Министерство науки и высшего образования
Российской Федерации

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

Кафедра теоретической и прикладной информатики

Лабораторная работа № 2

по дисциплине «Математические методы оптимального планирования эксперимента»

ПОСТРОЕНИЕ НЕПРЕРЫВНЫХ ОПТИМАЛЬНЫХ ПЛАНОВ ЭКСПЕРИМЕНТА

Факультет:	ПМИ
Группа:	ПМИМ-21
Вариант:	3
Студенты:	Демидович Е.
	Стародубцев С.
	Цыганков А.
Преподаватель:	Попов А.А.

Новосибирск

2022

1. Цель работы

Изучить алгоритмы, используемые при построении непрерывных оптимальных планов эксперимента.

2. Содержание работы

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

3. Постановка задачи

Исследуемая модель имеет вид:

$y = f^T(x)\theta + e = \sum_{l=1}^{m} f_l(x)\,\theta_l + e$, где y – значение зависимой переменной,

$f^T(x) = \left(f_1(x), \ldots, f_m(x)\right)$ – заданная вектор-функция от независимой векторной переменной x, которая может изменяться на заданном отрезке [-1, 1],

$\theta = (\theta_1, \ldots, \theta_m)^T$ – вектор неизвестных параметров, которые необходимо определить по результатам экспериментов (измерений),

e – ошибка, распределённая по нормальному закону.

Критерий A-оптимальности:

$$\varepsilon^* = \arg\min_{\varepsilon} tr(D(\varepsilon))$$

Необходимое и достаточное условие A-оптимальности:

$$\max_{x \in \bar{X}} trM^{-2}(\varepsilon^*)\,M(x) = trM^{-1}(\varepsilon^*)$$

Заданная вектор-функция:

$$f^T(x) = (1, x_1, x_2, x_1 * x_2, x_1^2, x_2^2)$$

Исходный план – полный факторный эксперимент из 49 точек, на уровнях –1, –0.75, –0.25, 0, +0.25, +0.75, +1, веса равны 1/49:

$\boldsymbol{x_1}$	-1	-1	…	1	1
$\boldsymbol{x_2}$	-1	-0.75	…	0.75	1
$\boldsymbol{p}$	0.02041	0.02041	…	0.02041	0.02041

Для построения оптимального плана будем использовать последовательный алгоритм.

<u>Шаг 1 - основная часть алгоритма:</u>

1. Выбираем невырожденный план ε^0 и номер итерации $s = 0$

2. Отыскиваем глобальный экстремум x^s: $x^s = arg\max_{x\epsilon\bar{X}} \varphi(x,\varepsilon^s)$, где $\varphi(x,\varepsilon^s) = f^T(x)\frac{\partial\Psi[\mathrm{M}(\varepsilon)]}{\partial \mathrm{M}(\varepsilon)}f(x) = f^T(x)\mathrm{M}^{-1}(\varepsilon)f(x) = d(x,\varepsilon)$

3. Проверяем необходимое и достаточное условие оптимальности, если выполняется, то заканчиваем:

$$\left|-\max_{x\epsilon\bar{X}}\varphi(x,\varepsilon^s) + trM(\varepsilon^s)\mathrm{M}^{-1}(\varepsilon)\right| \le \delta, \text{ где } \delta = \left|\max_{x\epsilon\bar{X}}\varphi(x,\varepsilon^s)\right| 0{,}01$$

4. Составляем план $\varepsilon^{s+1} = (1-\alpha^s)\varepsilon^s + \alpha^s\varepsilon(x^s)$, где $\alpha^s = \frac{1}{n^s}$. Пересчитываем веса $p_j^{s+1} = (1-\alpha^s)p_j^s, j = \overline{1,n_s}$ и добавляем x^s с весом $p_{n^s+1}^{s+1} = \alpha^s$.

5. Делаем сравнение, если $\Psi[\mathrm{M}(\varepsilon^s)] > \Psi[\mathrm{M}(\varepsilon^{s+1})]$, то α^s уменьшаем в два раза и переходим на шаг 4,иначе s заменяем на s+1 и осуществляем переход на шаг 2

Шаги 2 и 3 - очистка плана и удаление точек с малыми весами:

Проверяем:

1) $\left|\left|x_j^* - x_j^s\right|\right|^2$, $j = \overline{1,n_s}$, где μ – малое положительное число

2) $\left|p_j^* - p_j^s\right| \le \pi$, где π – малое положительное число;

3) план ε^s по сравнению с планом ε^* имеет "посторонние" точки $x_{n+1}^s, \dots, x_{n+v}^s$ с малыми весами $\tau \ge p_{n+1}^s \ge \cdots \ge p_{n+v}^s$;

4) вместо одной точки x_j^s, близкой к x_j^*, имеется набор точек $x_{j_1}^s, \dots, x_{j_l}^s$, каждая из которых близка к x_j^* : $\left|\left|x_j^* - x_j^s\right|\right|^2 \le \mu, k = 1,\dots,l$ и их суммарный вес близок к p_j^*: $\left|p_j^* - \sum_{k=1}^l p_{j_k}^s\right| \le \pi$

Производим очистку:

1. Точки, тяготеющие к одной из групп, объединяются по правилу

$$p_j^s = \sum_{k=1}^{l} p_{j_k}^s, x_j^s = \frac{1}{p_j^s\sum_{k=1}^l x_{j_k}^s p_{j_k}^s}$$

2. Точки с малыми весами, не тяготеющие ни к одной из групп, указанных в п.4 выбрасываются. Их веса перераспределяются между оставшимися точками.

После этого проверяем очищенный план на оптимальность.

4. Ход выполнения

Шаг 1 – основная часть алгоритма

Протокол выполнения

Номер итерации	***Длина плана***	***δ***	***d***
1	50	1.03263723	76.67195912
250	534	0.1983171	1.03761762
500	1034	0.19054281	0.51956565
750	1534	0.18787288	0.35287603

1000	2034	0.18688175	0.3083739
1250	2534	0.18601181	0.25631119
1500	3034	0.18526003	0.20564919
1540	3134	0.18498881	0.18179114

Проверка на А-оптимальность сгенерированного плана:

- $trM^{-1}(\varepsilon^*) = 18.317089642768735$
- $\max\limits_{x \in \tilde{X}} trM^{-2}(\varepsilon^*)\, M(x) = 18.31708964276874$

Шаг 2 – очистка плана

x1	***x2***	***p***
-1	-1	0.09308292
-1	-0.25	0.09554764
-1	0.25	0.00166054
-1	0.75	0.09350551
-0.75	-0.75	0.00083027
-0.75	0	0.00083027
-0.75	1	0.00083027
-0.25	-1	0.09518052
-0.25	-0.25	0.00249081
-0.25	0.25	0.00166054
-0.25	0.75	0.00249081
0	-0.75	0.00166054
0	0	0.22239221
0	1	0.09573963
0.25	-1	0.00083027
0.25	-0.25	0.00083027
0.25	0.25	0.00083027
0.25	0.75	0.00083027
0.75	-1.	0.09385967
0.75	-0.25	0.00249081
0.75	0.25	0.00166054
0.75	0.75	0.00249081
1	-0.75	0.00083027
1	0	0.0948255
1	1	0.09261889

Шаг 3 – удаление точек с малыми весами

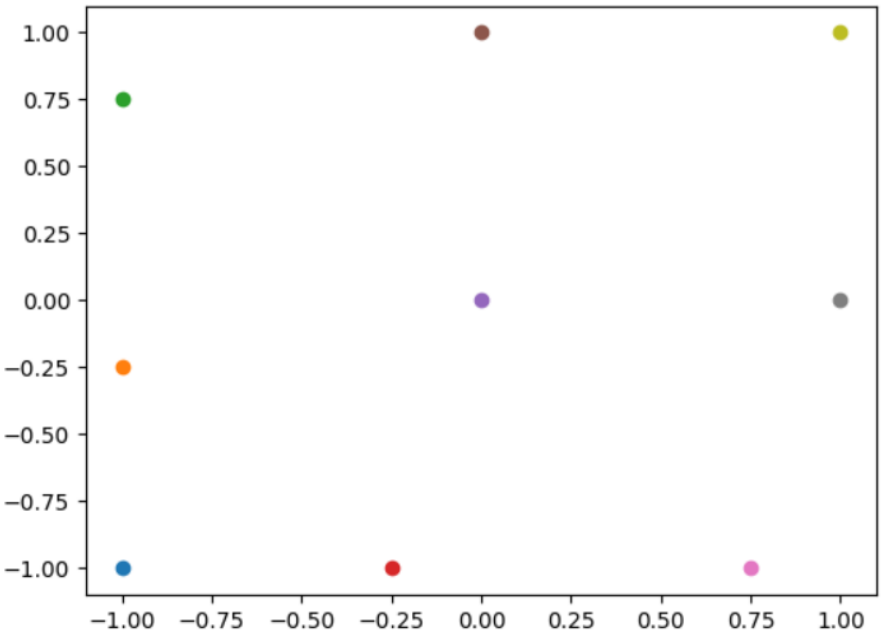
x1	***x2***	***p***
-1	-1	0.09308291584028708
-1	-0.25	0.09554763643963925
-1	0.75	0.09350550701473029
-0.25	-1	0.0951805191421396
0	0	0.2223922064137818
0	1	0.09573963154531294

0.75	-1	0.09385966956921558
1	0	0.09482550224490265
1	1	0.09261889389655475
-1	-1	0.09308291584028708

Проверка очищенного плана на A-оптимальность:

- $trM^{-1}(\varepsilon^*) = 29.45062226797519$
- $\max\limits_{x \in \bar{X}} trM^{-2}(\varepsilon^*)\, M(x) = 29.4506222679752$

График очищенного плана:



6. Код программы

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

m = 6
gamma = 2


# f(x)
def f(a_, b_):
    f_ = np.array([[1], [a_], [b_], [a_ * b_], [a_ ** 2], [b_ ** 2]])
    return f_


# чтение плана из файла
def read_plan():
    with open("data2.json", "r") as file:
        x1_ = []
        x2_ = []
        p_ = []
```

```
        plans = json.load(file)["plan"]
        for plan in plans:
            x1_.append(plan["x1"])
            x2_.append(plan["x2"])
            p_.append(plan["p"])
        return x1_, x2_, p_


# Построение матрицы M
def get_mat_m(c, d, p1, mat_m):
    mat_m = np.zeros((mat_m, mat_m))
    n = len(x1)
    for q in range(0, n - 1):
        mat_m += p1[q] * f(c[q], d[q]) @ np.transpose(f(c[q], d[q]))
    return mat_m


# Построение дисперсионной матрицы D
def get_mat_d(mat_m):
    mat_d = np.linalg.inv(mat_m)
    return mat_d


# Нахождение глобального экстремума
def get_global_max(mat_d):
    phi_max = -100
    argmax_x1 = 0
    argmax_x2 = 0
    l1 = -1
    i1 = -1
    while i1 <= 1.0:
        i2 = -1
        while i2 <= 1.0:
            phi = (np.transpose(f(i1, i2)) @ mat_d @ mat_d @ f(i1, i2)).item()
            if phi_max < phi:
                phi_max = phi
                argmax_x1 = i1
                argmax_x2 = i2
                l1 += 1
            i2 += 0.01
        i1 += 0.01
    return phi_max, argmax_x1, argmax_x2


# Проверка необходимых и достаточных условий
def check_condition(phi_, mat_m, mat_d):
    delta = np.abs(phi_) * 0.01
    print("delta = ", delta)
    d = np.abs(np.trace(mat_d @ mat_d @ mat_m) - phi_)  # A-plan
    print("d = ", d)
    return d <= delta


# Составление нового плана
def create_plan(i1, i2, ps1, b1, b2, a1):
    ns = len(ps1)
    for w in range(0, ns):
        ps1[w] = (1 - a1) * ps1[w]

    b1.append(i1)
    b2.append(i2)
    ps1 = np.append(ps1, a1)

    print("len(x1) = ", len(b1))
    print("len(x2) = ", len(b2))
    print("len(p) = ", len(ps1))
    return ps1, b1, b2
```

```
# Создание графика
def show_scatter(a_, b_):
    for i1 in range(0, len(a_)):
        plt.scatter(a_[i1], b_[i1])
    plt.plot()
    plt.show()


x1, x2, p = read_plan()

# Выбор плана e0, s = 0
s = 0
print("План e", s)
M = []
D = []

while True:
    M = get_mat_m(x1, x2, p, m)
    D = get_mat_d(M)

    phiMax, x1Max, x2Max = get_global_max(D)

    if check_condition(phiMax, M, D):
        print("алгоритм оптимален")
        print("x1 = ", x1)
        print("x2 = ", x2)
        print("p = ", p)
        break

    a = 1 / len(p)
    p, x1, x2 = create_plan(x1Max, x2Max, p, x1, x2, a)

    Ms = get mat m(x1, x2, p, m)
    Ds = get_mat_d(Ms)
    while True:
        if np.trace(Ds) > np.trace(D):
            print("det(Ms) = ", np.linalg.det(Ms), "det(M) = ", np.linalg.det(M))
            a /= gamma
            print("a = ", a)
            p, x1, x2 = create_plan(x1Max, x2Max, p, x1, x2, a)
            Ms = get_mat_m(x1, x2, p, m)
            Ds = get_mat_d(Ms)
        else:
            break

    M = get_mat_m(x1, x2, p, m)
    D = get_mat_d(M)
    print("s = ", s)
    if s % 250 == 0:
        print("len(x1) = ", len(x1), "x1:\n", x1)
        print("len(x2) = ", len(x2), "x2:\n", x2)
        print("len(p) = ", len(p), "p:\n", p)
        print("Проверка на оптимальность - неочищенный план A")
        print("Правая часть: ", np.trace(D))
        print("Левая часть: ", np.trace(D @ D @ M))

    s += 1

M = get_mat_m(x1, x2, p, m)
D = get_mat_d(M)

# Проверка на оптимальность
print("Первая проверка на оптимальность - неочищенный план A")
print("Правая часть: ", np.trace(D))
print("Левая часть: ", np.trace(D @ D @ M))
```

```
newX1 = [0]
newX2 = [0]
newP = [0]
newX1[0] = x1[0]
newX2[0] = x2[0]
newP[0] = p[0]
mu = 0.1
pi = 0.02
k = len(x1)
r = 0
i = 1
print("sum p = ", np.sum(p))

while True:
    if r >= len(p) - 1:
        break

    while i < len(p):
        scalar = (x1[r] - x1[i]) ** 2 + (x2[r] - x2[i]) ** 2
        if scalar <= mu:  # проверка на близость
            p[r] += p[i]  # добавление веса к точке
            p = np.delete(p, i)
            x1 = np.delete(x1, i)
            x2 = np.delete(x2, i)
            i -= 1
        i += 1
    r += 1
    i = r + 1

print("Выполняется очистка плана")

print("len(x1) = ", len(x1), "x1:\n", x1)
print("len(x2) = ", len(x2), "x2:\n", x2)
print("len(p) = ", len(p), "p:\n", p)

sumP = 0
newX1 = x1
newX2 = x2
newP = p
for i in range(1, len(newP)):  # найти в массиве p малые веса
    if newP[i] <= pi:
        sumP += newP[i]  # сумма малых весов
        newP[i] = 0

newX11 = []
newX21 = []
newP1 = []
for i in range(0, len(newP)):  # удаление из массивов элементов с нулевыми весами
    if newP[i] != 0:
        newP1.append(newP[i])
        newX11.append(newX1[i])
        newX21.append(newX2[i])

p = newP1
x1 = newX11
x2 = newX21

sumP = sumP / len(p)

print("План очищен!")
M = get_mat_m(x1, x2, p, m)
D = get_mat_d(M)

# Проверка на оптимальность
print("Вторая проверка на оптимальность - очищенный план А")
print("Правая часть: ", np.trace(D))
```

```
print("Левая часть: ", np.trace(D @ D @ M))
show_scatter(x1, x2)

print("len(x1) = ", len(x1), "x1:\n", x1)
print("len(x2) = ", len(x2), "x2:\n", x2)
print("len(p) = ", len(p), "p:\n", p)
```