**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

**Кафедра теории вероятностей и математической статистики**

**ОТЧЕТ**

по лабораторной работе №4

«Исследование статистических свойств оценок семивариограммы»

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.** Исследовать статистические свойства классической и робастной оценок семивариограммы гауссовского случайного процесса.

Пусть последовательных, полученных через равные промежутки времени наблюдений за случайным процессом. В качестве классической оценки семивариограммы рассмотрим статистику вида:

Положим

В качестве робастной оценки семивариограммы рассмотрим статистику:

Положим

1. Вычислить оценки семивариограммы и представить их графически.

Необходимо:

* Смоделировать наблюдений за гауссовским стационарным случайным процессом с непрерывным временем и известной ковариационной функцией.
* Найти вид семивариограммы процесса. Построить графики наблюдений, ковариационной функции и семивариограммы.
* Вычислить классическую и робастную оценки семивариограммы. На одном графике представить классическую, робастную оценки и истинную семивариограмму.
* Вычислить отклонения и для Сделать вывод. Отобразить графически.

1. Исследовать оценки семивариограммы на состоятельность в среднеквадратическом смысле.

Необходимо:

* Смоделировать 10 наборов наблюдений за гауссовским стационарным случайным процессом с непрерывным временем и известной ковариационной функцией. Число наблюдений n, соответствующих каждому набору, выбрать из таблицы 5 согласно варианту.
* оделировать енки семивариограммы на состоятельность в среднеквадратическом смысле. известной ковариационной функцией Вычислить значения дисперсий классической и робастной оценок для требуемых и .

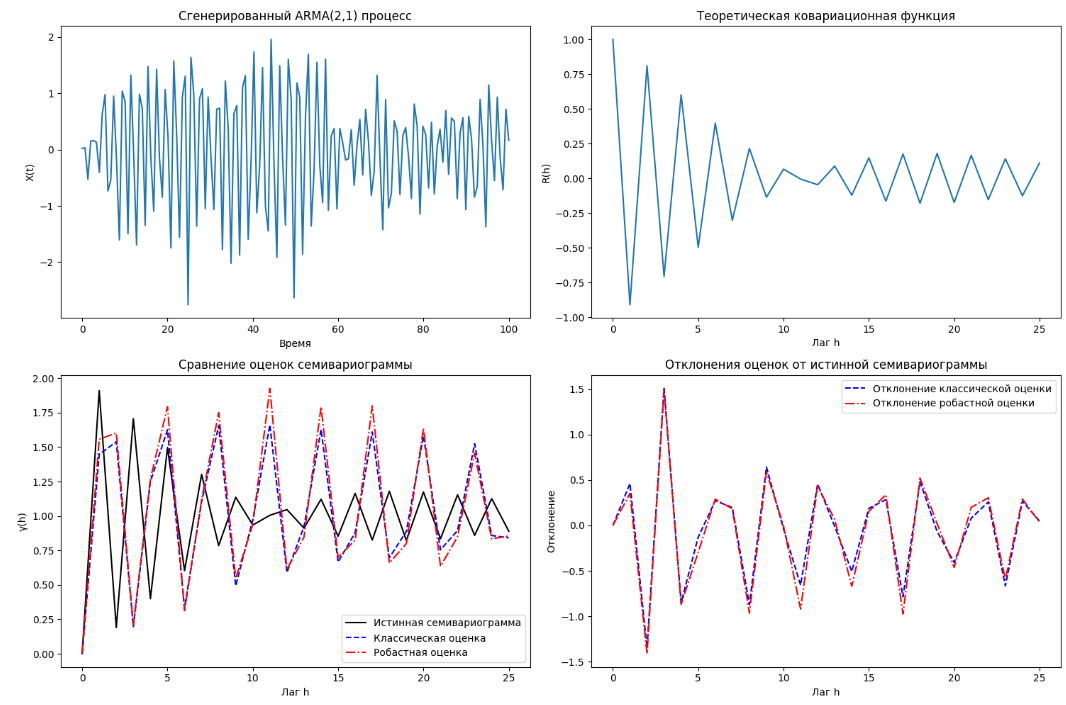
где гамма функция, т. е.

* Для каждого фиксированного из таблицы вариантов построить графики зависимостей дисперсий и от количества наблюдений , представить таблицы соответствующих зависимостей. Сделать сравнительный анализ и вывод о состоятельности оценок.

**Часть первая.**

**Исходные данные (алгоритм выполнения).** Смоделируем наблюдений за гауссовским стационарным случайным процессом с непрерывным временем и известной ковариационной функцией Алгоритм моделирования и результаты возьмем из лабораторной работы №1.

Найдем аналитический вид семивариограммы процесса: Графики наблюдений, ковариационной функции:

**

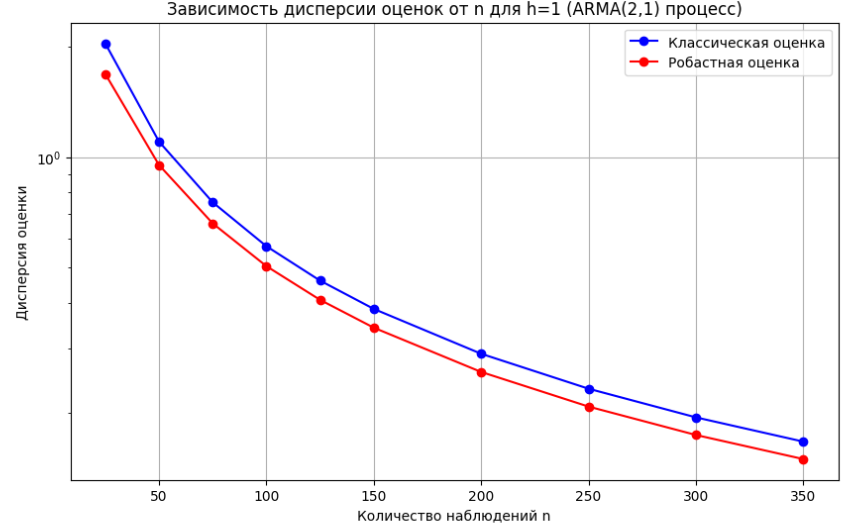
На графиках выше также представлены графики классической, робастной и истинной семивариограммы, а также отклонения и для

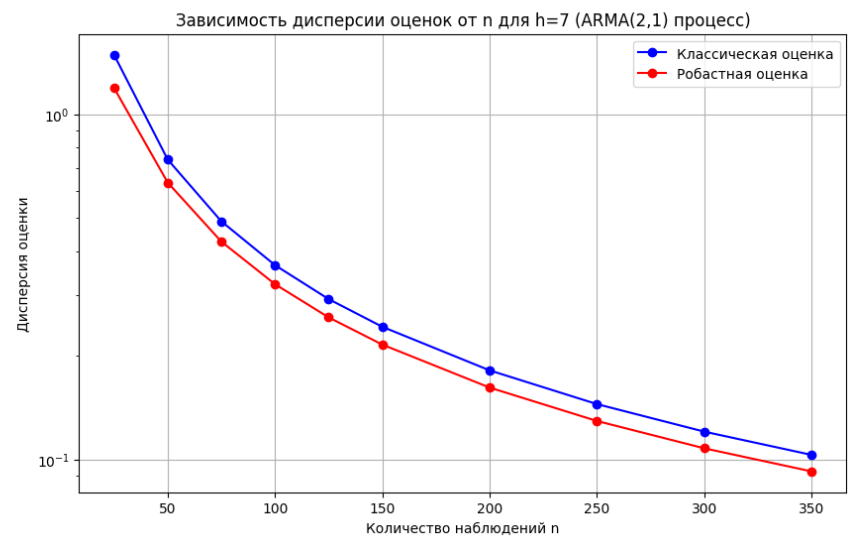
На графике сравнения истинной семивариограммы, классической и робастной оценок видно, что обе оценки стремятся следовать истинной семивариограмме, однако могут иметь различия в некоторых участках. График отклонений показывает, что обе оценки иногда значительно отклоняются от истинной семивариограммы, особенно в областях, где влияют выбросы.

Исследуем оценки семивариограммы на состоятельность в среднеквадратическом. Для этого смоделируем 10 наборов наблюдений за гауссовским стационарным СП с непрерывным временем и известной ковариационной функцией. Число наблюдений: . Лаги:

Вычислим значения дисперсий классической и робастной оценок для требуемых и . Результаты будут представлены ниже, в таблице зависимостей.

Построим графики зависимостей дисперсий от количества наблюдений для каждого фиксированного :





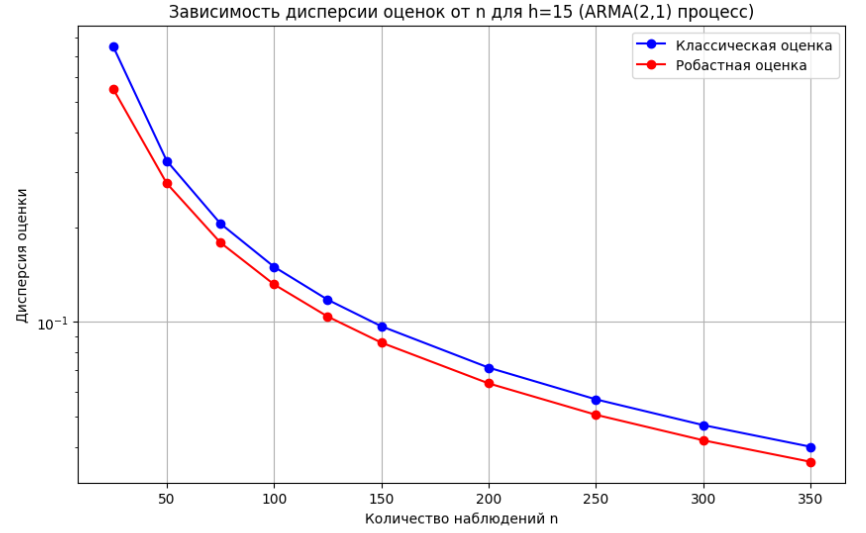
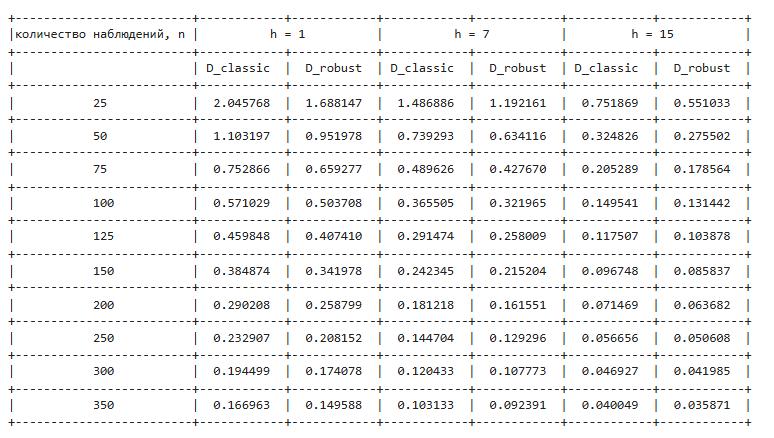


Таблица соответствующих зависимостей:



Для всех лагов (h = 1, 7, 15) наблюдается монотонное убывание дисперсий обеих оценок с ростом объема выборки n. Это соответствует теоретическому свойству состоятельности оценок. Робастная оценка имеет меньшую дисперсию, чем классическая, на всех n и h, это указывает на большую устойчивость робастной оценки к вариациям данных, особенно при малых n.

**Вывод.** Робастная оценка семивариограммы является предпочтительной в ситуациях, когда данные содержат выбросы или шум, так как она обеспечивает более надежные результаты. Классическая оценка может быть применима в условиях, когда данные достаточно чистые и не содержат значительных аномалий.

Обе оценки являются состоятельными, так как их дисперсии стремятся к нулю с ростом *n*. Это означает, что они дают точные результаты при больших объемах данных. Для малых n (n <100) предпочтительнее использовать робастную оценку, так как она дает меньшую дисперсию и более точное приближение к истинной семивариограмме. Для больших n (n ≥ 200) можно применять классическую оценку, но робастная остается более надежной при наличии шума в данных.

**Листинг программы**

Первая часть:

import numpy as np

import matplotlib.pyplot as plt

# 1. Теоретическая ковариационная функция

def theoretical\_cov(h, D=1.0, alpha=0.15, beta=3.0):

return D \* np.exp(-alpha\*np.abs(h)) \* np.cos(beta\*h)

# 2. Генерация процесса методом ARMA(2,1)

def generate\_arma\_process(n=150, T=100.0, D=1.0, alpha=0.15, beta=3.0):

dt = T / n

gamma = alpha \* dt

gamma0 = beta \* dt

alpha\_0 = np.exp(-gamma) \* (np.exp(-2\*gamma) - 1) \* np.cos(gamma0)

alpha1 = 1 - np.exp(-4\*gamma)

a0 = np.sqrt(D) \* alpha

a1 = np.sqrt(D) \* (alpha\_0 / alpha)

b1 = 2 \* np.exp(-gamma) \* np.cos(gamma0)

b2 = -np.exp(-2 \* gamma)

# Генерация белого шума

np.random.seed(43)

x = np.random.normal(0, 1, n)

# Инициализация процесса

xi = np.zeros(n)

xi[0] = a0 \* x[0]

xi[1] = a0 \* x[1] + a1 \* x[0] + b1 \* xi[0]

# Рекуррентное вычисление

for k in range(2, n):

xi[k] = a0 \* x[k] + a1 \* x[k-1] + b1 \* xi[k-1] + b2 \* xi[k-2]

return xi

# 3. Истинная семивариограмма

def true\_semivariogram(h, D=1.0, alpha=0.15, beta=3.0):

return D \* (1 - np.exp(-alpha\*np.abs(h)) \* np.cos(beta\*h))

# 4. Классическая оценка семивариограммы

def classical\_semivariogram\_estimate(X, h\_max):

n = len(X)

gamma\_hat = np.zeros(h\_max + 1)

for h in range(h\_max + 1):

if h < n:

diffs = (X[h:] - X[:n-h])\*\*2

gamma\_hat[h] = np.sum(diffs) / (2 \* (n - h))

else:

gamma\_hat[h] = 0

return gamma\_hat

# 5. Робастная оценка семивариограммы

def robust\_semivariogram\_estimate(X, h\_max):

n = len(X)

gamma\_hat\_rob = np.zeros(h\_max + 1)

for h in range(h\_max + 1):

if h < n:

abs\_diffs = np.abs(X[h:] - X[:n-h])

term1 = np.mean(abs\_diffs\*\*0.5)

denominator = 2 \* (0.457 + 0.494/(n-h) + 0.045/(n-h)\*\*2)

gamma\_hat\_rob[h] = (term1\*\*4) / denominator

else:

gamma\_hat\_rob[h] = 0

return gamma\_hat\_rob

# Параметры моделирования

n = 150

h\_max = 25 # максимальный лаг для оценки

D = 1.0

alpha = 0.085

beta = 3.0

T = 100.0

# Моделирование процесса

X = generate\_arma\_process(n, T, D, alpha, beta)

# Истинная семивариограмма

h\_values = np.arange(h\_max + 1)

true\_gamma = [true\_semivariogram(h, D, alpha, beta) for h in h\_values]

# Оценки семивариограммы

gamma\_hat\_classic = classical\_semivariogram\_estimate(X, h\_max)

gamma\_hat\_robust = robust\_semivariogram\_estimate(X, h\_max)

# Визуализация

plt.figure(figsize=(15, 10))

# График наблюдений процесса

plt.subplot(2, 2, 1)

plt.plot(np.linspace(0, T, n), X)

plt.title('Сгенерированный ARMA(2,1) процесс')

plt.xlabel('Время')

plt.ylabel('X(t)')

# График ковариационной функции

plt.subplot(2, 2, 2)

cov\_values = [theoretical\_cov(h, D, alpha, beta) for h in h\_values]

plt.plot(h\_values, cov\_values)

plt.title('Теоретическая ковариационная функция')

plt.xlabel('Лаг h')

plt.ylabel('R(h)')

# График семивариограмм

plt.subplot(2, 2, 3)

plt.plot(h\_values, true\_gamma, 'k-', label='Истинная семивариограмма')

plt.plot(h\_values, gamma\_hat\_classic, 'b--', label='Классическая оценка')

plt.plot(h\_values, gamma\_hat\_robust, 'r-.', label='Робастная оценка')

plt.title('Сравнение оценок семивариограммы')

plt.xlabel('Лаг h')

plt.ylabel('γ(h)')

plt.legend()

# График отклонений

deviation\_classic = np.array(true\_gamma) - gamma\_hat\_classic

deviation\_robust = np.array(true\_gamma) - gamma\_hat\_robust

plt.subplot(2, 2, 4)

plt.plot(h\_values, deviation\_classic, 'b--', label='Отклонение классической оценки')

plt.plot(h\_values, deviation\_robust, 'r-.', label='Отклонение робастной оценки')

plt.title('Отклонения оценок от истинной семивариограммы')

plt.xlabel('Лаг h')

plt.ylabel('Отклонение')

plt.legend()

plt.tight\_layout()

plt.show()

Вторая часть:

import numpy as np

import matplotlib.pyplot as plt

from scipy.special import gamma

# 1. Функции для моделирования процесса и вычисления семивариограммы

def theoretical\_cov(h, D=1.0, alpha=0.15, beta=3.0):

return D \* np.exp(-alpha\*np.abs(h)) \* np.cos(beta\*h)

def generate\_arma\_process(n=150, T=100.0, D=1.0, alpha=0.15, beta=3.0):

dt = T / n

gamma\_val = alpha \* dt

gamma0 = beta \* dt

alpha0 = np.exp(-gamma\_val) \* (np.exp(-2\*gamma\_val) - 1) \* np.cos(gamma0)

alpha1 = 1 - np.exp(-4\*gamma\_val)

a0 = np.sqrt(D) \* alpha

a1 = np.sqrt(D) \* (alpha0 / alpha)

b1 = 2 \* np.exp(-gamma\_val) \* np.cos(gamma0)

b2 = -np.exp(-2\*gamma\_val)

np.random.seed(42)

x = np.random.normal(0, 1, n)

xi = np.zeros(n)

xi[0] = a0 \* x[0]

xi[1] = a0 \* x[1] + a1 \* x[0] + b1 \* xi[0]

for k in range(2, n):

xi[k] = a0 \* x[k] + a1 \* x[k-1] + b1 \* xi[k-1] + b2 \* xi[k-2]

return xi

def true\_semivariogram(h, cov\_func):

return cov\_func(0) - cov\_func(h)

# 2. Функции для вычисления дисперсий оценок

def classic\_variance(gamma\_func, n, h):

total = 0.0

for s in range(1, n-h+1):

for t in range(1, n-h+1):

term = gamma\_func(abs(s-t + h)) + gamma\_func(abs(s-t - h)) - 2\*gamma\_func(abs(s-t))

total += term\*\*2

return total / (2\*(n-h)\*\*2)

def robust\_variance(gamma\_func, n, h):

gamma\_34 = gamma(3/4)

pi = np.pi

C1 = (np.pi\*\*(-0.5) - gamma\_34\*\*2/pi) \* gamma\_34\*\*6 / pi\*\*3

denominator = (0.457 + 0.494/(n-h) + 0.045/(n-h)\*\*2)\*\*2

total = 0.0

for s in range(1, n-h+1):

for t in range(1, n-h+1):

term = gamma\_func(abs(s-t + h)) + gamma\_func(abs(s-t - h)) - 2\*gamma\_func(abs(s-t))

total += term\*\*2

return (10 \* C1 \* total) / ((n-h)\*\*2 \* denominator)

# 3. Параметры исследования

n\_values = [25, 50, 75, 100, 125, 150, 200, 250, 300, 350]

h\_values = [1, 7, 15]

D = 1.0

alpha = 0.085

beta = 3.0

T = 100.0

# 4. Исследование для каждого h

results = {}

for h in h\_values:

classic\_vars = []

robust\_vars = []

for n in n\_values:

gamma\_func = lambda x: true\_semivariogram(x, lambda y: theoretical\_cov(y, D, alpha, beta))

classic\_var = classic\_variance(gamma\_func, n, h)

robust\_var = robust\_variance(gamma\_func, n, h)

classic\_vars.append(classic\_var)

robust\_vars.append(robust\_var)

results[h] = {'D\_classic': classic\_vars, 'D\_robust': robust\_vars}

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

plt.plot(n\_values, classic\_vars, 'bo-', label='Классическая оценка')

plt.plot(n\_values, robust\_vars, 'ro-', label='Робастная оценка')

plt.title(f'Зависимость дисперсии оценок от n для h={h} (ARMA(2,1) процесс)')

plt.xlabel('Количество наблюдений n')

plt.ylabel('Дисперсия оценки')

plt.yscale('log') # Логарифмическая шкала для лучшей визуализации

plt.grid(True)

plt.legend()

plt.show()

#5.Вывод результатов

n\_width = 25 # Ширина столбца с n

d\_width = 12 # Ширина столбцов с дисперсиями

# Функция для создания горизонтальной линии

def make\_line():

line = "+" + "-" \* n\_width + "+"

for \_ in h\_values:

line += "-" \* d\_width + "+" + "-" \* d\_width + "+"

return line

# Верхняя граница таблицы

print("\n" + make\_line())

# Заголовок с лагами

header = f"|{'количество наблюдений, n':^{n\_width}}|"

for h in h\_values:

header += f"{f'h = {h}':^{2\*d\_width+1}}|"

print(header)

# Линия под заголовком

print(make\_line())

# Подзаголовки с типами дисперсий

subheader = f"|{' ' \* n\_width}|"

for \_ in h\_values:

subheader += f"{'D\_classic':^{d\_width}}|{'D\_robust':^{d\_width}}|"

print(subheader)

# Основные данные

print(make\_line())

for i, n in enumerate(n\_values):

row = f"|{n:^{n\_width}}|"

for h in h\_values:

classic = f"{results[h]['D\_classic'][i]:.6f}"

robust = f"{results[h]['D\_robust'][i]:.6f}"

row += f"{classic:^{d\_width}}|{robust:^{d\_width}}|"

print(row)

print(make\_line())

**Часть вторая.**

**Постановка задачи:** Исследовать статистические свойства классической и робастной оценок семивариограммы гауссовского случайного процесса в случае нерегулярной сети наблюдения. Вид нерегулярности выбрать самостоятельно.

**Исходные данные (алгоритм выполнения):**

Нерегулярность я решила реализовать с помощью лакунарности (пропуска некоторых данных).

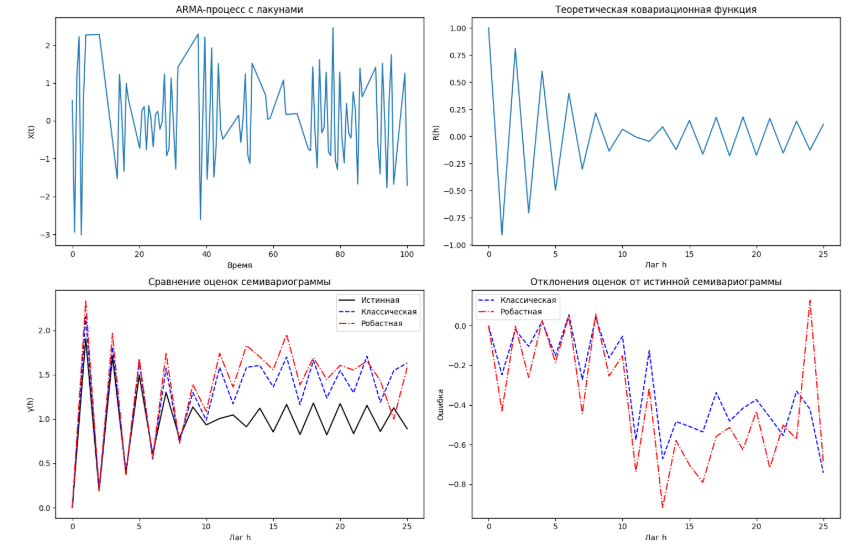
Смоделируем наблюдений за гауссовским стационарным случайным процессом в нерегулярной сети наблюдения с ковариационной функцией Для моделирования будем удалять точки внутри некоторых интервалов (выбранных случайно), строить ковариационную матрицу для расстояний между получившимися после удаления точками и генерировать гауссовский процесс.

Классическая оценка:

Где — множество всех пар наблюдений, у которых точный временной лаг (расстояние между измерениями) равен h, |N(h)| — количество таких пар.

Робастная оценка:

Найдем аналитический вид семивариограммы процесса: Графики наблюдений, ковариационной функции:



На графиках выше также представлены графики классической, робастной и истинной семивариограммы, а также отклонения и для

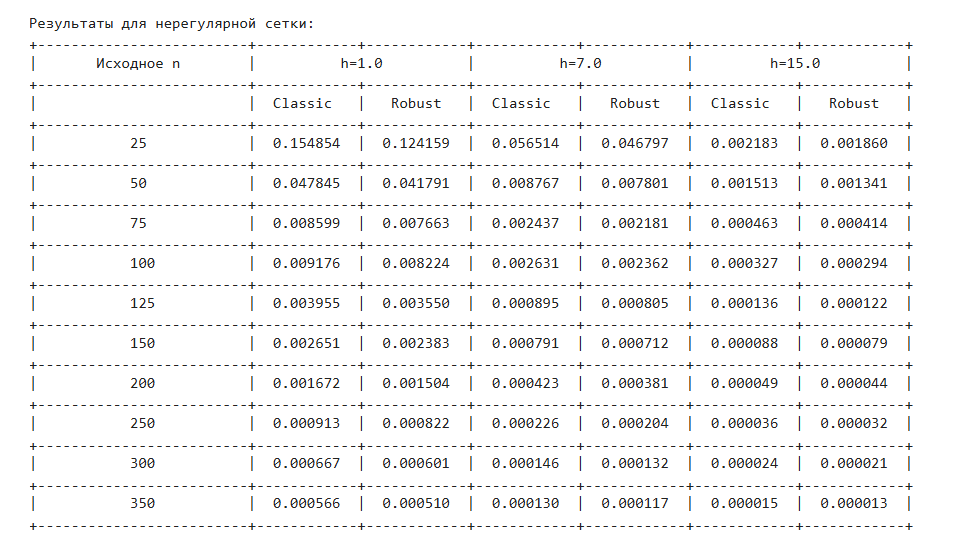
На графике сравнения истинной семивариограммы, классической и робастной оценок видно, что обе оценки стремятся следовать истинной семивариограмме, однако могут иметь различия в некоторых участках. График отклонений показывает, что обе оценки иногда значительно отклоняются от истинной семивариограммы, особенно в областях, где влияют выбросы.

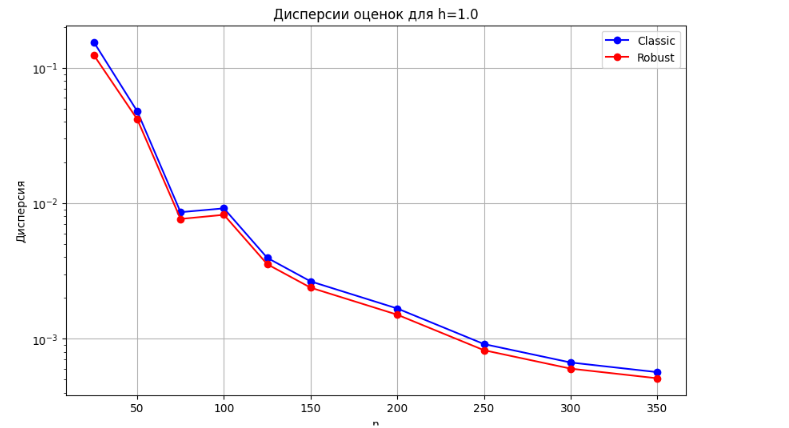
Исследуем оценки семивариограммы на состоятельность в среднеквадратическом. Для этого смоделируем 10 наборов наблюдений за гауссовским СП с известной ковариационной функцией. Число наблюдений: . Лаги:

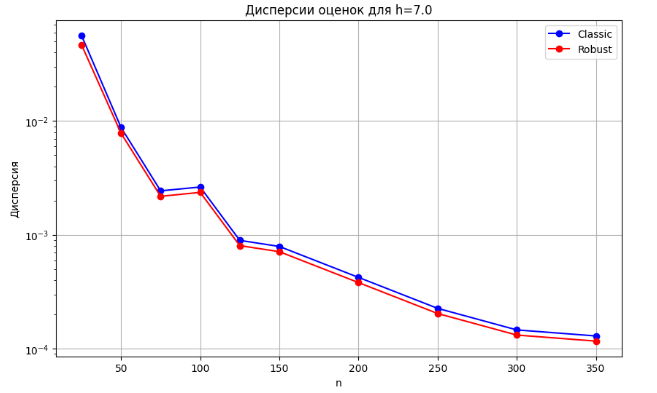
где гамма функция, т. е.

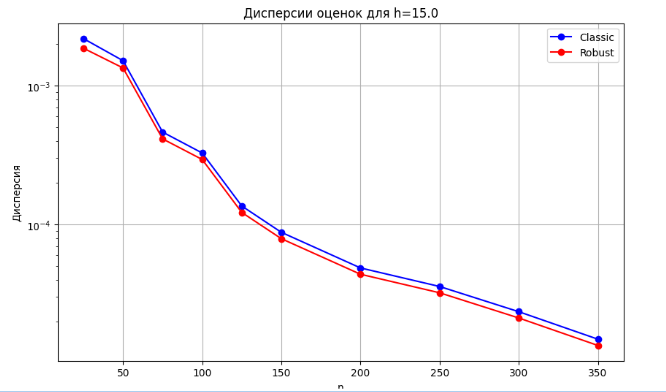
Вычислим значения дисперсий классической и робастной оценок для требуемых и . Результаты будут представлены ниже, в таблице зависимостей.

Построим графики зависимостей дисперсий от количества наблюдений для каждого фиксированного :



****

****

****

Для всех лагов (h = 1, 7, 15) наблюдается монотонное убывание дисперсий обеих оценок с ростом объема выборки n. Это соответствует теоретическому свойству состоятельности оценок. Робастная оценка имеет меньшую дисперсию, чем классическая, на всех n и h, это указывает на большую устойчивость робастной оценки к вариациям данных, особенно при малых n.

**Вывод.** Робастная оценка семивариограммы является предпочтительной в ситуациях, когда данные содержат выбросы или шум, так как она обеспечивает более надежные результаты. Классическая оценка может быть применима в условиях, когда данные достаточно чистые и не содержат значительных аномалий.

Обе оценки являются состоятельными, так как их дисперсии стремятся к нулю с ростом *n*. Это означает, что они дают точные результаты при больших объемах данных. Для малых n (n <100) предпочтительнее использовать робастную оценку, так как она дает меньшую дисперсию и более точное приближение к истинной семивариограмме. Для больших n (n ≥ 200) можно применять классическую оценку, но робастная остается более надежной при наличии шума в данных.

Листинг программы:

#первая часть

import numpy as np

import matplotlib.pyplot as plt

def theoretical\_cov(h, D=1.0, alpha=0.15, beta=3.0):

return D \* np.exp(-alpha \* np.abs(h)) \* np.cos(beta \* h)

def true\_semivariogram(h, D=1.0, alpha=0.15, beta=3.0):

return D \* (1 - np.exp(-alpha \* np.abs(h)) \* np.cos(beta \* h))

def generate\_arma\_process(n=150, T=100.0, D=1.0, alpha=0.15, beta=3.0):

dt = T / n

gamma = alpha \* dt

gamma0 = beta \* dt

alpha\_0 = np.exp(-gamma) \* (np.exp(-2\*gamma) - 1) \* np.cos(gamma0)

alpha1 = 1 - np.exp(-4\*gamma)

a0 = np.sqrt(D) \* alpha

a1 = np.sqrt(D) \* (alpha\_0 / alpha)

b1 = 2 \* np.exp(-gamma) \* np.cos(gamma0)

b2 = -np.exp(-2 \* gamma)

np.random.seed(43)

x = np.random.normal(0, 1, n)

xi = np.zeros(n)

xi[0] = a0 \* x[0]

xi[1] = a0 \* x[1] + a1 \* x[0] + b1 \* xi[0]

for k in range(2, n):

xi[k] = a0 \* x[k] + a1 \* x[k-1] + b1 \* xi[k-1] + b2 \* xi[k-2]

return xi

#добавление пропусков (лакун) в регулярную сетку

def add\_lacunarity(t\_regular, n\_gaps=15, gap\_size=0.03):

t\_irregular = t\_regular.copy()

total\_time = t\_regular[-1] - t\_regular[0]

gap\_size\_abs = gap\_size \* total\_time

for \_ in range(n\_gaps):

gap\_start = np.random.uniform(t\_irregular[0], t\_irregular[-1] - gap\_size\_abs)

mask = (t\_irregular < gap\_start) | (t\_irregular > gap\_start + gap\_size\_abs)

t\_irregular = t\_irregular[mask]

return t\_irregular

#генерирование гауссовского процесса

def generate\_gaussian\_process(t\_points, D=1.0, alpha=0.15, beta=3.0):

n = len(t\_points)

cov\_matrix = np.zeros((n, n))

for i in range(n):

for j in range(n):

cov\_matrix[i, j] = theoretical\_cov(t\_points[i] - t\_points[j], D, alpha, beta)

return np.random.multivariate\_normal(np.zeros(n), cov\_matrix)

#оценки семивариограммы

def irregular\_semivariogram(t, X, h\_max, method='classical'):

n = len(X)

gamma = np.zeros(h\_max + 1)

counts = np.zeros(h\_max + 1)

for i in range(n):

for j in range(i + 1, n):

h = int(np.round(np.abs(t[i] - t[j])))

if h <= h\_max:

if method == 'classical':

gamma[h] += (X[i] - X[j])\*\*2

elif method == 'robust':

gamma[h] += np.abs(X[i] - X[j])\*\*0.5

counts[h] += 1

if method == 'classical':

with np.errstate(divide='ignore', invalid='ignore'):

gamma = np.where(counts > 0, gamma / (2 \* counts), 0)

elif method == 'robust':

for h in range(h\_max + 1):

if counts[h] > 0:

term = gamma[h] / counts[h]

denominator = 2 \* (0.457 + 0.494/counts[h] + 0.045/counts[h]\*\*2)

gamma[h] = (term\*\*4) / denominator

else:

gamma[h] = 0

return gamma

# Параметры

np.random.seed(42)

D = 1.0

alpha = 0.085

beta = 3.0

T = 100.0

n = 150

h\_max = 25

n\_gaps = 15

gap\_size = 0.03

t\_regular = np.linspace(0, T, n)

X\_regular = generate\_arma\_process(n, T, D, alpha, beta)

t\_irregular = add\_lacunarity(t\_regular, n\_gaps, gap\_size)

X\_irregular = generate\_gaussian\_process(t\_irregular, D, alpha, beta)

#теоретическая семивариограмма

h\_values = np.arange(h\_max + 1)

true\_gamma = np.array([true\_semivariogram(h, D, alpha, beta) for h in h\_values])

#оценки семивариограммы

gamma\_irreg\_classic = irregular\_semivariogram(t\_irregular, X\_irregular, h\_max, 'classical')

gamma\_irreg\_robust = irregular\_semivariogram(t\_irregular, X\_irregular, h\_max, 'robust')

#аизуализация

plt.figure(figsize=(15, 10))

plt.subplot(2, 2, 1)

plt.plot(t\_irregular, X\_irregular)

plt.title('ARMA-процесс с лакунами')

plt.xlabel('Время')

plt.ylabel('X(t)')

plt.subplot(2, 2, 2)

cov\_values = [theoretical\_cov(h, D, alpha, beta) for h in h\_values]

plt.plot(h\_values, cov\_values)

plt.title('Теоретическая ковариационная функция')

plt.xlabel('Лаг h')

plt.ylabel('R(h)')

plt.subplot(2, 2, 3)

plt.plot(h\_values, true\_gamma, 'k-', label='Истинная')

plt.plot(h\_values, gamma\_irreg\_classic, 'b--', label='Классическая')

plt.plot(h\_values, gamma\_irreg\_robust, 'r-.', label='Робастная')

plt.title('Сравнение оценок семивариограммы')

plt.xlabel('Лаг h')

plt.ylabel('γ(h)')

plt.legend()

plt.subplot(2, 2, 4)

plt.plot(h\_values, true\_gamma - gamma\_irreg\_classic, 'b--', label='Классическая')

plt.plot(h\_values, true\_gamma - gamma\_irreg\_robust, 'r-.', label='Робастная')

plt.title('Отклонения оценок от истинной семивариограммы')

plt.xlabel('Лаг h')

plt.ylabel('Ошибка')

plt.legend()

plt.tight\_layout()

plt.show()

#вторая часть

import numpy as np

import matplotlib.pyplot as plt

from scipy.special import gamma

# 1. Функции для моделирования процесса и вычисления семивариограммы

def theoretical\_cov(h, D=1.0, alpha=0.15, beta=3.0):

return D \* np.exp(-alpha \* np.abs(h)) \* np.cos(beta \* h)

def true\_semivariogram(h, D=1.0, alpha=0.15, beta=3.0):

return D \* (1 - np.exp(-alpha \* np.abs(h)) \* np.cos(beta \* h))

def generate\_arma\_process(n=150, T=100.0, D=1.0, alpha=0.15, beta=3.0):

dt = T / n

gamma\_val = alpha \* dt

gamma0 = beta \* dt

alpha0 = np.exp(-gamma\_val) \* (np.exp(-2\*gamma\_val) - 1) \* np.cos(gamma0)

a0 = np.sqrt(D) \* alpha

a1 = np.sqrt(D) \* (alpha0 / alpha)

b1 = 2 \* np.exp(-gamma\_val) \* np.cos(gamma0)

b2 = -np.exp(-2 \* gamma\_val)

np.random.seed(43)

x = np.random.normal(0, 1, n)

xi = np.zeros(n)

xi[0] = a0 \* x[0]

xi[1] = a0 \* x[1] + a1 \* x[0] + b1 \* xi[0]

for k in range(2, n):

xi[k] = a0 \* x[k] + a1 \* x[k-1] + b1 \* xi[k-1] + b2 \* xi[k-2]

return xi

def add\_lacunarity(t\_regular, n\_gaps=15, gap\_size=0.03):

t\_irregular = t\_regular.copy()

total\_time = t\_regular[-1] - t\_regular[0]

gap\_size\_abs = gap\_size \* total\_time

for \_ in range(n\_gaps):

gap\_start = np.random.uniform(t\_irregular[0], t\_irregular[-1] - gap\_size\_abs)

mask = (t\_irregular < gap\_start) | (t\_irregular > gap\_start + gap\_size\_abs)

t\_irregular = t\_irregular[mask]

return t\_irregular

def generate\_gaussian\_process(t\_points, D=1.0, alpha=0.15, beta=3.0):

n = len(t\_points)

cov\_matrix = np.zeros((n, n))

for i in range(n):

for j in range(n):

cov\_matrix[i, j] = theoretical\_cov(t\_points[i] - t\_points[j], D, alpha, beta)

return np.random.multivariate\_normal(np.zeros(n), cov\_matrix)

# 2. Функции для вычисления дисперсий оценок

def classic\_variance\_irregular(t\_points, gamma\_func, h, tol=5.0):

pairs = []

n = len(t\_points)

# Находим пары с расстоянием h±tol

for i in range(n):

for j in range(i+1, n):

if abs(abs(t\_points[j] - t\_points[i]) - h) < tol:

pairs.append((i, j))

m = len(pairs)

if m < 2: # Минимум 2 пары для оценки

return np.nan

total = 0.0

for (i, j) in pairs:

delta = abs(t\_points[j] - t\_points[i])

term = gamma\_func(delta + h) + gamma\_func(abs(delta - h)) - 2\*gamma\_func(delta)

total += term\*\*2

return total / (2 \* m\*\*2)

def robust\_variance\_irregular(t\_points, gamma\_func, h, tol=5.0):

pairs = []

n = len(t\_points)

for i in range(n):

for j in range(i+1, n):

if abs(abs(t\_points[j] - t\_points[i]) - h) < tol:

pairs.append((i, j))

m = len(pairs)

if m < 2:

return np.nan

gamma\_34 = gamma(3/4)

pi = np.pi

C1 = (pi\*\*(-0.5) - gamma\_34\*\*2/pi) \* gamma\_34\*\*6 / pi\*\*3

denominator = (0.457 + 0.494/m + 0.045/m\*\*2)\*\*2

total = 0.0

for (i, j) in pairs:

delta = abs(t\_points[j] - t\_points[i])

term = gamma\_func(delta + h) + gamma\_func(abs(delta - h)) - 2\*gamma\_func(delta)

total += term\*\*2

return (10 \* C1 \* total) / (m\*\*2 \* denominator)

# Параметры

n\_values = [25, 50, 75, 100, 125, 150, 200, 250, 300, 350]

h\_values = [1.0, 7.0, 15.0]

D = 1.0

alpha = 0.085

beta = 3.0

T = 100.0

tol = 0.5\*h

results = {}

for h in h\_values:

classic\_vars = []

robust\_vars = []

for n in n\_values:

t\_regular = np.linspace(0, T, n)

t\_irregular = add\_lacunarity(t\_regular, n\_gaps=15, gap\_size=0.03)

gamma\_f = lambda x: true\_semivariogram(x, D, alpha, beta)

classic\_var = classic\_variance\_irregular(t\_irregular, gamma\_f, h, tol)

robust\_var = robust\_variance\_irregular(t\_irregular, gamma\_f, h, tol)

classic\_vars.append(classic\_var)

robust\_vars.append(robust\_var)

results[h] = {'D\_classic': classic\_vars, 'D\_robust': robust\_vars}

#Визуализация

def make\_line(h\_values, n\_width=25, d\_width=12):

line = "+" + "-" \* n\_width + "+"

for \_ in h\_values:

line += "-" \* d\_width + "+" + "-" \* d\_width + "+"

return line

print("\nРезультаты для нерегулярной сетки:")

print(make\_line(h\_values))

header = f"|{'Исходное n':^{25}}|"

for h in h\_values:

header += f"{f'h={h}':^{25}}|"

print(header)

print(make\_line(h\_values))

subheader = f"|{' ' \* 25}|"

for \_ in h\_values:

subheader += f"{'Classic':^{12}}|{'Robust':^{12}}|"

print(subheader)

print(make\_line(h\_values))

for i, n in enumerate(n\_values):

row = f"|{n:^{25}}|"

for h in h\_values:

classic = f"{results[h]['D\_classic'][i]:.6f}" if not np.isnan(results[h]['D\_classic'][i]) else "NaN"

robust = f"{results[h]['D\_robust'][i]:.6f}" if not np.isnan(results[h]['D\_robust'][i]) else "NaN"

row += f"{classic:^{12}}|{robust:^{12}}|"

print(row)

print(make\_line(h\_values))

for h in h\_values:

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

valid\_n = [n for n, v in zip(n\_values, results[h]['D\_classic']) if not np.isnan(v)]

classic\_vals = [v for v in results[h]['D\_classic'] if not np.isnan(v)]

robust\_vals = [v for v in results[h]['D\_robust'] if not np.isnan(v)]

plt.plot(valid\_n, classic\_vals, 'bo-', label='Classic')

plt.plot(valid\_n, robust\_vals, 'ro-', label='Robust')

plt.title(f'Дисперсии оценок для h={h}')

plt.xlabel('n')

plt.ylabel('Дисперсия')

plt.yscale('log')

plt.grid(True)

plt.legend()

plt.show()

**Сравнительный анализ регулярного и нерегулярного случая:**

Основные наблюдения по графикам

1. Регулярный случай:

* Оценки семивариограммы: Обе оценки хорошо соответствуют теоретической кривой, особенно на малых лагах.
* Отклонения: Отклонения оценок от истинной семивариограммы минимальны, причем робастная оценка показывает несколько лучшие результаты на больших лагах.
* Поведение процесса: Временной ряд демонстрирует четкую периодическую структуру, соответствующую параметрам модели.

2. Нерегулярный случай:

* Оценки семивариограммы: Оценки становятся менее точными, особенно на больших лагах. Классическая оценка демонстрирует более сильные колебания.
* Отклонения: Отклонения значительно больше, чем в регулярном случае. Робастная оценка показывает более устойчивое поведение.
* Поведение процесса: Временной ряд имеет пропуски, что приводит к потере информации и ухудшению оценок.

Вывод по дисперсиям оценок

Нерегулярные данные демонстрируют значительно меньшие дисперсии оценок по сравнению с регулярными. Это связано с тем, что в нерегулярном случае лаги вычисляются с округлением, а количество пар точек для каждого лага может сильно варьироваться. Регулярные данные дают более высокие дисперсии, но это объясняется тем, что оценки вычисляются строго по фиксированным лагам и в регулярном случае больше пар точек для каждого лага, что может приводить к накоплению ошибок.

Робастная оценка всегда имеет меньшую дисперсию, чем классическая, как для регулярных, так и для нерегулярных данных. Это ожидаемо, поскольку робастные методы устойчивее к выбросам и шумам. Разница между классической и робастной оценками более выражена в регулярном случае, где дисперсии выше.

Для обоих типов данных, чем больше лаг (h), тем меньше дисперсия. В регулярном случае дисперсия уменьшается с ростом h более плавно, чем в нерегулярном. Чем больше n, тем меньше дисперсия (для обоих методов и обоих типов данных).