

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

**СТАТИСТИЧЕСКИЙ И СПЕКТРАЛЬНЫЙ АНАЛИЗ.
ПРИМЕНЕНИЕ СТАТИСТИЧЕСКИХ КРИТЕРИЕВ И ТЕСТОВ.**

**Методические указания к выполнению
практического задания № 2**

Екатеринбург
2020

Содержание

Введение.....	3
1. Задание на лабораторную работу	3
2. Требования к оформлению отчета.....	12

Введение

На прошлой лабораторной работе мы изучили базовые средства работы с временными рядами, а также методы реализации собственных функций. Также там требовалось определить тип процесса, породившего данный ВР, что было достаточно трудным занятием, если не опираться на применение статистических критериев, которые будут рассматриваться в данной работе. Также наряду с автокорреляционной функцией изучаются спектральные свойства рядов.

1. Задание на лабораторную работу

Результатом выполнения лабораторной работы является оформленный отчет в виде *Jupyter*-тетради, в котором должны быть представлены и отражены все нижеперечисленные пункты:

- 1) Сначала импортируйте в свой код нужные библиотеки, функции и т.д.

```
import numpy as np  
import numpy.random as rand  
import matplotlib.pyplot as plt  
from scipy import signal  
import scipy.stats as stats  
from statsmodels.tsa import api as tsa  
from pandas.tools.plotting import autocorrelation_plot  
from statsmodels.graphics.tsaplots import plot_acf  
%matplotlib inline
```

- 2) Постройте периодический сигнал, имеющий **2 разных периода**:

```
t = np.linspace(0, 1, 4096)
x1 = np.sin(2*np.pi*10*t) + np.sin(2*np.pi*120*t)
plt.figure(figsize = (10, 5))
plt.plot(t, x1)
```

- 3) Оцените его периодограмму и оценку спектральной плотности мощности ряда с помощью метода Велша (Welch):

```
pd1, pdden1 = signal.periodogram(x1)
pdw1, pddenw1 = signal.welch(x1, nperseg = 1024)
plt.figure(figsize = (10, 5))
plt.semilogy(pd1, pdden1)
plt.semilogy(pdw1, pddenw1)
```

- 4) Создайте периодический сигнал с **изломом частоты**:

```
t = np.linspace(0, 1, 4096)
x2 = np.zeros(4096)
for i in range(0, len(t)//2):
    x2[i] = np.sin(2*np.pi*10*t[i])
for i in range(len(t)//2, len(t)):
    x2[i] = np.sin(2*np.pi*120*t[i])
plt.figure(figsize = (10, 5))
plt.plot(t, x2)
```

- 5) Оцените его спектр:

```
pd2, pdden2 = signal.periodogram(x2)
pdw2, pddenw2 = signal.welch(x2, nperseg = 1024)
plt.figure(figsize = (10, 5))
plt.semilogy(pd2, pdden2)
plt.semilogy(pdw2, pddenw2)
```

- 6) Постройте спектры этих двух сигналов на одном изображении:

```
plt.figure(figsize = (10, 5))
plt.semilogy(pd1, pdden1)
plt.semilogy(pd2, pdden2)
plt.figure(figsize = (10, 5))
plt.semilogy(pdw1, pddenw1)
plt.semilogy(pdw2, pddenw2)
```

- 7) Обратите внимание, насколько похожи между собой эти спектры, хотя исходные данные существенно отличаются. Это связано с тем, что данные спектральные оценки усредняют периоды по времени, теряя зависимость частоты от времени, если она существовала. Попробуем решить эту проблему на основе расчета частотно-временных характеристик ряда.

- 8) Построим **спектрограмму** заданного ряда, чтобы вычислить его частотно-временные характеристики. Для этого сначала нам потребуется обязательно рассчитать его частоту дискретизации:

```
fs = 1/(t[1]-t[0]) # fs = 1/dt = N/T
```

- 9) Строим спектрограмму для суммы двух периодик:

```
f, tx, Sxx = signal.spectrogram(x1, fs) # возвращаем частоту от времени
plt.figure(figsize = (10, 5))
plt.pcolormesh(tx, f, Sxx) # цвет – интенсивность спектрограммы
plt.ylabel('Frequency [Hz]')
plt.ylim(0, 150) # строим до 150 Гц, иначе будет до fs/2
plt.xlabel('Time [sec]')
plt.show()
```

- 10) Получится картина, по которой в принципе можно различить две постоянные периодики в районе 10 и 120 Гц, но точность весьма низкая. Все дело в том, что спектрограмма по факту представляет из

себя оконное преобразование Фурье, которое последовательно применяется к отдельным пересекающимся временным сегментам в рамках всего временного интервала. Следует изменить параметры сегментов спектрограммы для более ярко-выраженного результата:

```
f, tx, Sxx = signal.spectrogram(x1, fs, nperseg = 512, noverlap = 496, nfft=4096)
# длина каждого сегмента = 512, число пересекающихся точек между
сегментами = 496, длина FFT = 4096

plt.figure(figsize = (10, 5))
plt.pcolormesh(tx, f, Sxx)
plt.ylabel('Frequency [Hz]')
plt.ylim(0, 150) # строим до 150 Гц, иначе будет до fs/2
plt.xlabel('Time [sec]')
plt.show()
```

- 11) Теперь аналогично самостоятельно постройте спектрограмму второго ряда **x2**, представляющего собой отсчеты сигнала с изломом частоты. Подберите параметры его спектрограммы самостоятельно.
- 12) Сравните полученные спектрограммы двух рядов между собой и сделайте выводы (где есть зависимость от времени, а где ее нет).
- 13) Теперь создадим временной ряд с ЛЧМ (линейной частотной модуляцией) в диапазоне от 50 до 150 Гц:

```
tx = np.linspace(0, 1, 8192) # временной отрезок от 0 до 1 сек
w = signal.chirp(tx, f0=50, f1=150, t1=1, method='linear')
# от 50 до 150 Гц за 1 секунду, ЛЧМ

plt.plot(tx, w)
plt.title("Linear Chirp, from 50 to 150 Hz")
plt.xlabel('t (sec)')
plt.show()
```

- 14) Построим спектрограмму заданного ряда, чтобы вычислить его частотно-временные характеристики. Для этого сначала нам потребуется рассчитать его частоту дискретизации:

```
fs = 1/(tx[1]-tx[0]) # fs = 1/dt = N/T
```

- 15) Строим спектрограмму:

```
f, t, Sxx = signal.spectrogram(w, fs, nperseg = 512, noverlap = 496, nfft=4096)  
# длина каждого сегмента = 512, число пересекающихся точек между  
сегментами = 496, длина FFT = 4096  
plt.figure(figsize = (10, 5))  
plt.pcolormesh(t, f, Sxx, cmap='gray_r') # в оттенках серого цвета  
plt.ylabel('Frequency [Hz]')  
plt.ylim(0, 200)  
plt.xlabel('Time [sec]')  
plt.show()
```

- 16) Полученная черно-белая картина спектрограммы хорошо отражает линейную структуру частотной модуляции. Тем не менее, для оценки диапазона частотной модуляции хотелось бы иметь более точный численный инструмент – таковым является Преобразование Гильберта. Преобразование Гильберта позволяет однозначно определить понятие аналитического сигнала, через который можно определить и функцию амплитудной модуляции (АМ, мгновенная амплитуда), и функцию фазы, и функцию **мгновенной частоты** (*instantaneous frequency*), как производную от мгновенной фазы, то есть как раз искомую зависимость частоты от времени для ЧМ.

Аналитический сигнал: $w(t) = u(t) + jv(t) = a(t)e^{j\varphi(t)}$

Преобразование Гильберта: $v(t) = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{u(\tau)}{t - \tau} d\tau = H(u)$

Мгновенная амплитуда: $a(t) = |w(t)| = \sqrt{u^2(t) + v^2(t)}$.

Фаза: $\varphi(t) = \arctg\left(\frac{v(t)}{u(t)}\right) = \arccos\left(\frac{u(t)}{a(t)}\right) = \arcsin\left(\frac{v(t)}{a(t)}\right)$

Мгновенная частота: $\omega(t) = \dot{\varphi}(t) = \frac{u(t)\dot{v}(t) - \dot{u}(t)v(t)}{a^2(t)}$.

17) Запишем все это на языке *Python*:

```
analytic_signal = signal.hilbert(w) # аналитический сигнал
instantaneous_phase = np.unwrap(np.angle(analytic_signal))
# мгновенная фаза в развернутом непрерывном виде
instantaneous_frequency = (np.diff(instantaneous_phase) / (2.0*np.pi) * fs)
# мгновенная частота как производная от фазы, приведенная в Гц
plt.figure(figsize = (10, 5))
# из-за численного расчета производной массив мгновенной частоты
будет меньше массива времени на одну точку:
plt.plot(tx[1:], instantaneous_frequency)
plt.show()
```

18) Полученный график имеет четко выраженную линейную форму частоты от 50 до 150 Гц, за исключением краевых эффектов, которые все портят. Эти краевые искажения связаны с численным расчетом производной от мгновенной частоты и на практике для избавления от них полученный ряд сглаживают скользящим средним или линейной регрессионной кривой. Ну или просто отрезают значения по краям временного интервала.

- 19) Теперь, когда нам известно, как оценить частотно-временные характеристики ряда, постройте зависимость частоты от времени для следующих модельных временных рядов через **спектрограмму** и **преобразование Гильберта**:

```
tx = np.linspace(0, 1, 8192) # ЛЧМ в большом диапазоне  
w = signal.chirp(tx, f0=200, f1=3000, t1=1, method='linear')
```

- 20) Ряд с квадратичной частотной модуляцией:

```
tx = np.linspace(0, 1, 8192)  
w = signal.chirp(tx, f0=2000, f1=200, t1=1, method='quadratic')
```

- 21) Ряд с инверсной квадратичной частотной модуляцией:

```
tx = np.linspace(0, 1, 8192)  
w = signal.chirp(tx, f0=3200, f1=400, t1=1, method='quadratic', vertex_zero=False)
```

- 22) Ряд с логарифмической частотной модуляцией:

```
tx = np.linspace(0, 1, 8192)  
w = signal.chirp(tx, f0=2450, f1=300, t1=1, method='logarithmic')
```

- 23) Ряд с гиперболической частотной модуляцией:

```
tx = np.linspace(0, 1, 8192)  
w = signal.chirp(tx, f0=1500, f1=250, t1=1, method='hyperbolic')
```

- 24) Ряд с полиномиальной частотной модуляцией:

```
tx = np.linspace(0, 10, 8192)  
p = np.poly1d([2.5, -36.0, 125.0, 150.0])  
w = signal.sweep_poly(tx, p)
```

25) Ряд с частотной модуляцией другим гармоническим сигналом:

```
tx = np.linspace(0, 10, 2*8192)
mod = 500*np.cos(2*np.pi*0.25*tx)
w = 2 * np.sqrt(2) * np.sin(2*np.pi*300*tx + mod)
```

26) Временной ряд из **4 периодик без шума** (код составьте самостоятельно):

$$u(t) = \sin[2\pi t(f_1)] + \sin[2\pi t(f_2)] + \sin[2\pi t(f_3)] + \sin[2\pi t(f_4)]$$

27) В случае с последним временным рядом из нескольких периодик для Преобразования Гильберта должен получиться неожиданный результат – нечто совершенно не похожее ни коим образом на 4 разных, но постоянных периода. Связано это с тем, что сумма гармоник трактуется через Преобразование Гильберта как единый аналитический сигнал с некоторой сложной амплитудной модуляцией (сумма синусов может быть записана как произведение синуса на косинус). По этой причине, прежде чем применять метод аналитического сигнала и расчета мгновенной частоты, исходный временной ряд всегда сначала раскладывают в **аддитивную сумму компонент** в разных частотных областях.

28) Наконец, вспомним, как работать в *Python* с проверкой статистических гипотез.

- 29) Создайте временной ряд, как частную выборку из нормального распределения:

```
x = rand.randn(10000)
t = np.linspace(3, 5, num = 10000)
plt.figure(figsize = (10, 5))
plt.plot(t, x)
plt.show()
```

- 30) Произведите оценку ВР на **стационарность**. Сначала используйте известный **KPSS-тест**. Для этого есть функция `tsa.kpss(x)`, которая возвращает статистику теста **kpss_stat**, p-значение теста **p_value**, и другие полезные результаты (критические значения на разных перцентилях и т.д.).
- 31) Если KPSS-test в статистике близок к 0, то временной ряд является **стационарным** по КПСС-тесту, то есть **нулевая гипотеза** о тренд-стационарности ряда **принята**.
- 32) Если KPSS-test в статистике вернул значение существенно больше нуля, то временной ряд **не является стационарным** по КПСС-тесту, то есть **нулевая гипотеза** о тренд-стационарности ряда **отвергнута** и принята **альтернативная**.
- 33) Более точный показатель, на который следует обратить внимание – это **p-value**. Он, чаще всего, трактуется следующим образом: если значение **p-value** меньше **0.05**, то нулевая гипотеза **отклоняется**. Если значение больше **0.05**, то нулевая гипотеза **принимается**. При граничном значении **p-value** близком к **0.05** ответ является **неоднозначным** и гипотезу следует принимать или отвергать с большой осторожностью. На практике значение **p-value** близкое к **0.05** обычно обозначает недостаточную длину исходной выборки анализируемых данных.

- 34) Скорее всего, случайная выборка нормального шума будет стационарной, так как не меняет со временем свои статистические характеристики. Внесите явную нестационарность в этот ряд:

```
xv=x+(10*t**2-100*t+300)
```

```
plt.figure(figsize = (10, 5))
```

```
plt.plot(t, xv)
```

```
plt.show()
```

- 35) Примените к нему **KPSS-тест**. Сделайте выводы на основе анализа значений полученных **p-value**.

- 36) Теперь проверьте с помощью критерия Фишера две половинки (по времени) исходного временного ряда на соответствие дисперсий, и затем проделайте то же самое для модифицированного временного ряда. Критерий Фишера можно реализовать через функцию **stats.f_oneway(x, y)**

- 37) Проверьте с помощью критерия Стьюдента две половинки исходного временного ряда на соответствие мат. ожиданий (при предположении о равных дисперсиях), и затем проделайте то же самое для модифицированного временного ряда. Используйте функцию **scipy.stats.ttest_ind(x, y)**

- 38) Пояснить все полученные результаты.

2. Требования к оформлению отчета

Отчет в Jupyter-тетради должен обязательно содержать: номер лабораторной работы, ФИО студента, номер варианта (либо студенческий номер), номер группы, результаты выполнения работы с комментариями студента (комментарии пишутся после #) и изображениями.