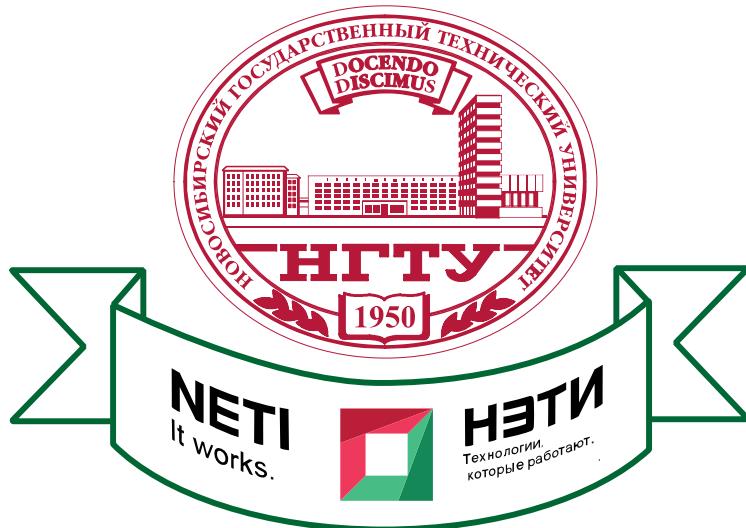


Министерство науки и высшего образования  
Российской Федерации

Федеральное государственное бюджетное  
образовательное учреждение высшего образования

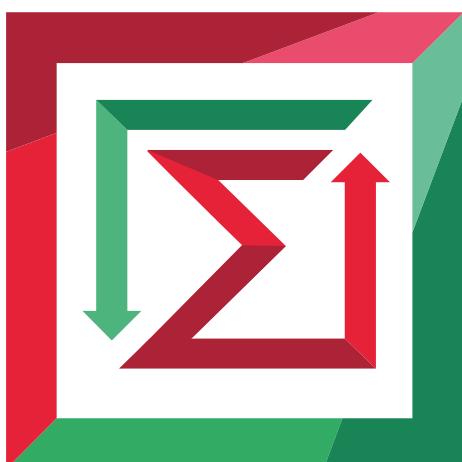
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Теоретической и прикладной информатики

Лабораторная работа № 1  
по дисциплине «Компьютерное моделирование»

### ПРЕДВАРИТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ



Факультет: ПМИ  
Группа: ПМИ-02  
Студент: Сидоров Даниил,  
Дюков Богдан  
Преподаватель: Карманов Виталий Сергеевич

Новосибирск

2026

## 1. Формулировка задания

- Удаление выбросов при помощи диаграмм рассеяния и Isolation forest.
- Фильтрация шума при помощи фильтра Чебышева 2-го рода.
- Сглаживание значений ряда алгоритмом взвешенного скользящего по 3 точкам.
- Проверка на стационарность при помощи теста Дикки-Фуллера и теста Квятковского-Филлипса-Шмидта-Шина.
- Проверка на наличие тренда методом Валлиса и Мура.

## 2. Цели работы

Предварительная обработка и анализ данных.

## 3. Описание выполненных действий

### Удаление выбросов

Произведем удаление выбросов для двух выборок двумя методами.

Получим диаграмму рассеяния:

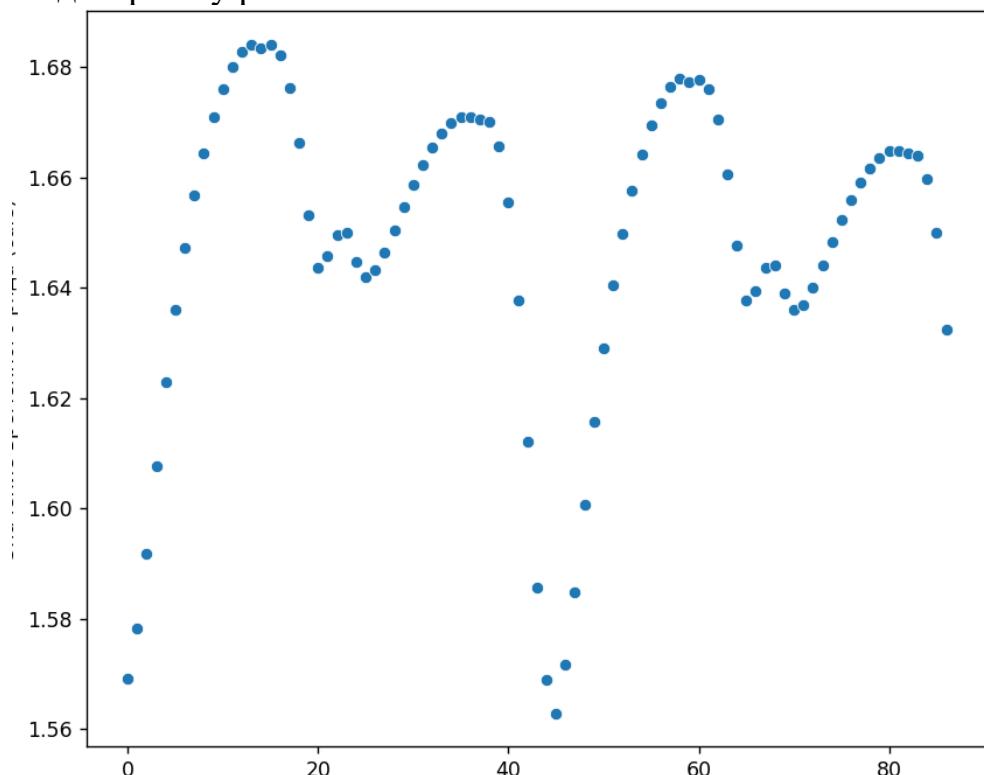
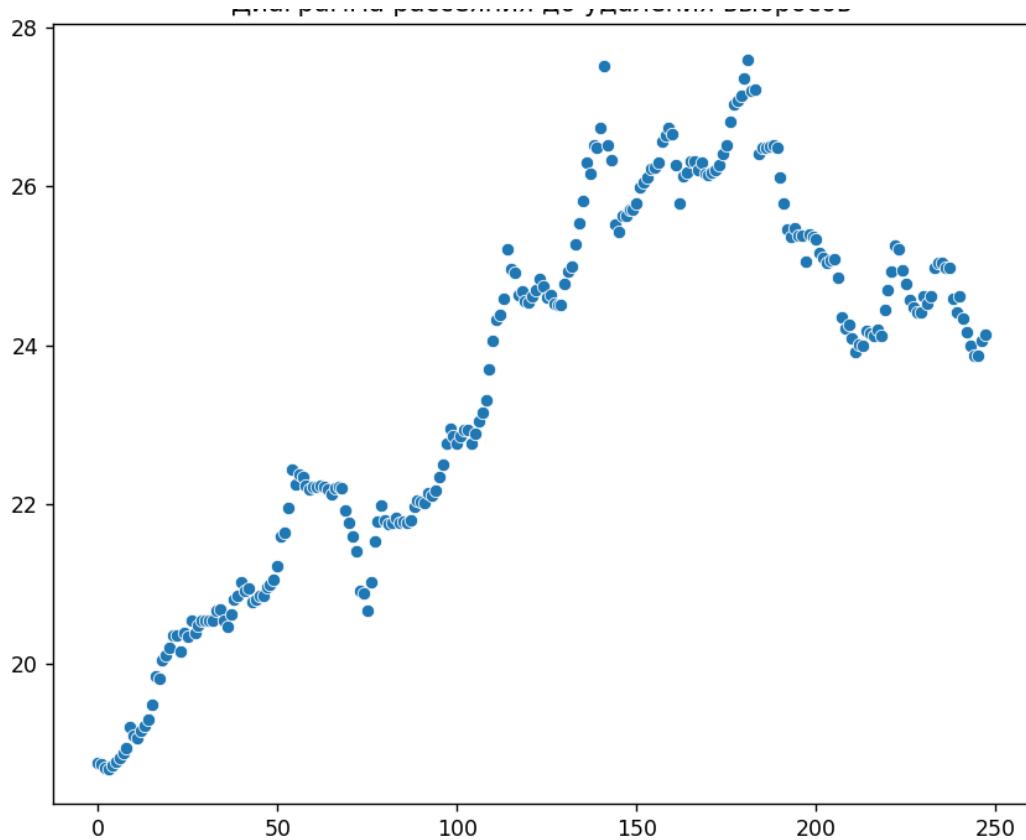


Рисунок 1 – Диаграмма рассеяния для данных-1

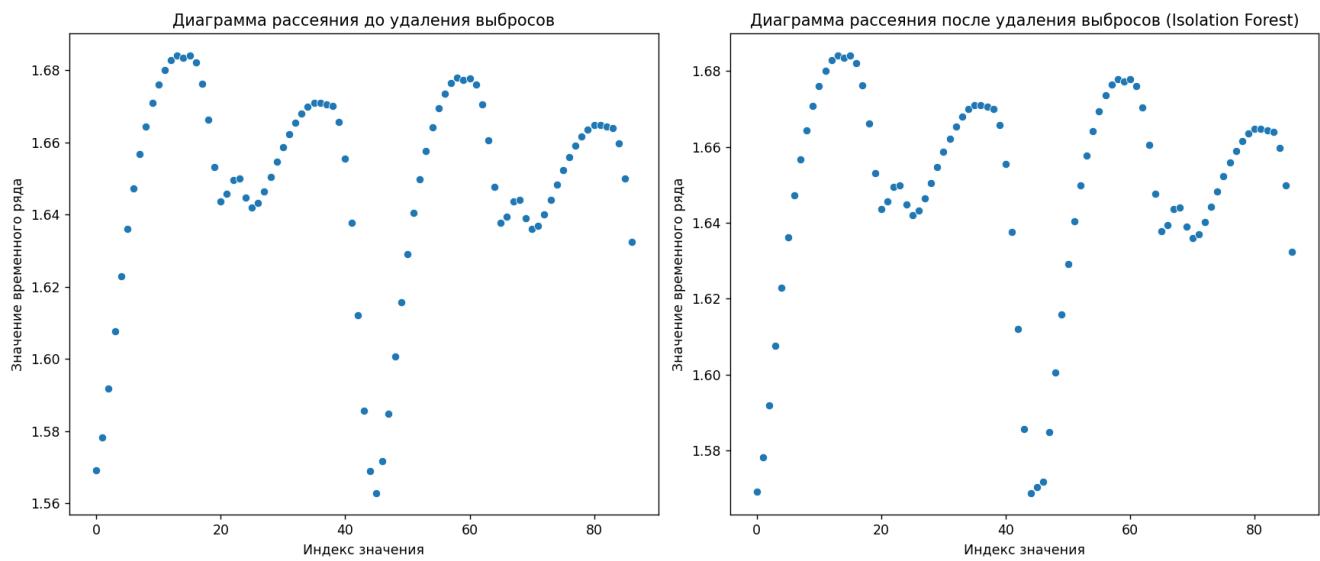
Можем видеть, что явных выбросов нет.



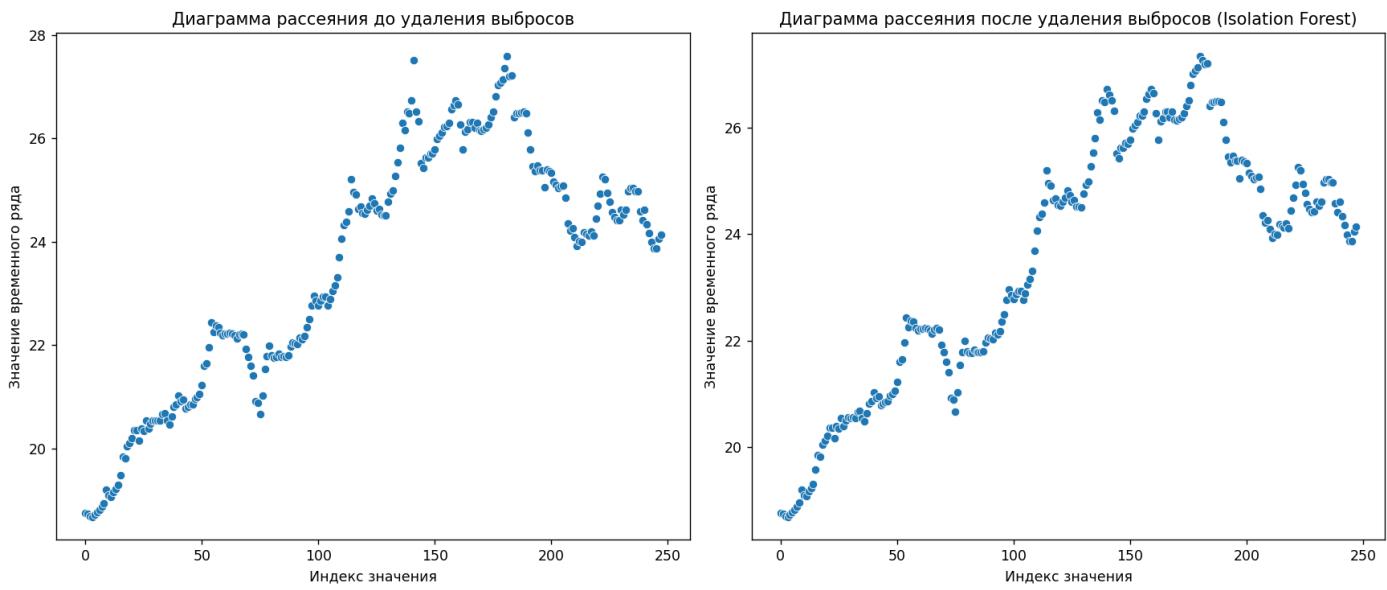
*Рисунок 2 – Диаграмма рассеяния для курса валют*

Можем видеть, что на диаграмме присутствуют точки, отдаленные от других, а значит выбросы есть.

Воспользуемся методом Isolation Forest, который более точно отследит выбросы и удалит их:



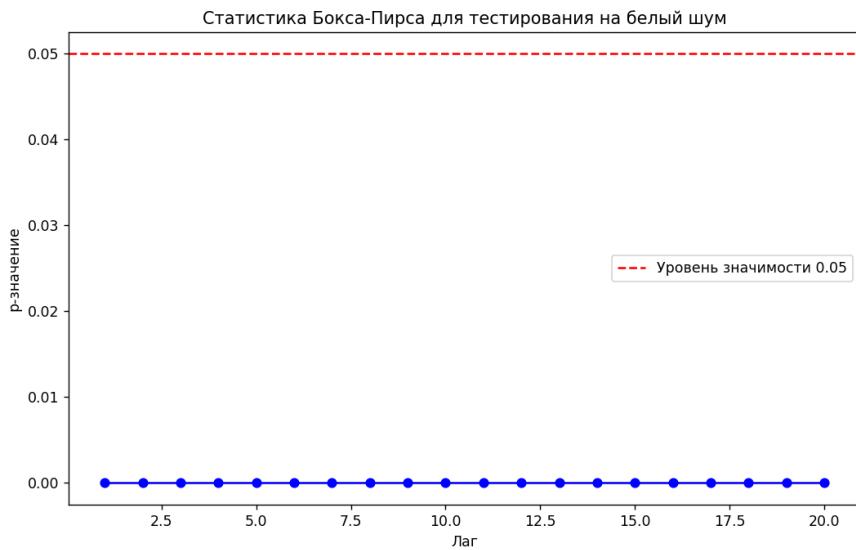
*Рисунок 3 – Удаление выбросов методом Isolation forest для данных-1*



*Рисунок 4 – Удаление выбросов методом Isolation forest для курса валют*

### Фильтрация шума

Вычислим статистику Бокса-Пирса для тестирования на наличие белого шума.



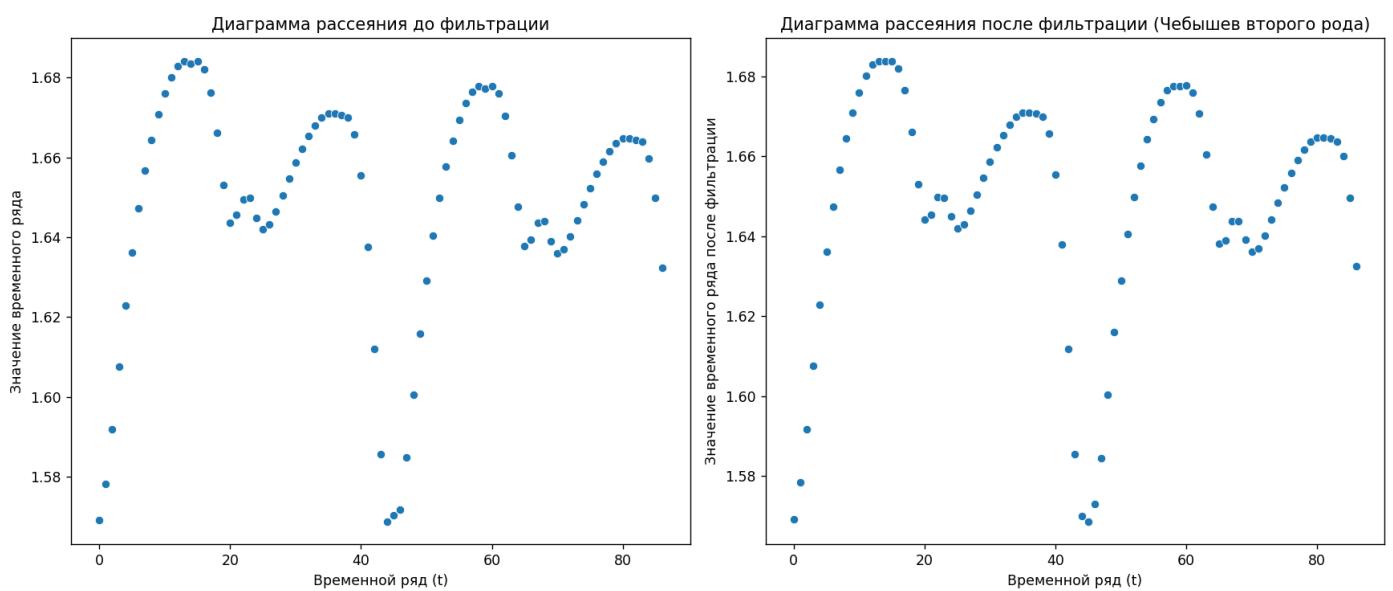
*Рисунок 5 – Статистика Бокса-Пирса для данных-1*

Для данных-1 не наблюдаем белого шума

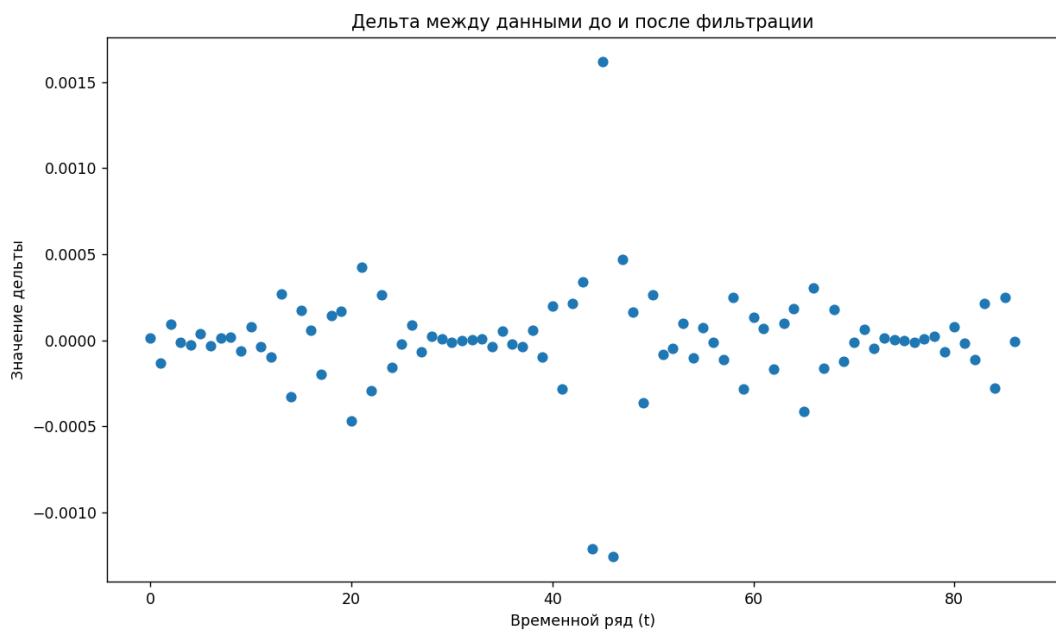


*Рисунок 6 – Статистика Бокса-Пирса для курса валют*

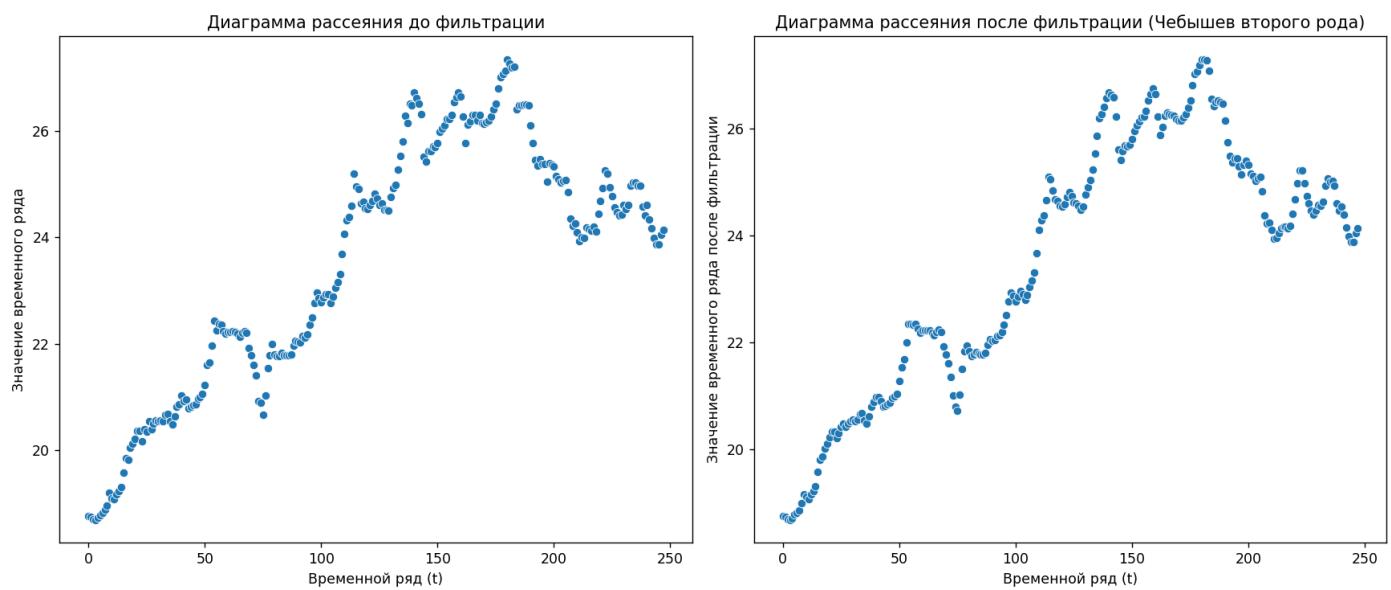
Для курса валют не наблюдаем белого шума. Делаем вывод, что фильтрация для удаления шумов не нужна, но ее можно применить для сглаживания данных:



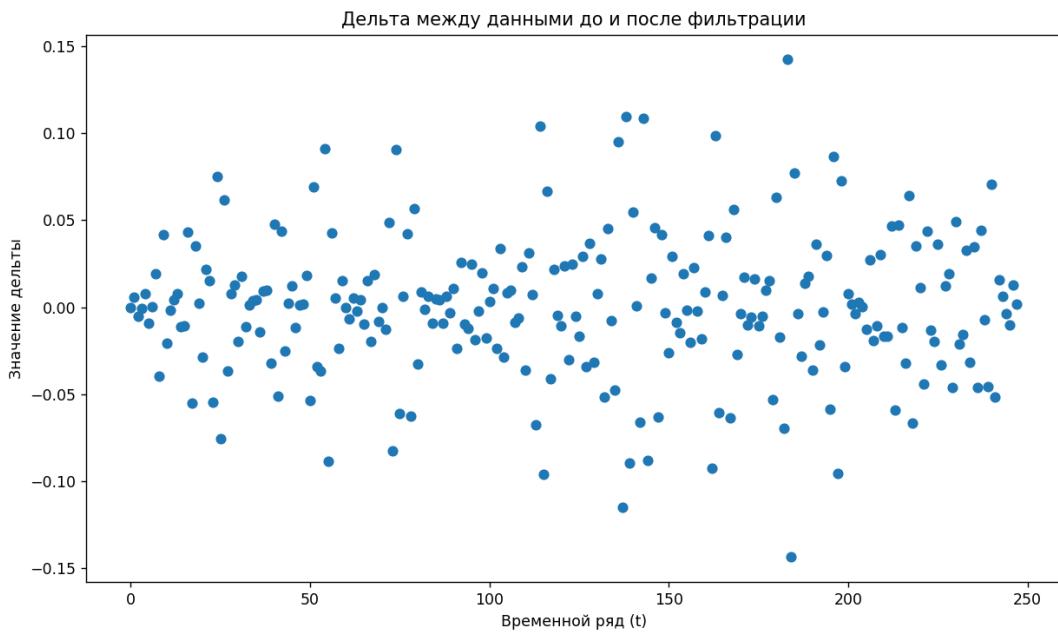
*Рисунок 7 – Фильтрация данных-1*



*Рисунок 8 – дельта между значениями до и после фильтрации для данных-1*



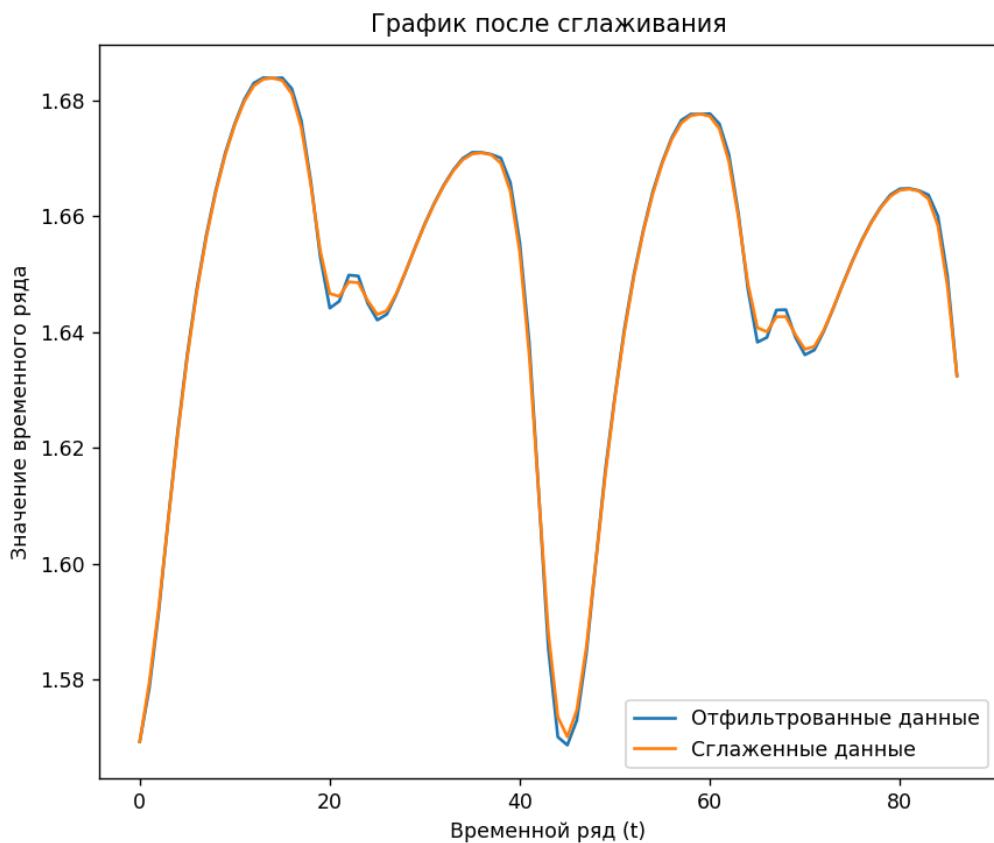
*Рисунок 9 – Фильтрация курса валют*



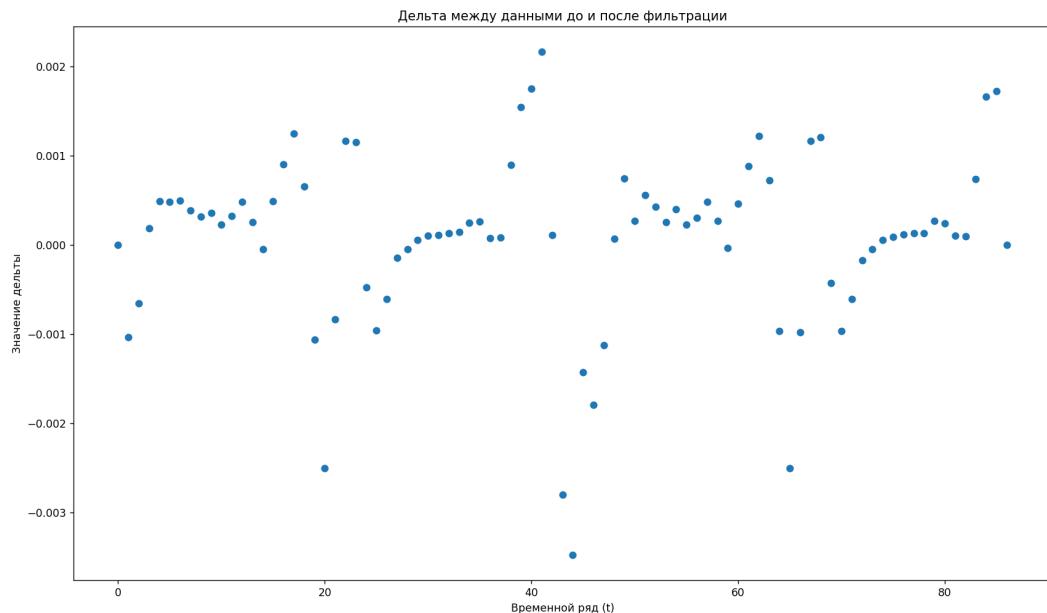
*Рисунок 10 – дельта между значениями до и после фильтрации для курса валют*

### Сглаживание значений ряда

Сгладим наши временные ряды:



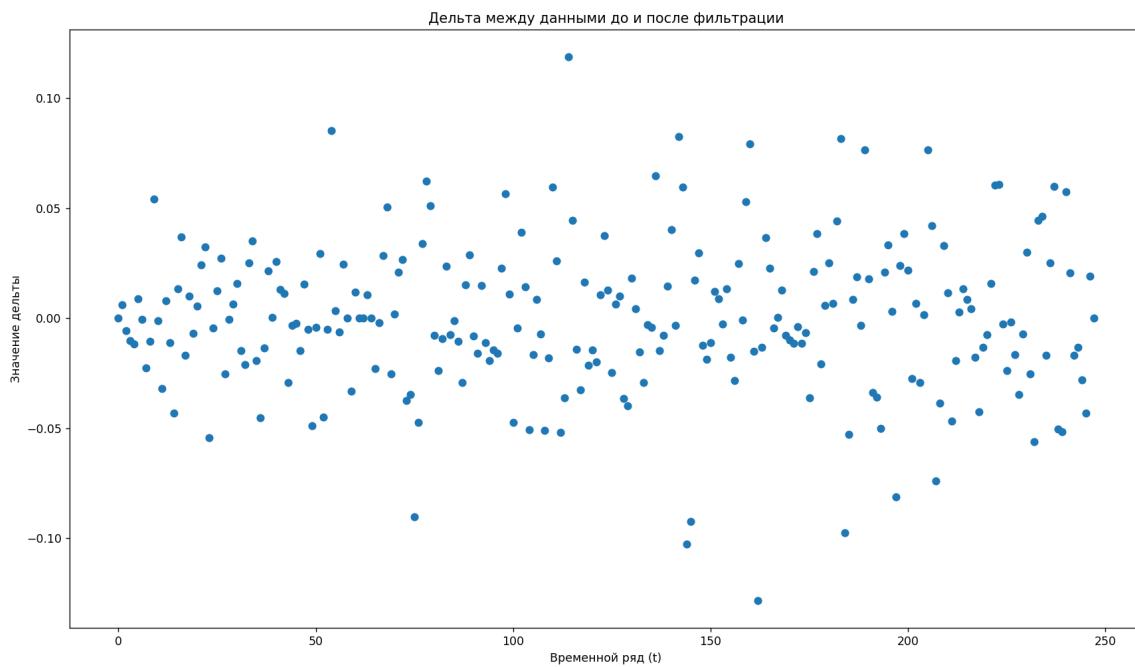
*Рисунок 11 – Фильтрация данных-1*



*Рисунок 12 – дельта между значениями до и после сглаживания для данных-1*



*Рисунок 13 – Фильтрация курса валют*



*Рисунок 14 – дельта между значениями до и после сглаживания для курса валют*

### **Проверка на стационарность**

Проверим наши временные ряды на стационарность после и до всех манипуляций.

#### **Данные-1:**

ADF-тест:

ADF статистика: -2.4292158223656077

р-значение: 0.13365965417839976

Критические значения:

1%: -3.5274258688046647

5%: -2.903810816326531

10%: -2.5893204081632653

KPSS-тест:

KPSS статистика: 0.058708085224112194

р-значение: 0.1

Лаги: 5

Критические значения:

10%: 0.347

5%: 0.463

2.5%: 0.574

1%: 0.739

ADF-тест не отвергает гипотезу о наличии единичного корня. **Временной ряд нестационарен.**

ADF-тест:

ADF статистика: -4.097849975127764

р-значение: 0.0009764803714909481

Критические значения:  
1%: -3.512738056978279  
5%: -2.8974898650628984  
10%: -2.585948732897085

KPSS-тест:  
KPSS статистика: 0.06994261208152026

р-значение: 0.1

Лаги: 5

Критические значения:

10%: 0.347

5%: 0.463

2.5%: 0.574

1%: 0.739

### **Изначальный временной ряд стационарен.**

**Курс валют:**

ADF-тест:

ADF статистика: -1.6474826590082277

р-значение: 0.45835362124440077

Критические значения:

1%: -3.460018927623594

5%: -2.8745897386146817

10%: -2.57372514086348

KPSS-тест:

KPSS статистика: 1.7510062224718013

р-значение: 0.01

Лаги: 10

Критические значения:

10%: 0.347

5%: 0.463

2.5%: 0.574

1%: 0.739

ADF-тест не отвергает гипотезу о наличии единичного корня. **Временной ряд нестационарен.**

ADF-тест:

ADF статистика: -2.1118769297668365

р-значение: 0.23976572508807864

Критические значения:

1%: -3.457664132155201

5%: -2.8735585105960224

10%: -2.5731749894132916

KPSS-тест:

KPSS статистика: 1.8266541678915285

p-значение: 0.01

Лаги: 10

Критические значения:

10%: 0.347

5%: 0.463

2.5%: 0.574

1%: 0.739

ADF-тест не отвергает гипотезу о наличии единичного корня.

### **Изначальный временной ряд нестационарен.**

Курс валют является нестационарным до и после обработки, что свойственно для курсов валют.

Данные-1 были стационарными до обработки и стали нестационарными после. Если смотреть на график, то заметно, что он идет на снижение.

### **Проверка на наличие тренда**

Так как наши данные нестационарные, то мы должны обнаружить тренд.

Проверим наличие тренда для данных-1:

'statistic': 3.254885148422351

'p.value': 0.0011343823046649926

**p.value<0.05. Гипотеза об отсутствии тренда отклоняется.**

Проверим наличие тренда для курса валют:

'statistic': 5.895123005292151

'p.value': 3.744023730731339e-09

**p.value<0.05. Гипотеза об отсутствии тренда отклоняется.**

## **4. Вывод**

В ходе выполнения лабораторной работы произвели удаление выбросов из временных рядов при помощи метода Isolation forest, убедились в отсутствии белых шумов и наличии автокорреляции, благодаря тесту Бокса-Пирса, произвели сглаживание данных фильтром Чебышева 2-го рода и методом взвешенного скользящего среднего по 3 точкам, убедились в нестационарности обоих рядов с помощью тестов Дикки-Фуллера и Квятковского-Филлипса-Шмидта-Шина, обнаружили наличие тренда, используя фазочастотный критерий Валлиса-Мура.

## 5. Листинг программы

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import interp1d
import seaborn as sns
from sklearn.ensemble import IsolationForest
import statsmodels.api as sm
from scipy.signal import cheby2, filtfilt
from statsmodels.tsa.stattools import adfuller, kpss
from scipy.stats import norm

# Загрузка данных из файла 1.xlsx
df = pd.read_excel('currencies.xlsx')

#df = pd.read_excel('data.xlsx')

# Оригинальная диаграмма рассеяния
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(x=df.index, y='curs', data=df)
plt.title('Диаграмма рассеяния до удаления выбросов')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда (curs)')

# Выявление и удаление выбросов

# удаление значений, которые находятся за пределами 3 стандартных отклонений от среднего
mean_val = df['curs'].mean()
std_dev = df['curs'].std()
df_cleaned = df[(df['curs'] >= mean_val - 2 * std_dev) & (df['curs'] <= mean_val + 2 * std_dev)]

# Сохранение восстановленных данных в файл result.xlsx
df_cleaned.to_excel('resultfirst.xlsx', index=False)

# Диаграмма рассеяния после удаления выбросов
plt.subplot(1, 2, 2)
sns.scatterplot(x=df_cleaned.index, y='curs', data=df_cleaned)
plt.title('Диаграмма рассеяния после удаления выбросов')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда (curs)')

# Отображение обоих графиков
plt.tight_layout()
plt.show()

# Оригинальная диаграмма рассеяния
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(x=df.index, y='curs', data=df)
plt.title('Диаграмма рассеяния до удаления выбросов')
plt.xlabel('Индекс значения')
plt.ylabel('Значение временного ряда')

# Выявление выбросов с использованием Isolation Forest
clf = IsolationForest(contamination=0.01, random_state=42)
df['outlier'] = clf.fit_predict(df[['curs']])

df['curs'] = np.where(df['outlier'] == -1, np.nan, df['curs'])
df['curs'] = df['curs'].interpolate(method='linear', limit_direction='both')

# Сохранение данных в файл resultsecond.xlsx
```

```

df.to_excel('resultsecond.xlsx', index=False)

# Диаграмма рассеяния после удаления выбросов
plt.subplot(1, 2, 2)
sns.scatterplot(x=df.index, y='curs', data=df)
plt.title('Диаграмма рассеяния после удаления выбросов (Isolation Forest)')
plt.xlabel('Индекс значения')
plt.ylabel('Значение временного ряда')

# Отображение обоих графиков
plt.tight_layout()
plt.show()

# Загрузка восстановленных данных из файла result.xlsx
df_cleaned = pd.read_excel('resultsecond.xlsx')

# Проверка на белый шум с использованием теста Бокса-Пирса
residuals = df_cleaned['curs'].diff().dropna() # Оставляем только остатки (разности между соседними
# значениями)
box_pierce_result = sm.stats.acorr_ljungbox(residuals, lags=20, return_df=True)

# Вывод результатов теста
print("Статистика Бокса-Пирса:")
print(box_pierce_result)

# Проверка наличия белого шума
alpha = 0.05
if (box_pierce_result['lb_pvalue'] > alpha).all():
    print("Остатки не показывают значимой автокорреляции. Вероятно, временной ряд представляет собой белый
шум.")
else:
    print("Остатки демонстрируют значимую автокорреляцию. Вероятно, временной ряд не является белым шумом.")

# Визуализация результатов
plt.figure(figsize=(10, 6))
plt.plot(box_pierce_result['lb_pvalue'], marker='o', linestyle='-', color='b')
plt.axhline(y=0.05, color='r', linestyle='--', label='Уровень значимости 0.05')
plt.title('Статистика Бокса-Пирса для тестирования на белый шум')
plt.xlabel('Лаг')
plt.ylabel('p-значение')
plt.legend()
plt.show()

# Загрузка данных из файла result.xlsx
df_cleaned = pd.read_excel('resultsecond.xlsx')

# Оригинальная диаграмма рассеяния
plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
sns.scatterplot(x=df_cleaned.index, y='curs', data=df_cleaned)
plt.title('Диаграмма рассеяния до фильтрации')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда')

# Фильтрация данных с использованием фильтра Чебышева второго рода
order = 2 # Порядок фильтра
Wn = 0.7 # Частота среза
rs = 5 # Затухание на заданной частоте в дБ

b, a = cheby2(N=order, Wn=Wn, rs=rs, btype='lowpass', analog=False, output='ba')
df_cleaned['filtered_curs'] = filtfilt(b, a, df_cleaned['curs'])

# Сохранение фильтрованных данных в файл resultCheb.xlsx
df_cleaned.to_excel('resultCheb.xlsx', index=False)

# Диаграмма рассеяния после фильтрации
plt.subplot(1, 2, 2)
sns.scatterplot(x=df_cleaned.index, y='filtered_curs', data=df_cleaned)
plt.title('Диаграмма рассеяния после фильтрации (Чебышев второго рода)')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда после фильтрации')

# Отображение обоих графиков
plt.tight_layout()
plt.show()

# Загрузка отфильтрованных данных из файла resultCheb.xlsx
df_filtered = pd.read_excel('resultCheb.xlsx')

# Оригинальная диаграмма рассеяния

```

```

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.plot(df_filtered['filtered_curs'], label='Отфильтрованные данные')
plt.title('График до сглаживания')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда')

# Сглаживание данных с использованием взвешенного скользящего среднего по 3 точкам
window_size = 3
weights = [0.25, 0.5, 0.25] # Веса для взвешенного среднего

df_filtered['smoothed_curs'] = df_filtered['filtered_curs'].rolling(window=window_size,
center=True).apply(lambda x: (x * weights).sum(), raw=True)

# Добавление крайних точек после сглаживания
df_filtered['smoothed_curs'].iloc[0] = df_filtered['filtered_curs'].iloc[0]
df_filtered['smoothed_curs'].iloc[-1] = df_filtered['filtered_curs'].iloc[-1]

# Смещение массива на 1 индекс
#df_filtered_e = df_filtered['smoothed_curs']
# Сохранение сглаженных данных в файл resultSmooth.xlsx
df_filtered.to_excel('resultSmooth.xlsx', index=False)

# График после сглаживания
plt.subplot(1, 2, 2)
plt.plot(df_filtered['filtered_curs'], label='Отфильтрованные данные')
plt.plot(df_filtered['smoothed_curs'], label='Сглаженные данные')
plt.title('График после сглаживания')
plt.xlabel('Временной ряд (t)')
plt.ylabel('Значение временного ряда')
plt.legend()

# Отображение обоих графиков
plt.tight_layout()
plt.show()

# Загрузка сглаженных данных из файла resultSmooth.xlsx
df_smoothed = pd.read_excel('resultSmooth.xlsx')

# Тест Дикки-Фуллера (ADF-тест)
result_adf = adfuller(df_smoothed['smoothed_curs'])
print('ADF-тест:')
print(f'ADF статистика: {result_adf[0]}')
print(f'p-значение: {result_adf[1]}')
print('Критические значения:')
for key, value in result_adf[4].items():
    print(f'{key}: {value}')

# Тест Квятковского Филлипса Шмидта Шина (KPSS-тест)
result_kpss = kpss(df_smoothed['smoothed_curs'], regression='c')
print('\nKPSS-тест:')
print(f'KPSS статистика: {result_kpss[0]}')
print(f'p-значение: {result_kpss[1]}')
print(f'Лаги: {result_kpss[2]}')
print('Критические значения:')
for key, value in result_kpss[3].items():
    print(f'{key}: {value}')

# Вывод результата в консоль
if result_adf[1] < 0.05 and result_kpss[1] > 0.05:
    print('\nВременной ряд стационарен.')
elif result_adf[1] >= 0.05:
    print('\nADF-тест не отвергает гипотезу о наличии единичного корня. Временной ряд нестационарен.')
elif result_kpss[1] <= 0.05:
    print('\nKPSS-тест отвергает гипотезу о стационарности. Временной ряд нестационарен.')
else:
    print('\nНеоднозначные результаты. Требуется дополнительный анализ.')

# Тест Дикки-Фуллера (ADF-тест)
result_adf = adfuller(df['curs'])
print('ADF-тест:')
print(f'ADF статистика: {result_adf[0]}')
print(f'p-значение: {result_adf[1]}')
print('Критические значения:')
for key, value in result_adf[4].items():
    print(f'{key}: {value}')

```

```

# Тест Квятковского Филиппса Шмидта Шина (KPSS-тест)
result_kpss = kpss(df['curs'], regression='c')
print('\nKPSS-тест:')
print(f'KPSS статистика: {result_kpss[0]}')
print(f'p-значение: {result_kpss[1]}')
print(f'Лаги: {result_kpss[2]}')
print('Критические значения:')
for key, value in result_kpss[3].items():
    print(f'{key}: {value}')

# Вывод результата в консоль
if result_adf[1] < 0.05 and result_kpss[1] > 0.05:
    print('\nИзначальный временной ряд стационарен.')
elif result_adf[1] >= 0.05:
    print('\nADF-тест не отвергает гипотезу о наличии единичного корня. Изначальный временной ряд нестационарен.')
elif result_kpss[1] <= 0.05:
    print('\nKPSS-тест отвергает гипотезу о стационарности. Изначальный временной ряд нестационарен.')
else:
    print('\nНеоднозначные результаты. Требуется дополнительный анализ.')

# Загрузка сглаженных данных из файла resultSmooth.xlsx
df_smoothed = pd.read_excel('resultSmooth.xlsx')

data = df_smoothed['smoothed_curs'].values

def wm_test(x):
    n = len(x)
    h = n // 2 # Number of phases

    # Calculate the test statistic
    z_numerator = np.abs(h - (2 * n - 7) / 3)
    z_denominator = np.sqrt((16 * n - 29) / 90)

    # Include continuity correction for n <= 30
    if n <= 30:
        z_denominator += -0.5

    z = z_numerator / z_denominator

    # Calculate p-value
    p_value = 2 * norm.cdf(-np.abs(z))

    # Determine the alternative hypothesis
    alternative = "two-sided" if z_numerator > (2 * n - 7) / 3 else "less"

    # Prepare result
    result = {
        "method": "Wallis and Moore Phase-Frequency Test",
        "data.name": "x",
        "statistic": z,
        "alternative": alternative,
        "p.value": p_value
    }

    return result

result = wm_test(data)
print(result)

# Извлечение выборок
observed_values = df_smoothed['curs']
smoothed_values = df_smoothed['smoothed_curs']

# Расчет RMSE
rmse = np.sqrt(((observed_values - smoothed_values) ** 2).mean())

print(f'RMSE: {rmse}')

```