|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Практическое задание №2 | | |
| по дисциплине «Планирование и анализ экспериментов» | | |
| **Построение непрерывных оптимальных планов эксперимента** | | |
|  | | |
|  |  |  |
| Группа ПМ-12 | Вострецова екатерина |
| Вариант 4 | зиянуров артём |
|  | хамитова екатерина |
|  |  |
|  |  |
| Преподаватели | Попов александр александрович |
|  |  |
| Новосибирск,2025 | | |

# **Задание**

1. Изучить условия оптимальности планов эксперимента и алгоритмы синтеза непрерывных оптимальных планов эксперимента.
2. Разработать программу построения непрерывных оптимальных планов эксперимента, реализующую последовательный или комбинированный алгоритм. Применить программу для построения оптимального плана для тестового примера из варианта заданий. Для отчета предусмотреть выдачу на печать протокола решения по итерациям. При большом числе итераций предусмотреть вывод протокола с некоторой дискретностью.
3. Оформить отчет, включающий в себя постановку задачи, протокол решения, а также текст программы.
4. Защитить лабораторную работу.

**Вариант 4**

Двухфакторная квадратичная модель на квадрате [–1, +1]. Начальный план – сетка с шагом 0.25 по каждой координате. Веса равные. Строить A оптимальные планы. Комбинированный алгоритм (неполный его вариант).

# **Постановка задачи**

У – значение зависимой переменной,

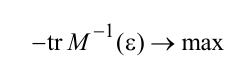
е – ошибка,

– заданная вектор функция, от независимой переменной х,

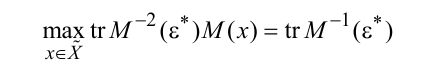
*–* вектор неизвестных параметров

1. **Ход работы**

Критерий оптимальности:

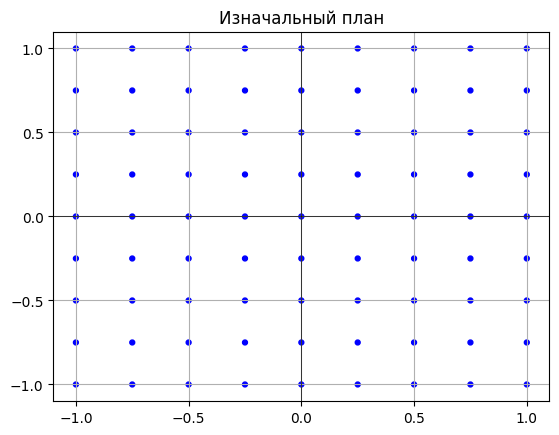


Необходимые и достаточные условия оптимальности:



|  |  |  |
| --- | --- | --- |
| Итерация | Функционал критерия | Ошибка |
| 1 | 2,912E+01 | 5,217E-01 |
| 50 | 1,984E+01 | 1,333E-01 |
| 100 | 1,846E+01 | 3,078E-02 |
| 150 | 1,813E+01 | 2,166E-02 |
| 200 | 1,800E+01 | 1,039E-02 |
| 250 | 1,794E+01 | 1,035E-02 |
| 300 | 1,789E+01 | 1,916E-03 |
| 350 | 1,789E+01 | 9,922E-06 |
| 351 | 1,789E+01 | 9,922E-06 |

**Графики планов**

****

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

Контент, сгенерированный ИИ, может содержать ошибки.**

1. **Текст программы**

#импорт библиотек

import numpy as np

import matplotlib.pyplot as plt

from numpy.linalg import inv

#параметры, постоянные

m=6#насколько я поняла - это по факту размер нашего вектора иксов\тет

n = 81# с шагом 0.25 у нас будет выходить это

p = [1/n] \* n # веса равные, равны вот этому

#наша функция

def f(x):

    func = np.array([1,x[0],x[1],x[0]\*x[1],x[0]\*\*2,x[1]\*\*2])

    return func

#создаем наш x

x1 = np.arange(-1,1.1,0.25)

x2 = np.arange(-1,1.1,0.25)

x = []

for i in range(len(x1)):

    for j in range(len(x2)):

        x.append([round(x1[i],2), round(x2[j],2)])

x = np.array(x)

def Mcalc(x, p):

    M = np.zeros([m, m])

    for i in range(n):

        M += p[i] \* np.outer(f(x[i]),f(x[i]).T)

    return M

def A\_functional(M):

    return np.trace(np.linalg.inv(M))

def phi(x, D):

    return np.trace(np.dot(np.dot(D, D), np.outer(f(x), f(x).T)))

def grad\_func\_A(M, x):

    grad = []

    D = np.linalg.inv(M)

    for i in range(n):

        grad.append(phi(x[i], D))

    return grad / np.linalg.norm(grad)

def NecessaryCond(fi, mean\_no0):

    for i in range(n):

        if p[i] != 0 and abs(mean\_no0 - fi[i]) > 1e-5:

            return False

    return True

def SufficientCond(Grd, fi, mean\_no0):

    flag = 1

    for i in range(n):

        Grd[i] = fi[i] - mean\_no0

        if p[i] == 0.0:

            if Grd[i] > 0:

                flag = 0

            else:

                Grd[i] = 0

    if(flag == 1):

        return True

    return False

def q\_sort(N, p, fi):

    q = N - np.count\_nonzero(p)

    mean = 0

    for i in range(N):

        if (p[i] != 0):

            mean += fi[i]

    mean /= (N - q)

    return mean

def step(Grd):

    lyambda = 0.1

    for i in range(n):

        if(Grd[i] < 0 and lyambda > -p[i] / Grd[i]):

            lyambda =  -p[i] / Grd[i]

    for i in range(n):

        p[i] += lyambda \* Grd[i]

    return p

def gradient\_plan\_A(p, x):

    neccocaryCond = False

    iterNum = 1

    iterProc = []

    kMax = 10000

    Grd = np.zeros(n)

    while neccocaryCond == 0 and iterNum < kMax:

        # Проверяем условия оптимальности

        M = Mcalc(x, p)

        D = np.linalg.inv(M)

        fi = grad\_func\_A(M, x)

        # Количество не нулевых компонент

        mean\_no0 = q\_sort(n, p, fi)

        f1= NecessaryCond(fi, mean\_no0)

        f2 = SufficientCond(Grd, fi, mean\_no0)

        if (iterNum % 50 == 0 or iterNum == 1):

            print(iterNum, A\_functional(M), np.linalg.norm(Grd), sep='\t')

        iterNum += 1

        if (f1 and f2):

            break

        else:

            step(Grd)

    print(iterNum, A\_functional(M), np.linalg.norm(Grd), sep='\t')

    return p

print(" i", " A(->max)", " необходимо и достаточно (->0)", "p\_sum")

p0 = gradient\_plan\_A(p, x)

print(p0)

total = sum(p0)

p1 = [x / total for x in p0]

x = np.arange(-1, 1.1, 0.25)

y = np.arange(-1, 1.1, 0.25)

X, Y = np.meshgrid(x, y)

sizes = [1/n\*1000]\*n

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)

plt.title("Изначальный план")

plt.show()

x = np.arange(-1, 1.1, 0.25)

y = np.arange(-1, 1.1, 0.25)

X, Y = np.meshgrid(x, y)

sizes = [val\*1000 for val in p/sum(p)]

plt.scatter(X, Y, s=sizes, c='blue')

plt.axhline(0, color='black', linewidth=0.5)

7

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)

plt.title("Полученый план")

plt.show()