МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ЧАСТОТНО-ВРЕМЕННЫЕ ХАРАКТЕРИСТИКИ ВРЕМЕННЫХ РЯДОВ

Методические указания к выполнению лабораторной работы № 7

Екатеринбург

2019

Уральский федеральный университет имени первого Президента России Б.Н.Ельцина

Содержание

Введение

Целью данной лабораторной работы является изучение студентами методов адаптивного анализа временных рядов, с их дальнейшей оценкой частотно-временных характеристик через преобразование Гильберта или спектрограммы. В конце студентам самостоятельно предлагается воссоздать метод эмпирической модовой декомпозиции.

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
%matplotlib inline
```

2) 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()
```

3) 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

4) Строим спектрограмму по умолчанию:

   f, t, Sxx = signal.spectrogram(w, fs) # возвращаем частоту от времени

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

   plt.pcolormesh(t, f, Sxx) # цвет – интенсивность спектрограммы

   plt.ylabel('Frequency [Hz]')

   plt.ylim(0, 200) # строим до 200 Гц, иначе будет до fs/2

   plt.xlabel('Time [sec]')

   plt.show()

5) Получится картина, по которой в принципе можно различить диапазон от 50 до 150 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 спектрограммы для более ярко-выраженного результат:

   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()

6) Полученная черно-белая картина спектрограммы гораздо лучше отражает линейную структуру частотной модуляции, как и ее диапазон. 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: $v(t) = \dfrac{1}{\pi}\int\limits_{-\infty}^{+\infty} \dfrac{u(\tau)}{t-\tau} d\tau = H(u)$

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

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

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

В виде кода на *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()
```

7) 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 регрессионной кривой.
8) Поэтому, на основе Ваших навыков из лабораторной работы №3 **постройте линейный тренд** для мгновенной частоты **instantaneous_frequency** по методу линейной регрессии и по методу скользящего среднего сглаживания. Оцените диапазон частоты (значение частоты в начальной и конечной точках) в обоих случаях.

9) 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, не забывая сглаживать мгновенную частоту регрессионной кривой:

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

10) Ряд с квадратичной частотной модуляцией:

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

11) 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)
```

12) Ряд с логарифмической частотной модуляцией:

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

13) Ряд с гиперболической частотной модуляцией:

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

14) 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)
```

15) 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)
```

16) Ряд с частотным изломом:

```
tx = np.linspace(0, 1, 4096)
xf = np.zeros(4096)
for i in range(0, len(tx)//2):
   xf[i] = np.sin(2*np.pi*100*tx[i])
for i in range(len(tx)//2, len(tx)):
   xf[i] = np.sin(2*np.pi*1200*tx[i])
```

17) Временной ряд из **4 гармоник без шума**:

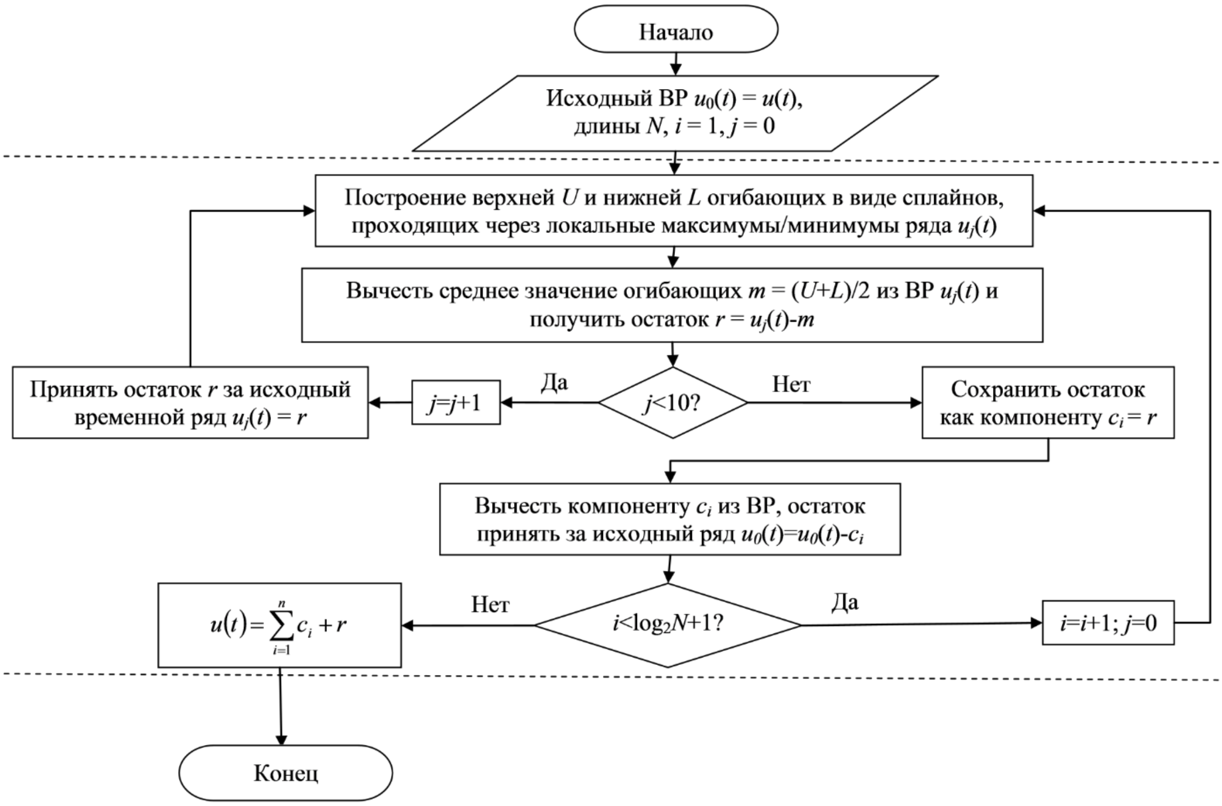
$$u(t) = \sin\left[2\pi t\left(f_1\right)\right] + \sin\left[2\pi t\left(f_2\right)\right] + \sin\left[2\pi t\left(f_3\right)\right] + \sin\left[2\pi t\left(f_4\right)\right]$$

18) 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).

19) 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. На уже известны несколько подобных методов – регрессия, сингулярный спектральный анализ, вейвлет-декомпозиция и т.д. В этот раз мы используем адаптивный метод Эмпирической Модовой Декомпозиции.

20) И так, Ваше новое задание – **реализовать** базовый метод Эмпирической Модовой Декомпозиции (**ЭМД**) в *Python*, блок-схема представлена ниже.

21) Кратко, суть метода – провести огибающие кривые в виде кубических сплайнов через максимумы (верхняя кривая) и минимумы (нижняя кривая) ряда, найти их среднее и вычесть из ряда (повторить 10 раз). Полученный остаток принять за компоненту, вычесть из ряда и начать все заново, пока не получатся все желаемые компоненты.



22) Итак, с учетом вышесказанного, Вам могут пригодиться следующие функции *Python* (могут быть использованы и любые другие функции по Вашему усмотрению, список просто в помощь):

scipy.signal.argrelmax

scipy.signal.argrelmin

scipy.signal.argrelextrema

scipy.signal.find_peaks

scipy.interpolate

scipy.interpolate.CubicSpline

и реализации сплайнов из других библиотек

23) Также разрешено ограничивать число мод вручную (когда их число заранее известно по модельному ряду) вместо эмпирической формулы ($\log_2 N$+1) – для простоты реализации и декомпозиции.

24) В конечной реализации будут присутствовать краевые эффекты у полученных мод – это не является ошибкой, а есть системная погрешность самого метода ЭМД без модификаций.

25) Полученную реализацию ЭМД сначала апробируйте на самом простом модельном временном ряде – сумме двух периодик без шума, например:

t = np.linspace(0, 1, 1024)

f1 = 10, f2 = 40

F=np.sin(2*np.pi*f1*t)+np.sin(2*np.pi*f2*t)

26) Затем готовый метод ЭМД примените к ряду, состоящему из 4 гармоник без шума (см. пункт 17), и для каждой полученной компоненты постройте зависимость мгновенной частоты от времени.

27) Затем добавьте в модельный ряд шум и посмотрите, как изменятся результаты.

28) Аналогично, попробуйте провести декомпозицию следующих временных рядов: с изломом частоты (см. пункт 16), случайный ряд белого шума с нормальным распределением, экспоненциальный зашумленный тренд:

t = np.linspace(0, 4, 4096)

Fexp = np.exp(-0.4*np.pi*t) + 0.2*rand.randn(len(t))

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.