**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра теории вероятностей и математической статистики**

**ОТЧЕТ**

по лабораторной работе №5

«Точечный кригинг»

учебной дисциплины

«Математические методы анализа данных»

Вариант №3

**Выполнила:**

Лавринович Анна Павловна

3 курс 7а группа, специальность «прикладная математика»

**Преподаватель:**

Цеховая Татьяна Вячеславовна,

кандидат физико-математических наук, доцент

Минск, 2025

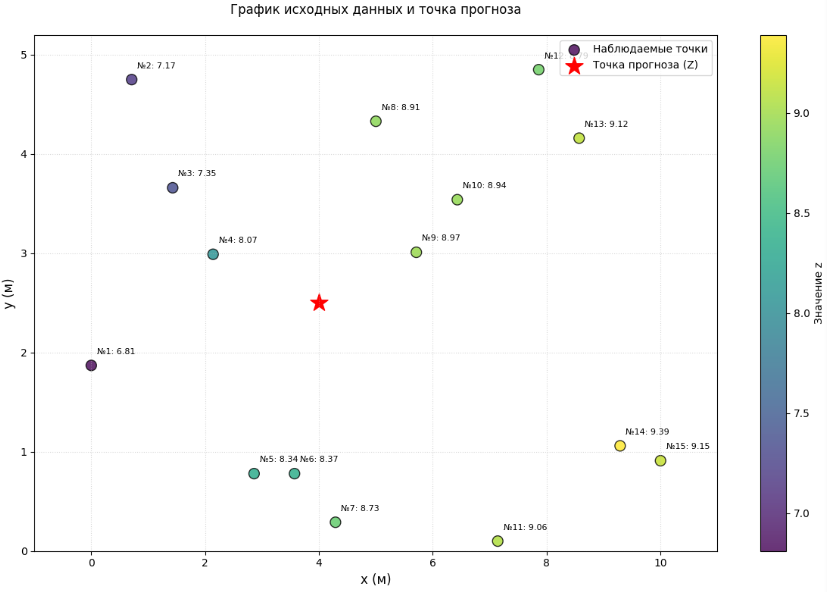
**Постановка задачи.** Самостоятельно подобрать ряд исходных данных объёмом n=15. Провести точечный (обычный) кригинг и построить прогноз в ненаблюдаемой точке.

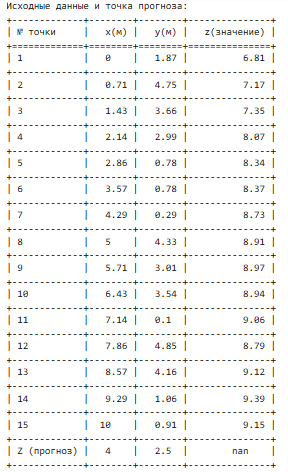
Необходимо:

* Выбрать стационарные исходные данные в двумерном пространстве. Представить их в виде таблицы и графически.
* Построить оценку семивариограммы.
* Подобрать модель семивариограммы.
* Записать аналитический вид модели семивариограммы. Указать радиус влияния.
* Рассчитать весовые коэффициенты модели кригинга и осуществить прогноз в ненаблюдаемой точке.

**Исходные данные (алгоритм выполнения).**

1. Подберём ряд исходных данных в двумерном пространстве объёмом n=15 и представим этот ряд графически.

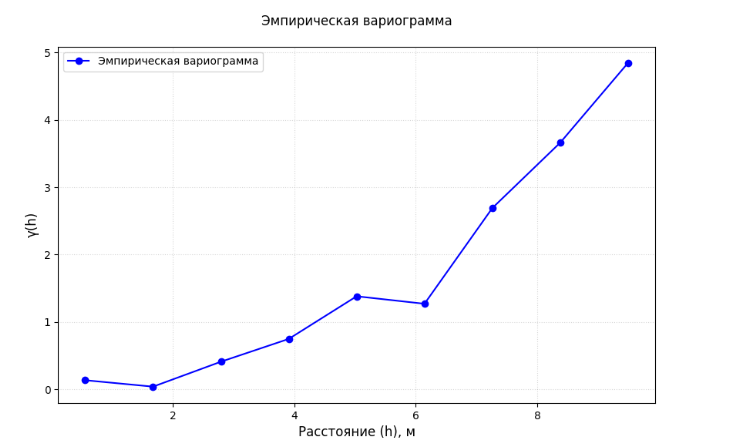




1. Построим оценку семивариограммы

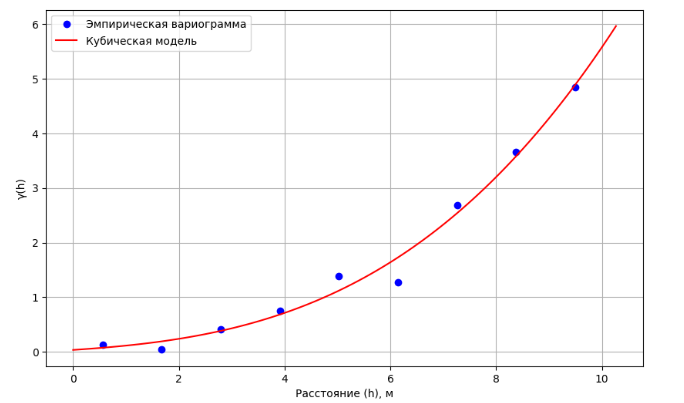
,

где – количество пар точек в лаге.



Оценка семивариограммы демонстрируют ожидаемое поведение — увеличение дисперсии разностей с ростом расстояния. Корреляция значений очень сильна на расстояниях до ~3 м, затем постепенно убывает.

1. В качестве модели семивариограммы возьмём кубическую модель, параметры для которой подберём с помощью метода наименьших квадратов.



Кубическая модель семивариограммы адекватно аппроксимирует эмпирическую вариограмму на рассматриваемом интервале. Наблюдается ожидаемый рост дисперсии с расстоянием, что указывает на наличие пространственной автокорреляции в данных. Существенных систематических отклонений модели от эмпирических данных не наблюдается.

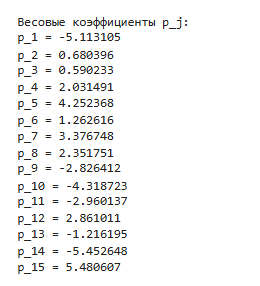
1. Кубическая модель: γ(h) = 0.034 + 0.064h + 0.011h² + 0.004h³

В качестве радиуса влияния мы берём точку, где модель впервые достигает значения дисперсии.

Радиус влияния: 3.60 м

1. Рассчитаем весовые коэффициенты модели кригинга.

Для этого мы строим находим решение уравнения , где A – матрица ковариций между точками (D - γ(h)), а b – ковариации между ненаблюдаемой точкой и всеми остальными.



Несмотря на большой разброс в полученных значениях, сумма весовых коэффициентов равна единице, а дисперсия ошибки мала (1.77), значит весовые коэффициенты наедены правильно.

Теперь можем осуществить прогноз в ненаблюдаемой точке:

Прогнозируемое значение в точке Z: 9.479124

**Вывод**: Метод кригинга позволил построить качественный прогноз на основе ограниченного набора пространственных данных. Построенная модель семивариограммы корректно отражает пространственную структуру данных, а полученные результаты демонстрируют эффективность метода для решения задач пространственного анализа.

Листинг программы:

import pandas as pd

import matplotlib.pyplot as plt

from tabulate import tabulate

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from scipy.spatial.distance import pdist, squareform

from scipy.optimize import curve\_fit

data = {

"№ точки": [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, "Z (прогноз)"],

"x(м)": [0.0, 0.71, 1.43, 2.14, 2.86, 3.57, 4.29, 5.0, 5.71, 6.43, 7.14, 7.86, 8.57, 9.29, 10.0, 4.0],

"y(м)": [1.87, 4.75, 3.66, 2.99, 0.78, 0.78, 0.29, 4.33, 3.01, 3.54, 0.1, 4.85, 4.16, 1.06, 0.91, 2.5],

"z(значение)": [6.81, 7.17, 7.35, 8.07, 8.34, 8.37, 8.73, 8.91, 8.97, 8.94, 9.06, 8.79, 9.12, 9.39, 9.15, None]

}

df = pd.DataFrame(data)

print("Исходные данные и точка прогноза:")

print(tabulate(df, headers='keys', tablefmt='grid', showindex=False, missingval="nan"))

plt.figure(figsize=(12, 8))

observed = df.iloc[:-1]

scatter = plt.scatter(observed["x(м)"], observed["y(м)"],

c=observed["z(значение)"], cmap='viridis',

s=100, edgecolor='k', alpha=0.8, label='Наблюдаемые точки')

prediction = df.iloc[-1:]

plt.scatter(prediction["x(м)"], prediction["y(м)"],

c='red', marker='\*', s=300, label='Точка прогноза (Z)')

for \_, row in observed.iterrows():

plt.text(row["x(м)"] + 0.1, row["y(м)"] + 0.1,

f'№{row["№ точки"]}: {row["z(значение)"]}',

fontsize=8, ha='left', va='bottom')

plt.colorbar(scatter, label='Значение z')

plt.xlabel('x (м)', fontsize=12)

plt.ylabel('y (м)', fontsize=12)

plt.title('График исходных данных и точка прогноза', pad=20)

plt.grid(ls=':', alpha=0.5)

plt.legend(loc='upper right')

plt.xlim(-1, 11)

plt.ylim(0, 5.2)

plt.tight\_layout()

plt.show()

#эмпирическая вариограмма

observed = df.iloc[:-1] # исключаем точку прогноза

coords = observed[["x(м)", "y(м)"]].values

z\_values = observed["z(значение)"].values

#вычисляем матрицу расстояний

dist\_matrix = squareform(pdist(coords, 'euclidean'))

#вычисляем матрицу квадратов разностей z

z\_diff\_matrix = np.zeros\_like(dist\_matrix)

for i in range(len(z\_values)):

for j in range(len(z\_values)):

z\_diff\_matrix[i, j] = (z\_values[i] - z\_values[j]) \*\* 2

#автоматическое разбиение на 10 интервалов

max\_distance = np.max(dist\_matrix)

lag\_bins = np.linspace(0, max\_distance, 10)

gamma = []

h\_lags = []

n\_pairs = []

for i in range(len(lag\_bins) - 1):

h\_min = lag\_bins[i]

h\_max = lag\_bins[i + 1]

mask = (dist\_matrix > h\_min) & (dist\_matrix <= h\_max)

N\_h = np.sum(mask) / 2 # учитываем только верхний треугольник матрицы

if N\_h > 0:

gamma\_h = np.sum(z\_diff\_matrix[mask]) / (2 \* N\_h)

gamma.append(gamma\_h)

h\_lags.append((h\_min + h\_max) / 2)

n\_pairs.append(N\_h)

table\_data = []

for i in range(len(h\_lags)):

table\_data.append([round(h\_lags[i], 3), int(n\_pairs[i]), round(gamma[i], 5)])

print(tabulate(table\_data, headers=["Лаг (h)", "Кол-во пар", "γ(h)"], tablefmt="grid"))

#график вариограммы

plt.figure(figsize=(10, 6))

plt.plot(h\_lags, gamma, 'bo-', label='Эмпирическая вариограмма')

plt.xlabel('Расстояние (h), м', fontsize=12)

plt.ylabel('γ(h)', fontsize=12)

plt.title('Эмпирическая вариограмма', pad=20)

plt.grid(ls=':', alpha=0.5)

plt.legend()

plt.show()

#кубическая модель

def cubic\_model(h, a0, a1, a2, a3):

return a0 + a1\*h + a2\*h\*\*2 + a3\*h\*\*3

popt, pcov = curve\_fit(cubic\_model, h\_lags, gamma)

a0, a1, a2, a3 = popt

print(f"Кубическая модель: γ(h) = {a0:.3f} + {a1:.3f}h + {a2:.3f}h² + {a3:.3f}h³")

#проверяем условие отсутствия касания с линией дисперсии

sigma\_sq = np.var(z\_values)

print(f"Дисперсия исходных данных = {sigma\_sq:.2f}")

#проверяем производные в этих точках

roots = np.roots([a3, a2, a1, a0 - sigma\_sq])

real\_roots = roots[np.isreal(roots)].real

for h\_root in real\_roots:

derivative = a1 + 2\*a2\*h\_root + 3\*a3\*h\_root\*\*2

if np.abs(derivative) < 1e-6:

print("Модель касается линии дисперсии")

#радиус влияния

positive\_roots = real\_roots[real\_roots > 0]

if len(positive\_roots) > 0:

range\_est = min(positive\_roots)

print(f"Радиус влияния: {range\_est:.2f} м")

else:

print("Модель не достигает дисперсии в пределах рассмотренных лагов.")

plt.figure(figsize=(10, 6))

plt.plot(h\_lags, gamma, 'bo', label='Эмпирическая вариограмма')

plt.plot(h\_pred, cubic\_model(h\_pred, \*popt), 'r-', label='Кубическая модель')

plt.xlabel('Расстояние (h), м')

plt.ylabel('γ(h)')

plt.legend()

plt.grid(True)

plt.show()

#считаем весовые коэффициенты

target\_point = np.array([4.0, 2.5]) #точка Z

#расстояния от точек до Z

for i in range(len(coords)):

dist\_to\_target = np.array([np.linalg.norm(coords[i] - target\_point)])

gamma\_target = cubic\_model(dist\_to\_target, a0, a1, a2, a3)

#ковариационная функция: C(h) = D - γ(h)

def covariance(h):

return sigma\_sq - cubic\_model(h, a0, a1, a2, a3)

#матрица А

n = len(coords)

A = np.zeros((n + 1, n + 1))

for i in range(n):

for j in range(n):

A[i, j] = covariance(dist\_matrix[i, j])

#условие Σp\_j = 1

A[-1, :n] = 1.0

A[:n, -1] = 1.0

#правая часть

b = np.zeros(n + 1)

b[:n] = covariance(dist\_to\_target)

b[-1] = 1.0

solution = solve(A, b)

p\_j = solution[:n]

mu = solution[-1]

#прогноз и ошибка

z\_pred = np.sum(p\_j \* z\_values)

kriging\_variance = sigma\_sq - np.sum(p\_j \* covariance(dist\_to\_target))

df.loc[df["№ точки"] == "Z (прогноз)", "z(значение)"] = z\_pred

print("Весовые коэффициенты p\_j:")

for i in range(n):

print(f"p\_{i+1} = {p\_j[i]:.6f}")

print(f"\nМножитель Лагранжа μ = {mu:.6f}")

print(f"\nПрогнозируемое значение в точке Z: {z\_pred:.6f}")

print(f"Дисперсия ошибки кригинга: {kriging\_variance:.6f}")

print(f"\nСумма весов: {np.sum(p\_j):.6f}")

print("\nИтоговые данные с прогнозом:")

print(tabulate(df, headers='keys', tablefmt='grid', showindex=False, missingval="nan"))