Отчет о проделанной работе

Лосева Елизавета Юрьевна
Группа **8.2**
Лабораторная работа **1**
Вариант **3**

Скриншот текста варианта из списка вариантов:

3. Провести стратегическое и тактическое планирование модельного эксперимента. Выходной реакцией системы является случайная величина, распределенная по закону распределения экстремального (минимального) значения. Факторами являются параметры: a ϵ (5, 8); b ϵ (3, 4). Оценить показатель эффективности вероятность исхода реакции системы > 6. Доверительный интервал d = 0.3 с уровнем значимости α = 0.01.

Скриншот используемых формул:

| | | | | |
|---|---|---|---|---|
| 11 | Распределение экстремального (минимального) значения | $Xe: a, b$ | $f(x)=\frac{1}{b}exp\left(-\frac{x-a}{b}\right)exp\left[-exp\left(\frac{x-a}{b}\right)\right]$, $-\infty < x < \infty$, $m = a+b(-0{,}57721)$, $D = b^2\pi^2/6$ | $Xe: a, b \sim a + b\ln\left(\ln R^{(0)}\right)$ |

Код лабораторной работы с комментариями (можно также скриншотами):

```
import numpy as np
from scipy.stats import norm
from scipy.integrate import quad
import matplotlib.pyplot as plt
import math
import sys
import io

sys.stdout = io.TextIOWrapper(sys.stdout.buffer, encoding='utf-8')

# Функция генерации случайной величины
def systemeqv(a, b):
    """
    Генерация случайной величины, распределенной
    по закону экстремального (минимального) значения:
    Xe(a,b) = a + b * ln(-ln(R)), где R ~ U(0,1)
    """
    R = np.random.rand()
```

```
    while R == 0 or R == 1:
        R = np.random.rand()
    return a + b * math.log(-math.log(R))


# Плотность распределения экстремального значения
def extreme_pdf(x, a, b):
    """
    f(x) = (1/b) * exp(-(x-a)/b) * exp(-exp((x-a)/b))
    """
    if b <= 0:
        return 0.0
    z = (x - a) / b
    if z > 100:
        return 0.0
    return (1 / b) * np.exp(-z) * np.exp(-np.exp(z))


# Вычисление теоретической вероятности через интеграл
def theoretical_probability(a, b, x_bound=6.0):
    """
    P(X > x_bound) = ∫[x_bound, ∞] f(x) dx
    """
    if b <= 0:
        return 0.0

    # Верхний предел ограничиваем, чтобы избежать переполнения
    upper_limit = min(a + 15 * b, 100)

    try:
        result, _ = quad(extreme_pdf, x_bound, upper_limit, args=(a, b),
limit=200, epsabs=1e-8, epsrel=1e-8)
    except Exception:
        result = 0.0

    return result


# задание факторов и диапазонов значений факторов
nf = 2  # число факторов
minf = np.array([5, 3])
maxf = np.array([8, 4])

# Генерация дробного двухуровневого факторного плана: уровни -1 и +1
# a, b - факторы, ab - их взаимодействие (a*b)
a = np.array([-1, +1, -1, +1]) # столбец a
b = np.array([-1, -1, +1, +1]) # столбец b
ab = a * b # столбец ab
```

```
fracplan = np.column_stack((a, b, ab)) # замена fracfact в MATLAB
# вычисление количества экспериментов (строчки в матрице)
N = 2 ** nf

print("fracplan =")
print(fracplan)

# фиктивный фактор (столбец из единиц)
X = np.column_stack((np.ones((N, 1)), fracplan)).T
print("\nX =")
print(X)

# Определение данных для проведения эксперимента
fraceks = np.zeros((N, nf))
for i in range(N):
    for j in range(nf):
        fraceks[i, j] = minf[j] + (fracplan[i, j] + 1) * (maxf[j] - minf[j]) / 2

print("\nfraceks =")
print(fraceks)


# Задание параметров тактического планирования
d = 0.03      # доверительный интервал
alpha = 0.01  # уровень значимости
p_max_variance = 0.25  # max[p(1-p)] = 0.25

# определение t-критического
tkr_alpha = norm.ppf(1 - alpha / 2)

# Определение требуемого числа испытаний
NE = int(np.ceil(tkr_alpha ** 2 * p_max_variance / d ** 2))
print(f"\nТребуемое число испытаний NE = {NE}")

# Инициализация массива выходных значений Y
Y = np.zeros(N)

# Цикл по совокупности экспериментов стратегического плана
for j in range(N):
    a_val = fraceks[j, 0]
    b_val = fraceks[j, 1]

    results = np.zeros(NE) # Массив для хранения результатов испытаний
    # Цикл статических испытаний
    for k in range(NE):
        results[k] = systemeqv(a_val, b_val)

    # Оценка вероятности P(X > 6)
```

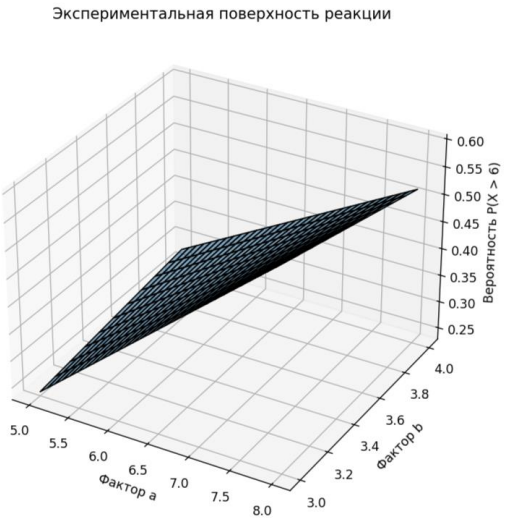
```
    Y[j] = np.mean(results > 6)

print("\nМассив выходных значений Y по экспериментам:")
print(Y)


# Вычисление матрицы C = X * X'
C = X @ X.T

# Вычисление коэффициентов регрессии: b_ = inv(C) * X * Y'
b_ = np.linalg.solve(C, X @ Y)

print("\nКоэффициенты линейной регрессии b_ =")
print(b_)

# Формирование сетки значений факторного пространства
stepA = 0.1
stepB = 0.1
A = np.arange(minf[0], maxf[0] + 0.05, stepA)
B = np.arange(minf[1], maxf[1] + 0.05, stepB)
A_grid, B_grid = np.meshgrid(A, B)

# Нормированные факторы
An = 2 * (A - minf[0]) / (maxf[0] - minf[0]) - 1
Bn = 2 * (B - minf[1]) / (maxf[1] - minf[1]) - 1
An_grid, Bn_grid = np.meshgrid(An, Bn)

# Экспериментальная поверхность реакции
Yc = b_[0] + An_grid * b_[1] + Bn_grid * b_[2] + An_grid * Bn_grid * b_[3]

# Теоретическая поверхность реакции
Yo_grid = np.zeros_like(A_grid)
for i in range(A_grid.shape[0]):
    for j in range(A_grid.shape[1]):
        Yo_grid[i, j] = theoretical_probability(A_grid[i, j], B_grid[i, j], 6.0)


# Построение графиков
fig = plt.figure(figsize=(14, 6))

# Экспериментальная поверхность
ax1 = fig.add_subplot(121, projection='3d')
surf1 = ax1.plot_surface(A_grid, B_grid, Yc, edgecolor='k', alpha=0.5)
ax1.set_xlabel('Фактор a')
ax1.set_ylabel('Фактор b')
ax1.set_zlabel('Вероятность P(X > 6)')
ax1.set_title('Экспериментальная поверхность реакции')
ax1.grid(True)
```

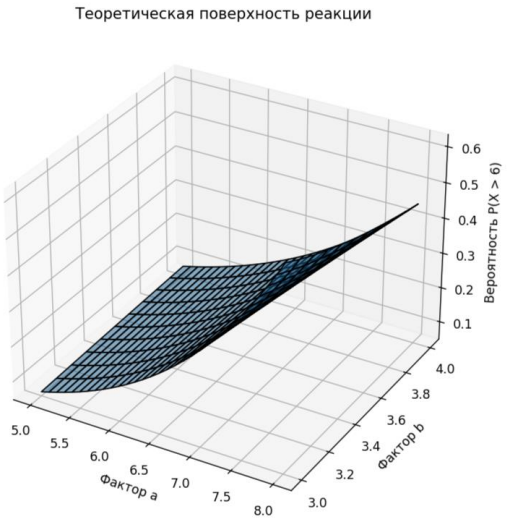
```
# Теоретическая поверхность
ax2 = fig.add_subplot(122, projection='3d')
surf2 = ax2.plot_surface(A_grid, B_grid, Yo_grid, edgecolor='k', alpha=0.5)
ax2.set_xlabel('Фактор a')
ax2.set_ylabel('Фактор b')
ax2.set_zlabel('Вероятность P(X > 6)')
ax2.set_title('Теоретическая поверхность реакции')
ax2.grid(True)

plt.tight_layout()
plt.show()
```

Определения:

1. Стратегическое планирование - определение структуры эксперимента (какие факторы изучать, на каких уровнях)
2. Тактическое планирование - определение количества испытаний для достижения требуемой точности
3. Факторы - входные параметры системы (a и b), которые мы варьируем
4. Уровни факторов - конкретные значения факторов в экспериментах (-1 и +1 в кодированном виде)
5. Матрица планирования - таблица, определяющая условия проведения каждого эксперимента
6. Доверительный интервал (d) - точность оценки вероятности события P(X > 6)
7. Уровень значимости (α) - вероятность ошибочного вывода
8. Регрессионная модель - математическая зависимость отклика системы от факторов
9. Поверхность реакции - графическое представление зависимости отклика от факторов
10. Распределение экстремальных значений – распределение вероятностей, описывающее экстремальные значения в выборках
11. Показатель эффективности – вероятность того, что выходная реакция системы превысит заданное значение (P(X > 6))

Полученные графики и результаты:


Экспериментальная поверхность реакции
Вероятность P(X > 6)
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
Фактор a
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Фактор b
3.0
3.2
3.4
3.6
3.8
4.0


Теоретическая поверхность реакции
Вероятность P(X > 6)
0.6
0.5
0.4
0.3
0.2
0.1
Фактор a
5.0
5.5
6.0
6.5
7.0
7.5
8.0
Фактор b
3.0
3.2
3.4
3.6
3.8
4.0

```
fracplan =
[[-1 -1  1]
 [ 1 -1 -1]
 [-1  1 -1]
 [ 1  1  1]]

X =
[[ 1.  1.  1.  1.]
 [-1.  1. -1.  1.]
 [-1. -1.  1.  1.]
 [ 1. -1. -1.  1.]]

fraceks =
[[5. 3.]
 [8. 3.]
 [5. 4.]
 [8. 4.]]

Требуемое число испытаний NE = 1844

Массив выходных значений Y по экспериментам:
[0.23590022 0.60032538 0.27982646 0.51572668]

Коэффициенты линейной регрессии b_ =
[ 0.40794469  0.15008134 -0.01016811 -0.03213124]
```

Вывод о проделанной работе: Проведено стратегическое и тактическое планирование эксперимента для системы с распределением экстремального минимального значения. Построенные экспериментальная и теоретическая поверхности реакции вероятности $P(X > 6)$ хорошо согласуются. Рассчитанное число испытаний обеспечило требуемую точность оценки. Работа подтвердила эффективность методов факторного анализа для исследования вероятностных характеристик систем.