|  |  |  |
| --- | --- | --- |
| 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 № 4 | | |
| по дисциплине «Математические методы оптимального планирования эксперимента» | | |
|  | | |
| **оптимальное планирование эксперимента для****нелинейных регрессионных моделей** | | |
|  | | |
|  | Факультет: | ПМИ |
| Группа: | ПМИМ-21 |
| Вариант: | 3 |
| Студенты: | Демидович Е. |
|  | Стародубцев С. |
|  | Цыганков А. |
| Преподаватель: | Попов А.А. |
|  |  |
|  | | |
| Новосибирск | | |
| 2022 | | |

# Цель работы

Изучить методы оптимального планирования эксперимента при нелинейной параметризации функции отклика.

# Задание

1. Изучить понятия локально-оптимального планирования информационной матрицы при нелинейной параметризации функции отклика, ознакомиться с видом производственной функции Кобба-Дугласа
2. По заданному типу технологии сформировать имитационную модель в виде производственной функции Кобба-Дугласа. При этом задать истинные значения для параметров, нелинейно входящих в модель. Выход модели зашумить, уровень шума установить в пределах 15%–20% от мощности полезного сигнала.
3. Выбрать план для затравочного эксперимента, состоящий из небольшого числа наблюдений, и смоделировать на его основе экспериментальные данные.
4. Оценить параметры модели по полученным экспериментальным данным. Для этого необходимо перейти к линейной модели, воспользовавшись логарифмическим представлением уравнения модели наблюдения. Параметры преобразованной модели тогда можно оценить обычным "линейным" МНК.
5. Построить локально-оптимальный план эксперимента для исходной нелинейной модели, воспользовавшись разработанной ранее программой синтеза дискретных оптимальных планов и полученными оценками параметров модели. Число наблюдений должно в 4-5 раз превышать число параметров модели.
6. По сформированной ранее (п.2) имитационной модели провести имитационный эксперимент в точках полученного локально-оптимального плана. Провести оценку параметров и вычислить норму отклонения оценок от их истинных значений. Вычислительный эксперимент повторить не менее 100 раз, каждый раз с новой реализацией помехи. Вычислить среднее значение нормы отклонения оценок. Процедуру повторить, используя в качестве плана эксперимента случайно расположенные точки в факторном пространстве. В серии вычислительных экспериментов случайный план фиксируется (выбирается один раз). Сделайте вывод об эффективности оптимального планирования эксперимента для идентификации заданной нелинейной модели.
7. Оформить отчет, включающий в себя постановку задачи, оценки параметров по затравочному эксперименту, полученный локально-оптимальный план, результаты проведенного в п. 6 исследования и текст программы.
8. Защитить лабораторную работу.

# Вариант

Технология Кобба-Дугласа. Число входных ресурсов 2. Возрастающая отдача от масштаба. Ресурсы изменяются в пределах [0, 5]. Локально-D-оптимальное планирование.

# Постановка задачи

Модель наблюдения описывается уравнением

y – значение зависимой переменной, – фактор независимых переменных, – вектор неизвестных параметров, ошибка наблюдения, – нелинейная функция вектора параметров, в моделе Кобба-Дугласса представлена как

При возрастающей отдаче от масштаба параметры удовлетворяют ограничению

Информационная матрица для нелинейной модели определяется, как:

План локально D-оптимальный, если

Пусть, уровень шума 20%,

# Результаты

МНК-оценки параметров по затравочному эксперименту:

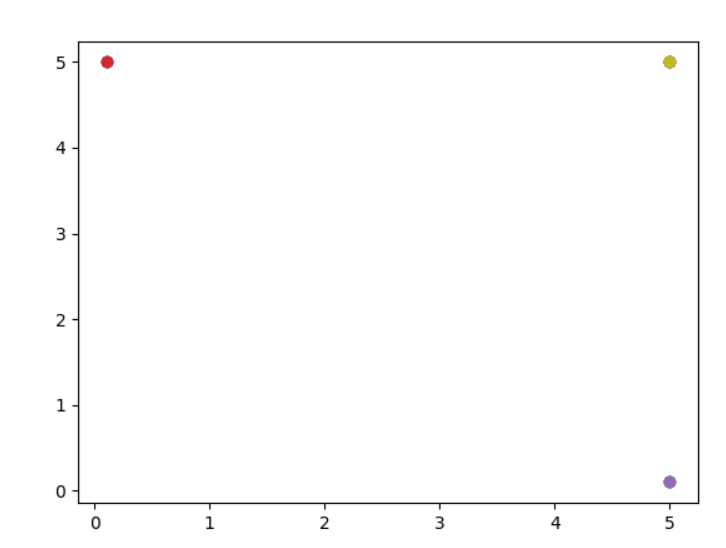
Изменение определителя информационной матрицы M

|  |  |
| --- | --- |
| № итерации |  |
| 1 | 0.002423830451882872 |
| 2 | 0.026759353585257818 |
| 3 | 0.050286121951075555 |
| 4 | 0.0926539330025328 |
| 5 | 0.13308095437979764 |
| 6 | 0.18815322394564693 |
| 7 | 0.23674274447657254 |
| 8 | 0.2896232343595149 |
| 9 | 0.32086109638074434 |
| 10 | 0.35772976465844036 |
| 11 | 0.39150351126356825 |
| 12 | 0.4241606785233429 |
| 13 | 0.4559250703946063 |
| 14 | 0.48122288178022066 |
| 15 | 0.49655311911899785 |

Полученный оптимальный план:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 5.0 | 5.0 | 5.0 | 0.1 | 5.0 | 5.0 | 0.1 | 0.1 | 5.0 | 5.0 | 5.0 | 5.0 | 0.1 | 0.1 | 5.0 | 5.0 |
|  | 0.1 | 0.1 | 5.0 | 5.0 | 5.0 | 5.0 | 5.0 | 5.0 | 0.1 | 0.1 | 5.0 | 0.1 | 5.0 | 5.0 | 5.0 | 0.1 |

График полученного плана

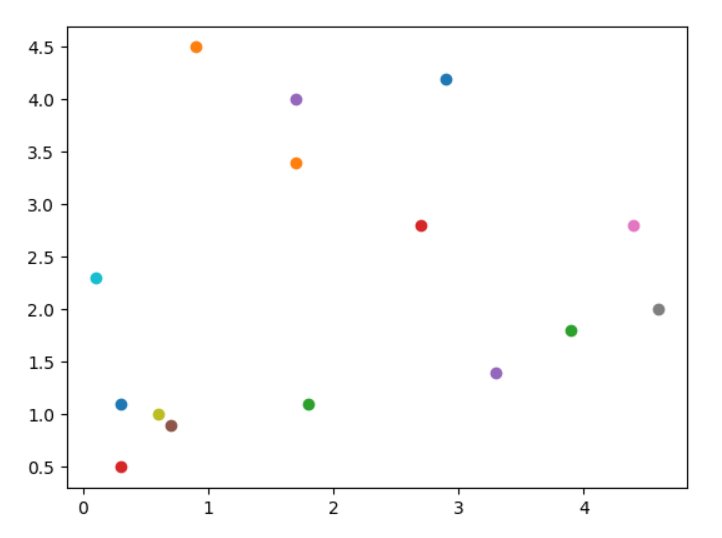


Результаты проведения вычислительного эксперимента (количество повторений 100):

План полученный из случайно расположенных точек в факторном пространстве:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 2.9 | 0.9 | 1.8 | 0.3 | 1.7 | 0.7 | 4.4 | 4.6 | 0.6 | 0.1 | 0.3 | 1.7 | 3.9 | 2.7 | 3.3 |
|  | 4.2 | 4.5 | 1.1 | 0.5 | 4.0 | 0.9 | 2.8 | 2.0 | 1.0 | 2.3 | 1.1 | 3.4 | 1.8 | 2.8 | 1.4 |

График полученного плана:



Результаты проведения вычислительного эксперимента (количество повторений 100):

# Код программы

import numpy as np

import random

import matplotlib.pyplot as plt

def func(x, theta):

    return np.array([x[0] \*\* theta[1] \* x[1] \*\* theta[2],

                     theta[0] \* theta[1] \* x[0] \*\* (theta[1] - 1) \* x[1] \*\* theta[2],

                     theta[0] \* theta[2] \* x[0] \*\* theta[1] \* x[1] \*\* (theta[2] - 1)])

def makeY(plan, theta, p = 0.2):

    Y = list(map(lambda x: theta[0] \* x[0]\*\*theta[1] \* x[1]\*\*theta[2], plan))

    Y = list(map(lambda y: y + random.normalvariate(0, p \* y), Y))

    return Y

def makePlan(grid = np.linspace(0, 5, 1001), m = 10):

    return list(map(lambda x: [random.choice(grid), random.choice(grid)], range(m)))

def makeX(plan):

    return np.array(list(map(lambda x: [1.0, np.math.log(x[0]), np.math.log(x[1])], plan)))

def OLS(X, Y):

    thetahead = np.dot(np.dot(np.linalg.inv(np.dot(X.T, X)), X.T), Y)

    thetahead[0] = np.math.e \*\* thetahead[0]

    return thetahead

def makeM(x, N, theta):

    M = np.zeros((len(func(x[0], theta)), len(func(x[0], theta))))

    for i in range(len(x)):

        M += 1.0/N \* make\_partM(func(x[i], theta))

    return M

def make\_partM(fx):

    M = np.zeros((len(fx), len(fx)))

    for i in range(len(fx)):

        for j in range(len(fx)):

            M[i][j] = fx[i] \* fx[j]

    return M

def makeD(M):

    return np.linalg.inv(M)

def d(x, D, newx, thetahead):

    return np.dot(np.dot(func(x, thetahead), D), func(newx, thetahead).T)

def Delta(x, D, N, newx, thetahead):

    return 1./float(N) \* (d(newx, D, newx, thetahead) - d(x, D, x, thetahead))\

           - 1./float(N)\*\*2 \* (d(x, D, x, thetahead) \* d(newx, D, newx, thetahead) - d(x, D, newx, thetahead)\*\*2)

def findMaxforOneX(x, D, N, grid, thetahead):

    maxdot = [grid[0], grid[0]]

    maxvalue = Delta(x, D, N, maxdot, thetahead)

    for x1 in grid:

        for x2 in grid:

            value = Delta(x, D, N, [x1, x2], thetahead)

            if value > maxvalue:

                maxvalue = value

                maxdot = [x1, x2]

    return [maxvalue, maxdot]

def findMaxforAll(X, D, N, grid, thetahead):

    listofmax = [findMaxforOneX(x,D,N,grid,thetahead) for x in X]

    return [\*max(listofmax),listofmax.index(max(listofmax))]

def makeOptimalPlan(thetahead, plan, grid, N):

    eps = 0.001

    iteration = 0

    print(thetahead)

    while True:

        M = makeM(plan, N, thetahead)

        print("det", np.linalg.det(M))

        D = makeD(M)

        print(iteration)

        delta = findMaxforAll(plan, D, N, grid, thetahead)

        if delta[0] > eps:

            plan[delta[2]] = delta[1]

        else:

            break

        iteration += 1

    return plan

def RSS(Y, X, thetahead):

    Yhead = np.dot(X, thetahead)

    return np.dot(Y - Yhead, Y - Yhead)

def Experiment(theta, plan):

    ARSS = 0

    Ahead = 0

    for i in range(100):

        Y = makeY(plan, theta)

        Y = np.log(Y)

        X = makeX(plan)

        thetahead = OLS(X, Y)

        ARSS += RSS(Y, X, thetahead)

        Ahead += np.dot(thetahead - theta, thetahead - theta)

    ARSS /= 100

    Ahead /= 100

    return ARSS, Ahead

def draw\_graph(points):

    for p in points:

        plt.scatter(p[0], p[1])

    plt.plot()

    plt.show()

N = 15

grid = np.linspace(0.1, 5, 50)

plan = makePlan(grid = grid)

theta = [0.4, 0.4, 0.4]

Y = makeY(plan, theta)

Y = np.log(Y)

X = makeX(plan)

thetahead = OLS(X, Y)

firstplan = makePlan(grid = grid, m = N)

optplan = makeOptimalPlan(thetahead, firstplan, grid, N)

ARSS, Ahead = Experiment(theta, optplan)

print("Optplan")

print(optplan)

print("Average RSS ", ARSS)

print("Average norm head ", Ahead)

draw\_graph(optplan)

firstplan = makePlan(grid = grid, m = N)

ARSS, Ahead = Experiment(theta, firstplan)

print("Random plan")

print(firstplan)

print("Average RSS ", ARSS)

print("Average norm head ", Ahead)

draw\_graph(firstplan)