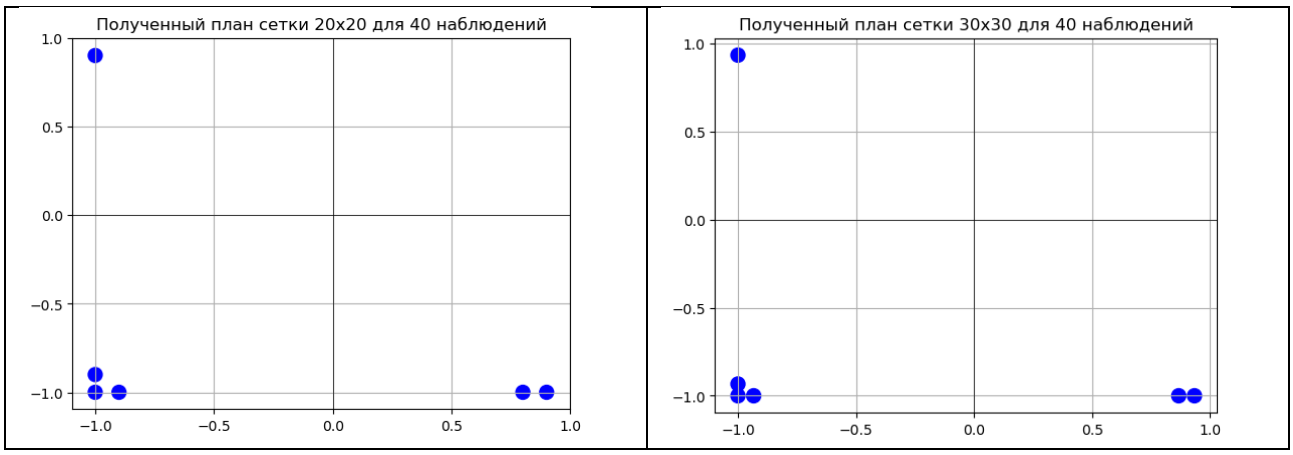
Зависимость определителя от сетки:



Кривая сверху – при сетке 30x30

Графики планов:

Сетка 20x20	Сетка 30x30
	
	
	
	



Наилучшими планами стали планы где количество было наибольшим(размер сетки 30x30), также больший размер сетки лучше максимизирует определитель.

3. Текст программы

```
#импорт библиотек
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import inv

#изначальные параметры
m = 6#количество неизвестных параметров
# для сетки 30x30 изначальное количество точек будет равно 900
N_list = [20, 25, 30, 35, 40]#для такой размерности нужно делать
grid_size = [20, 30]#30
n = grid_size[0]*grid_size[0]
s = n#для итераций
p=[0]*n#наши изначальные пэшки
# наша функция
def func(x):
    return np.array([1, x[0], x[1], x[0]*x[1], x[0]**2, x[1]**2])
# Информационная матрица
def M_calc(x, p, m):
    n_rg = np.count_nonzero(p)
    # Если n_rg < m, то матрица вырожденная, мы её регуляризируем

    M = np.zeros((m, m))
    for i in range(n):
        M += p[i] * (func(x[i]) @ func(x[i]).T)

    # Регуляризация
    if n_rg <= m:
        gamma = 1
        M += gamma * np.eye(m)
    return M
#Дисперсионная матрица
def D_calc(M):
```

```
    return np.linalg.inv(M)
def X_calc(grid_size):
    #создаем по 20(30) точек в этом отрезке
    x1 = np.linspace(-1,1,grid_size)
    x2 = np.linspace(-1,1,grid_size)
    #для создания в целом квадрата точек
    X1, X2 = np.meshgrid(x1, x2)#матрицы, где каждая строка копия x1(x2)
    X = np.column_stack([X1.ravel(), X2.ravel()])#преобразе матрицы в век-
тора, и выводим координаты
    return X
def d_calc(x, D):
    return func(x) @ D @ func(x).T
#изначальные матрицы и x
x = X_calc(grid_size[0])
M = M_calc(x, p, m)
D = D_calc(M)
len(x)

determinats = [[], []]
p = {20 : [],
     30 : []}
for grid in range(2):
    x = X_calc(grid_size[grid])
    n = grid_size[grid]*grid_size[grid]
    for j in range(len(N_list)):
        #тут начинается алгоритм
        CurP = np.zeros(n)
        for s in range(N_list[j]):
            M = M_calc(x, CurP, m)
            D = D_calc(M)

            maxD = 0
            i = -1
            for i in range(len(x)):
                if (s < m and CurP[i] == 0) or (s >= m):
                    cur_d = d_calc(x[i], D)
                    if maxD < cur_d:
                        maxD = cur_d
                        ind = i
            # перераспределение весов
            for k in range(n):
                if CurP[k] != 0:
                    if k == ind:
                        CurP[k] = (CurP[k] * s + 1) / (s + 1)
                    else:
                        CurP[k] = CurP[k] * s / (s + 1)
                else:
                    if k == ind:
                        CurP[k] += 1 / (s + 1)
        p[grid_size[grid]].append(CurP)
        determinats[grid].append(np.linalg.det(M))
```

```
    def printGrid(p, N, j, N_list):
        x = np.arange(-1, 1, 2/N)
        y = np.arange(-1, 1, 2/N)
        X, Y = np.meshgrid(x, y)
        sizes = []
        for i in range(len(p)):
            if p[i] > 0:
                sizes.append(100)
            else:
                sizes.append(0)

        plt.scatter(X, Y, s=sizes, c='blue')
        plt.axhline(0, color='black', linewidth=0.5)
        plt.axvline(0, color='black', linewidth=0.5)
        plt.xticks([-1, -0.5, 0, 0.5, 1])
        plt.yticks([-1, -0.5, 0, 0.5, 1])
        plt.grid(True)
        titl = f'Полученный план сетки {N}x{N} для {N_list[j]} наблюдений'
        plt.title(titl)
        plt.show()
    N = 30
    i=4
    printGrid(p[N][i], N,i,N_list)

    plt.plot(N_list, determinats[0], color='r', label='20')
    plt.plot(N_list, determinats[1], color='b', label='30')
    plt.show()
```