МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет имени
первого Президента России Б. Н. Ельцина»

ПОСТРОЕНИЕ ТИПОВЫХ МОДЕЛЕЙ АРПСС (ARIMA) И ПРОГНОЗИРОВАНИЕ ВР МЕТОДАМИ ARIMA И SSA «Гуссеница»

Методические указания к выполнению
практического задания № 5

Екатеринбург
2023

Содержание

Введение 3

1. Задание на лабораторную работу 3

2. Требования к оформлению отчета 20

Введение

Напомним, что в общем виде модель авторегрессии – скользящего среднего порядка (p, q) АРПСС (ARIMA) выглядит как:

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + ... + \phi_p \tilde{z}_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - ... - \theta_q a_{t-q}.$$

Эта модель временных рядов имеет целый ряд преимуществ в сравнении с другими моделями, одно из которых – это возможность их оперативного прогноза по построенной модели. В связи с этим ни одна методика анализа и изучения временных рядов не может обойтись без рассмотрения подобного класса задач.

Метод сингулярного спектрального анализа SSA относится к адаптивным методам, и, потому, весьма эффективен для анализа и прогноза множества различных временных рядов, в том числе и нестационарных. Одним из важных замечаний, которое нужно сделать на начальном этапе данной лабораторной работы, состоит в том, что алгоритм SSA требует очень большой объем ОЗУ для больших значений окна L.

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
import h5py
from statsmodels.tsa import api as tsa
```

```
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.arima_model import ARIMA
%matplotlib inline
```

2. Для начала попробуем создать собственные АРПСС ряды первого и второго порядков и изучить их автокорреляционные функции.

3. Создадим два АР(1) процесса первого порядка:

$$z_t = 0.8z_{t-1} + a_t \quad \text{и} \quad z_t = -0.8z_{t-1} + a_t$$

где a_t – случайная нормально распределенная величина малой амплитуды (порядка 0.2), $z_0 = 1$. Постройте полученные ВР.

```
z1 = np.zeros(100)
z2 = np.zeros(100)
z1[0] = 1
z2[0] = 1
for i in range(1,100):
    z1[i] = 0.8 * z1[i - 1] + 0.2 * np.random.randn()
    z2[i] = -0.8 * z2[i - 1] + 0.2 * np.random.randn()
```

4. Постройте для этих рядов функции автокорреляции с помощью функции **plot_acf** лаг можно взять равный 50:

5. Сравните эти графики между собой: найдите их сходства и различия, а также характерные особенности, которые позволяют отнести их к модели АР первого порядка.

6. Оцените весовой параметр этих процессов (как если бы Вы не знали о них) с помощью формулы $\phi = \rho_1$, на основе функции автокорреляции. Также удостоверьтесь, что для модели АР(1) коэффициенты автокорреляции изменяются по степенному закону $\rho(l) = \phi^l$.

7. Аналогичным образом постройте два СС(1) процесса среднего-скользящего первого порядка:

$$z_t = a_t - 0.8a_{t-1} \qquad \text{и} \qquad z_t = a_t - (-0.8)a_{t-1}$$

где a_t – случайная нормально распределенная величина. Необходимо не только рассчитать ВР, но и построить.

```
z3 = np.zeros(100)
z4 = np.zeros(100)
ar = 0.2 * np.random.randn(100)
for i in range(1, 100):
    z3[i] = ar[i] - 0.8 * ar[i - 1]
    z4[i] = ar[i] + 0.8 * ar[i - 1]
```

8. Постройте для этих рядов функции автокорреляции, достаточно взять 25 лагов (четверть от длины ряда).

9. Сравните эти графики между собой: найдите их сходства и различия, а также характерные особенности, которые позволяют отнести их к модели СС первого порядка.

10. Оцените весовой параметр этих процессов (как если бы Вы не знали о них) с помощью формулы ниже, на основе функции автокорреляции.

$$\theta_1^2 + \theta_1 / \rho_1 + 1 = 0,\ |\theta_1| < 1$$

11. Также удостоверьтесь, что для модели СС(1) коэффициенты автокорреляции соответствуют формуле

$$\rho_k = \begin{cases} \dfrac{-\theta_1}{1+\theta_1^2}, & k = 1 \\ 0, & k \geq 2 \end{cases}$$

12. Наконец, создайте временной ряд процесса АРСС(1, 1):

$$z_t = 0.8z_{t-1} + a_t - 0.3a_{t-1} \qquad \text{и} \qquad z_t = -0.8z_{t-1} + a_t - 0.3a_{t-1}$$

где a_t – случайная нормально распределенная величина, $z_0 = 1$.

Напишите код *Python* самостоятельно на основе комбинации предыдущих примеров.

13. Постройте графики этих рядов и графики их автокорреляционных функций.

14. Есть и другой, более высокоуровневый способ генерации рядов АРПСС. Используем следующую функцию для создания АРСС (2, 2):

```
from statsmodels.tsa.arima_process import arma_generate_sample
ar = np.array([0.75, -0.25])    # дописать, что рассчитывается в данной строке
ma = np.array([0.65, 0.35])     # дописать, что рассчитывается в данной строке
y = arma_generate_sample(np.r_[1, -ar], np.r_[1, ma], 100) # дописать, что рассчитывается в данной строке
```

15. Теперь проведем анализ неизвестного ряда на типовом примере, а затем каждый из студентов проводит анализ собственного ВР по вариантам (номер варианта = последние две цифры студенческого билета).

16. Значения исходного ряда (всего их 24) приведены ниже:

```
TEST = [0.00, 9.99, 12.89, 10.70, 5.12, -1.21, -6.50, -7.96, -4.30, 0.42, 3.41, 4.50, 3.57, 2.24, 1.78, 0.89, -1.20, -3.43, -2.35, -0.85, -0.21, -0.08, 0.95, 0.45]
```

17. Постройте график ВР и его автокорреляционную функцию.

18. По ним можно судить, что ВР, в достаточной степени, **стационарен**, а, так как, эта функция является **знакопеременной**, то один из членов АР модели имеет отрицательный вес.

19. Создадим три пробные модели АРПСС для проверки ряда на АР(1) = АРПСС(1, 0, 0), АР(2), АР(3), без тренда (trend = ‘nc’):

```
arima1 = ARIMA(TEST, order = (1, 0, 0)) # пояснить
model_fit1 = arima1.fit(disp = False, trend='nc') # пояснить
```

```
print(model_fit1.summary())      # пояснить
arima2 = ARIMA(TEST, order = (2, 0, 0)) # пояснить
model_fit2 = arima2.fit(disp = False, trend='nc') # пояснить
print(model_fit2.summary())
arima3 = ARIMA(TEST, order = (3, 0, 0)) # пояснить
model_fit3 = arima3.fit(disp = False, trend='nc') # пояснить
print(model_fit3.summary())
```

20. Будут выведены три таблицы со всевозможной информацией, например, как ниже:

ARMA Model Results

Dep. Variable:	y	No. Observations:	24
Model:	ARMA(2, 0)	Log Likelihood	-41.543
Method:	css-mle	S.D. of innovations	1.201
Date:	Sun, 10 Mar 2019	AIC	89.086
Time:	23:50:38	BIC	92.620
Sample:	0	HQIC	90.024

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1.y	1.5108	0.056	27.117	0.000	1.402	1.620
ar.L2.y	-0.9641	0.035	-27.509	0.000	-1.033	-0.895

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	0.7836	-0.6506j	1.0185	-0.1103
AR.2	0.7836	+0.6506j	1.0185	0.1103

21. В этой таблице значения коэффициентов модели авторегрессии AP(2) написаны в столбце **coef**. СКВО их расчета – в следующем столбце.

22. **Как по этим таблицам выбрать наилучшую модель?** Во-первых, стоит обратить внимание на значение **AIC** – информационный критерий Акаике, который показывает максимальное правдоподобие модели при штрафовании за избыточные параметры системы. Считается, что наилучшей будет модель с **наименьшим** значением критерия AIC.

23. Аналогично есть **BIC** – Байесовский информационный критерий, модификация AIC. Данный критерий налагает больший штраф на увеличение количества параметров по сравнению с AIC.

24. Аналогично есть **HQIC** –информационный критерий Ханнана-Куинна (Hannan-Quinn), который асимптотически более точный метод чем BIC для дискретных параметров.

25. В любом случае, лучшей моделью будет та, что имеет наименьшее значение информационного критерия среди множества других. Рекомендуется, в первую очередь, выбирать по критерию **BIC**, так как он сильнее штрафует за переобучение модели и увеличение числа параметров по сравнению с другими. В нашем случае для тестового ВР, для любых информационных критериев, это модель АР(2).

26. Другим методом выбора модели может служить построение моделей АРПСС выбранного порядка и с найденными коэффициентам на графиках совмещенно.

```
plt.plot(model_fit.fittedvalues)
```

Например, для приведенного примера модель АР(1) совсем слабо подходит к ВР, АР(2) и АР(3) близки, АР(3) почти не отличается от АР(2), но избыточен по числе параметров (3>2), а значит АР(2) является наиболее оптимальной моделью ВР.

27. Теперь попробуйте найти весовые коэффициенты для АР моделей только **1 и 2 порядка** самостоятельно. Для этого Вам потребуется построить автокорреляционную функцию этого ряда.

28. Для нахождения весового коэффициента АР(1) используйте следующую формулу:

$$\phi = \rho_1.$$

где $\rho_1 - r(1)$ оценка автокорреляционной функции.

29. Для АР(2) используйте следующие формулы:

$$\phi_1 = \frac{\rho_1(1-\rho_2)}{1-\rho_1^2}, \quad \phi_2 = \frac{\rho_2 - \rho_1^2}{1-\rho_1^2}.$$

30. Убедитесь, что полученные веса будут близки к тем, что были получены с помощью функций *Python*.

31. Теперь в зависимости от своего варианта, который определяется по последним двум цифрам студ. билета, выберите из выданных преподавателей ***mat*-файлов** тот, который имеет номер Вашего варианта и загрузите из него временной ряд ***Z***.

32. Постройте график ВР и его автокорреляционную функцию.

33. Оцените порядок АРСС модели с помощью класса ARIMA. Для упрощения задачи выбора модели используйте только чистые АР или СС модели, то есть класс `ARIMA` с `order = (p, 0, 0)` или `order = (0, 0, q)`.

34. Выберите модель с наиболее подходящей структурой и вычислите для нее коэффициенты. Поясните в отчете выбор модели.

35. В дальнейшем попробуйте подобрать такую модель АРПСС (ARIMA) со всевозможными параметрами order = (p, d, q), которая будет наилучшей для данного ВР среди всех других по одному из информационных критериев.

36. Теперь обратимся к **прогнозированию** на основе АРПСС моделей. Загрузите из mat-файла **Fort.mat** ряд, содержащий отсчеты некоторого реального ВР, всего 174 отсчета в вектор-строке.

37. Отобразите полученный временной ряд.

38. Мы будем производить **ретроспективный прогноз**, аналогично предыдущей лабораторной работе. Для этого отрежем от данного ряда последние 24 точки (которые мы и будем прогнозировать).

39. Прежде, чем строить модель АРПСС, **обратите внимание**: модели АРПСС строятся для рядов с около-нулевым средним, что неверно для заданного временного ряда. Поэтому – **сначала постройте линейный тренд прогнозируемого ряда** (см. линейную регрессию первого порядка из предыдущей лаб. работы)**, а затем вычтите его из исходного ряда**, приведя

его к нулевому среднему значению (к так называемой **тренд-стационарной** форме).

40. Подберите для данного приведенного к нулю ВР, у которого к тому же отрезали последние 24 точки, модель АРПСС (p, d, q) некоторого порядка (все параметры целиком и полностью определяются самим студентом) по таблицам и информационным критериям. Например, была найдена некоторая наилучшая модель:

```
arimaz = ARIMA(Z_minus_trend, order = (p, d, q)) # пояснить
model_fit = arimaz.fit(disp = False) # пояснить
print(model_fit.summary()) # пояснить
```

41. Тогда график прогноза по данной модели вместе с доверительными интервалами строится очень легко:

```
model_fit.plot_predict(0, len(Fort))
```

42. Но хотелось бы все же увидеть – как же этот прогноз по АРПСС модели соотносится с исходными известными 24 прогнозными точками (ведь прогноз все-таки ретроспективный). Для этого нужно из исходного ряда ***Fort*** тоже вычесть линейный тренд и соотнести их на одном изображении:

```
plt.figure(figsize = (10, 5))
model_fit.plot_predict(0, len(Fort)) # пояснить
plt.plot(t0, Fort-(trend_as_func_of_t0), 'r') # пояснить
plt.show()
```

43. Сами прогнозные значения по модели АРПСС можно получить с помощью функции **predict**:

```
model_fit.predict(len(Z), len(Fort))
```

44. Используйте эти значения для оценки точности прогноза на основе оценок из предыдущей лабораторной работы (все, кроме коэффициентов несоответствия).

45. Также попробуйте построить АРПСС модель для прогнозирования данного ряда, **но без исходного вычитания из него линейного тренда**.

Отметьте получившиеся отличия в работе функций *Python* и точности конечных результатов.

46. Аналогично, для данной модели постройте графики прогноза с доверительными интервалами относительно оригинального ряда ***Fort***, а также оцените точность прогноза на основе оценок из пункта 11 выше.

47. Не забудьте в отчет-тетрадь добавить необходимые рисунки и таблицы результатов.

48. Метод сингулярного спектрального анализа SSA реализуется в 2 этапа – **разложение** и **группировка**. Соответственно, нам потребуется написать несколько функций – для сингулярного разложения ряда и затем для его обратной группировки.

49. Начнем с этапа разложения. Это будет функция

def SSA_modes(F, L):

которая принимает два параметра: сам временной ряд F и длину окна разложения L.

50. Внутри функции нам понадобится определить размерность траекторной $\boldsymbol{X}$ матрицы L x K.

```
N = len(F)
K = N - L + 1
X = np.empty((L, K))
```

51. Самостоятельно заполните элементы данной матрицы $\boldsymbol{X}$ точками массива ряда $F = f(t) = \{f(t_0),...,f(t_{N-1})\}$ через их «вложение» по столбцам $X_i = (f_{i-1},...,f_{i+L-2})^T, 1 \le i \le K$:

$$X = (x_{ij})_{i,j=1}^{L,K} = \begin{pmatrix} f_0 & f_1 & f_2 & \cdots & f_{K-1} \\ f_1 & f_2 & f_3 & \cdots & f_K \\ f_2 & f_3 & f_4 & \cdots & f_{K+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{L-1} & f_L & f_{L+1} & \cdots & f_{N-1} \end{pmatrix}$$

52. Прежде чем переходить дальше, проверьте полученную матрицу на правильность построения для малых значений длины окна L.

53. Теперь в этой функции можно реализовать второй шаг метода SSA – это шаг сингулярного разложения. Сначала нужно создать полную матрицу $S = XX^T$.

```
S = np.dot(X, X.T)
```

54. Для разложения мы используем функцию ниже, через которую сразу же получим нужное **сингулярное разложение** SVD = *Singular Value Decomposition*.

```
U, A, _ = np.linalg.svd(S)
```

Здесь $\boldsymbol{U}$ – матрица собственных векторов, $\boldsymbol{A}$ – массив собственных чисел $\left(\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_L \geq 0\right)$. Третий выходной результат функции нам не понадобится.

55. Еще нужна матрица траекторных векторов $V = X^T U$. Рассчитайте данную матрицы самостоятельно.

56. В результате выполнения своей работы функция `SSA_modes(F, L)` должна вернуть **три** значения: массив собственных чисел $\boldsymbol{A}$, матрицу собственных векторов $\boldsymbol{U}$ и матрицу траекторных векторов $\boldsymbol{V}$.

57. Проверьте правильность и работоспособность данной функции на следующем простом примере:

```
ts = np.array([3, 2, 1, 2, 3, 2, 1, 2, 3, 2, 1, 2, 3]) # мини временной ряд
A, U, V = SSA_modes(ts, 3) # его разложение с длиной окна = 3
print('Название - ', A)
print('Название - ', U)
print('Название - ',V)
```

58. При правильной реализации функции должны получиться следующие результаты:

Для $\boldsymbol{A}$: `[129.66842566  12.          3.33157434]`

Для ***U***: `[[-5.78869570e-01  7.07106781e-01  4.06091149e-01]`
`[-5.74299610e-01  4.14039445e-16 -8.18645196e-01]`
`[-5.78869570e-01 -7.07106781e-01  4.06091149e-01]]`

Для ***V***: `[[-3.46407750e+00  1.41421356e+00 -1.29257973e-02]`
`[-2.88977789e+00  0.00000000e+00  8.05719399e-01]`
`[-3.46407750e+00 -1.41421356e+00 -1.29257973e-02]`
`[-4.03837711e+00  8.88178420e-16 -8.31570994e-01]`
`[-3.46407750e+00  1.41421356e+00 -1.29257973e-02]`
`[-2.88977789e+00  0.00000000e+00  8.05719399e-01]`
`[-3.46407750e+00 -1.41421356e+00 -1.29257973e-02]`
`[-4.03837711e+00  8.88178420e-16 -8.31570994e-01]`
`[-3.46407750e+00  1.41421356e+00 -1.29257973e-02]`
`[-2.88977789e+00  0.00000000e+00  8.05719399e-01]`
`[-3.46407750e+00 -1.41421356e+00 -1.29257973e-02]]`

59. Далее нам нужна функция, которая будет реализовывать этап восстановления ряда. Пусть это будет функция **SSA_group**, у которой входными параметрами являются массив собственных значений ***A***, массив собственных векторов ***U***, массив траекторных векторов ***V***, длина ряда ***N*** и массив группировки компонент ***I***. Выходной параметр всего один – это массив, который содержит отсчеты восстановленного ряда.

60. При вызове этой функции мы, очевидно, передаем ей уже найденные величины `A, U, V, N=len(F)`. Новым параметром будет только ***I***. Длину окна ***L*** можно найти из длины массива собственных чисел: `L = len(A)`. Еще нам понадобится $K = N - L + 1$.

61. Пусть группировку ***I*** мы будем задавать в виде **массива номеров компонент**, которые мы хотим группировать вместе. Тогда шаг группировки выглядит всего двумя строками:

```
V = V.transpose()
Z = np.dot(U[:, I], V[I, :])
```

что соответствует выражению $Z = X_{I_1} + ... + X_{I_m}$

62. Далее идет этап **диагонального усреднения**. Нам потребуется восстановить временной ряд $G = g_0,...,g_{N-1}$ той же длины, что и исходный ряд: `G = np.zeros(N)`. При этом еще понадобится $L^* = \min(L,K)$, $K^* = \max(L,K)$.

63. Далее Вы должны самостоятельно по формулам ниже построить процедуру **диагонального усреднения**.

a.
$$g_k = \begin{cases} \frac{1}{k+1}\sum_{m=0}^{k} Z^*_{m,k-m} & 0 \le k < L^* - 1, \\ \frac{1}{L^*}\sum_{m=0}^{L^*-1} Z^*_{m,k-m} & L^* - 1 \le k < K^*, \\ \frac{1}{N-k}\sum_{m=k-K^*+1}^{N-K^*} Z^*_{m,k-m} & K^* \le k < N+1. \end{cases}$$

64. Проверка работоспособности функции **SSA_group** очень проста – нужно всего лишь сгруппировать полученное разложение из пункта 12 по всем 3 компонентам **[0, 1, 2]** и получить исходный массив:

```
ts1 = SSA_group(A, U, V, len(ts), [0, 1, 2])
print(ts1)
```

```
i.[3. 2. 1. 2. 3. 2. 1. 2. 3. 2. 1. 2. 3.]
```

65. Если все правильно реализовано, для тестового ряда постройте каждую компоненту отдельно (массив группировки [0], затем [1], затем [2]), и их попарные комбинации ([0, 1], [0, 2], [1, 2]). Изобразите их на рисунках относительно исходного временного ряда.

66. Отметьте для себя характерные особенности полученных компонент. Во-первых, 0-компонента содержит некоторое среднее плавающее значение ряда (тренд), а уже 1-компонента и 2-компонента имеют среднее значение близкое к нулю. Во-вторых, 1-компонента и 2-компонента имеют одинаковый период, так как любая периодическая составляющая методом SSA всегда разлагается на парные компоненты. В-третьих, амплитуда 1-компоненты выше амплитуды 2-компоненты, так как массив собственных чисел

упорядочен по убыванию, то есть с ростом номера компоненты ее «вклад» в исходный ряд уменьшается.

67. Важно отметить, что с ростом длины окна ***L*** разложения, все составляющие ряда будут «расплываться» по нескольким компонентам. То есть тренд не всегда есть 0-компонента, а скорее комбинация компонент с номерами близкими к нулю, а периодика не равна одной паре компонент, а есть комбинация нескольких пар с близкими номерами. Тем не менее, общий характер особенностей сохраняется.

68. Теперь применим готовый метод SSA к некоторым периодическим временным рядам.

69. Постройте следующий модельный ряд из 2 периодик с шумом:

```
t = np.linspace(0, 1, 1024)
f1 = 10
f2 = 50
F=1.7*np.sin(2*np.pi*f1*t)+np.sin(2*np.pi*f2*t)+0.2*rand.randn(len(t))
plt.figure(figsize = (10, 5))
plt.plot(t, F, 'k')
plt.plot(t, 1.7*np.sin(2*np.pi*f1*t), 'b')
plt.plot(t, np.sin(2*np.pi*f2*t), 'r')
plt.show()
```

здесь черным изобразится модельный ряд с шумом, синим – первая периодика, красным – вторая периодика.

70. Самостоятельно подберите такую длину окна и метод группировки компонент, чтобы с помощью метода SSA выделить компоненты, наиболее близкие к исходным периодикам в модельном ряде. Постройте их на рисунке совмещенно с исходным зашумленным рядом ***F***.

71. Теперь аналогичной методикой попытаемся построить тренд для сильно зашумленного ВР. Пусть задан ВР:

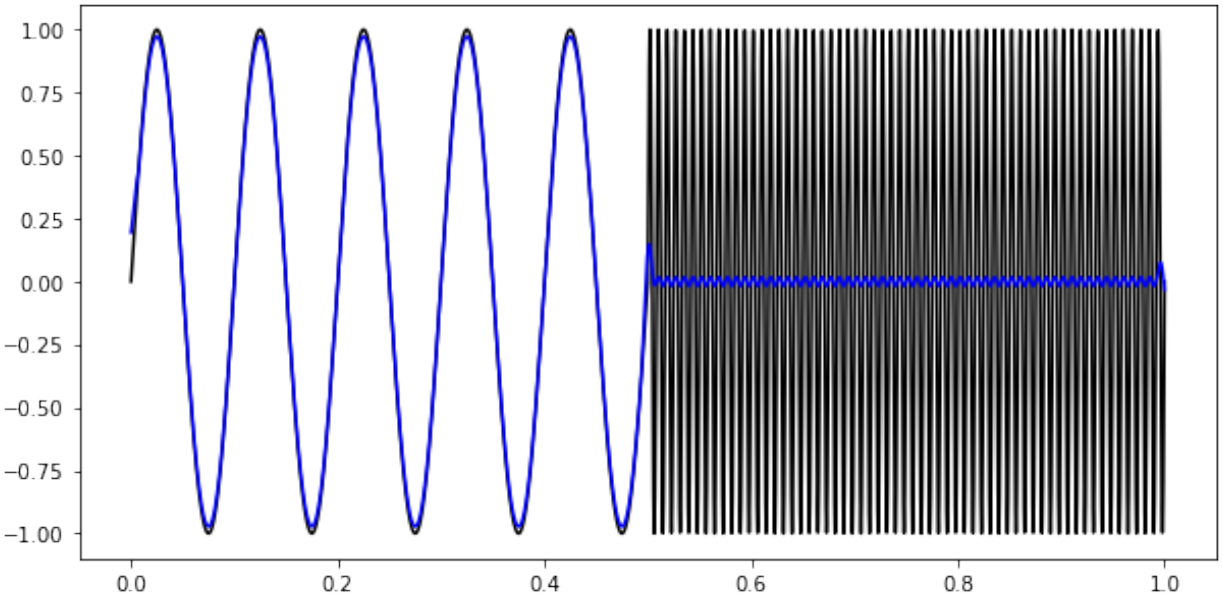
```
t = np.linspace(0,4,4096)
F = np.exp(-0.4*np.pi*t) + 0.5*rand.randn(len(t))
```

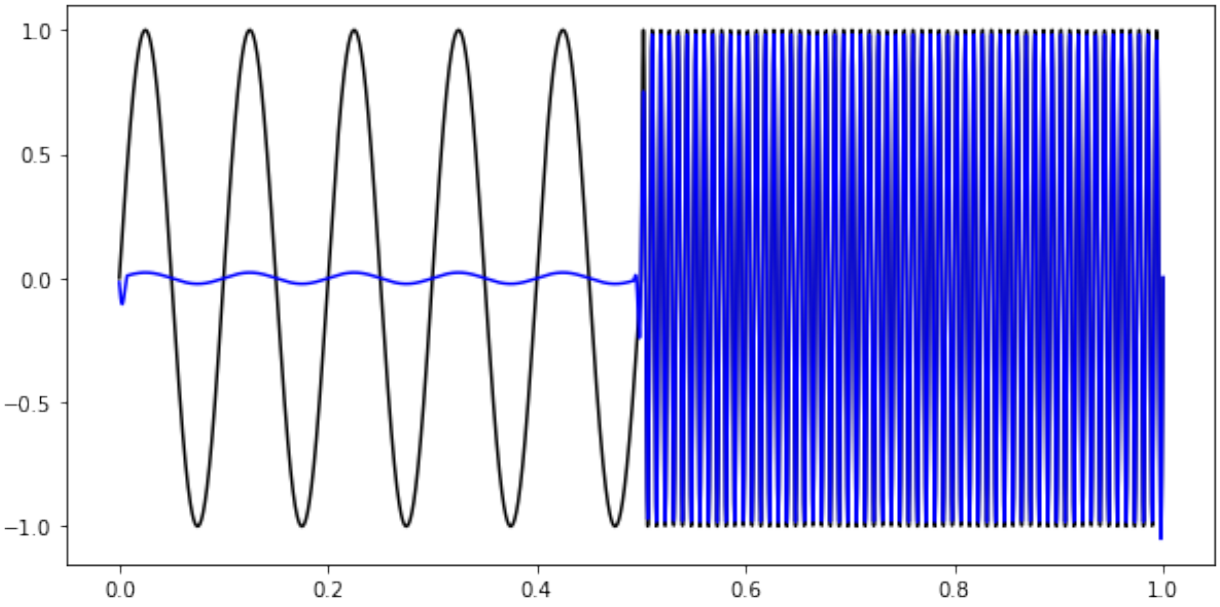
72. Используя метод SSA выделите этот экспоненциальный тренд **np.exp(-0.4*np.pi*t)**. Длину окна и метод группировки определите сами. Постройте на графиках исходный тренд и тот, что был получен Вами с помощью метода SSA.

73. Создайте периодический сигнал с **изломом частоты** смотри в прошлых работах.

74. Самостоятельно подберите такую длину окна и метод группировки компонент, чтобы с помощью метода SSA выделить компоненты, наиболее близкие к исходным периодикам на двух половинках временного интервала в модельном ряде. Постройте их вместе с исходным рядом на одном рисунке.

75. Например, должен получиться результат как на рисунках:





76. Смоделируйте временной ряд из **4 гармоник с шумом**, и разделите его на компоненты с помощью метода SSA:

$$u(t) = 2\sin(2\pi f_1 t),\ 1{,}5\sin(2\pi f_2 t),\ 0{,}8\sin(2\pi f_3 t),\ 0{,}5\sin(2\pi f_4 t) + \xi$$

Ряд строится для `t = np.linspace(0,1,1024)` строятся полученные результаты. Результаты оформить в виде графиков полученных компонент и компоненты, описанные в формуле $2\sin(2\pi f_1 t),\ 1{,}5\sin(2\pi f_2 t),\ 0{,}8\sin(2\pi f_3 t),\ 0{,}5\sin(2\pi f_4 t)$.

Вариант	1,20	2,19	3,18	4,17	5,16	6,15	7,14	8,13	9,12	10,11
f1	10	20	30	40	50	60	10	20	30	10
f2	40	30	15	10	20	10	50	40	10	30
f3	100	110	90	130	115	100	105	100	125	100
f4	150	120	170	150	160	180	150	150	160	150

77. Теперь на основе метода SSA реализуем **прогноз** временных рядов. Пусть в результате декомпозиции методом SSA получены все необходимые собственные тройки $\left(\sqrt{\lambda_i}, U_i, V_i\right)$. Прогноз строится на $\boldsymbol{M}$ точек вперед.

78. Тогда прогноз методом **SSA-R** строится следующей последовательностью действий. Сначала вычислим норму последнего вектора из матрицы ***U*** для заданной группировки компонент:

```
vu = np.linalg.norm(U[-1, I])
```

79. Нам надо вычислить ряд весовых коэффициентов:

$$R = \left(a_{L-2},...,a_0\right)^T = \frac{1}{1-v^2}\sum \pi_i P_i^{\nabla}$$

для чего потребуется следующая последовательность команд:

```
R = np.sum(U[L - 1, I] * U[0:L - 1, I], 1)
R = R / (1 - vu * vu)
```

80. Пусть ВР, восстановленный методом SSA по группировке ***I*** компонент, называется ***G***, а ряд новой длины ***N***+***M***, то есть восстановленный ряд и его прогноз, называется ***Q***. Тогда

$$Q_i = \begin{cases} g_i & ,i < N \\ \sum_{j=0}^{L-2} a_j g_{i-j-1}, & i = N,...,N+M-1 \end{cases}$$

81. Постройте прогноз методом SSA-R для ряда из пункта 23 (две периодики) на **256** точек вперед, подберите для него наилучшие параметры. Сравните – совпадают ли параметры длины окна ***L*** и группировки ***I*** для высокой точности декомпозиции и прогноза, или нет?

82. Постройте прогноз методом SSA-R для ряда из пункта 30 на **256** точек вперед, подберите для него наилучшие параметры. Сравните полученные параметры с теми, что были при декомпозиции.

83. Используйте ранее загруженный **Fort.mat** временной ряд, содержащий отсчеты некоторого реального ВР, всего 174 отсчета в вектор-строке:

84. Постройте его **ретроспективный прогноз** методом SSA-R, подберите параметры самостоятельно. Начальная точка прогноза определяется студентом самостоятельно. **Длина прогноза** студентами выбирается самостоятельно, она должна быть **не меньше 24** отсчетов. Графики исходного ряда *Fort* и прогноза строятся вместе, так как они имеют малую длину и вполне могут поместиться рядом с достаточной точностью.

85. Теперь самостоятельно реализуйте метод прогноза на основе SSA с итерационной аппроксимацией (название в лекциях – стохастический SSA-прогноз).

86. Проведем *частичный* SSA-анализ заданного ВР: **без диагонального усреднения** (последний этап).

87. К ВР добавляется всего **один новый случайный отсчет** из диапазона уже имевшихся уровней ряда:

$$F'_{N+1} = \left(f_0, ..., f_{N-1}, f_{new}\right), f_{new} \in \left[\min(F); \max(F)\right]$$

88. Этот ряд длины ***N***+1 подвергается SSA-декомпозиции (**только** шаги разложения и формирования траекторной матрицы), но **без изменения оценки параметров**, то есть только на основе их предыдущих оценок.

89. Полученные собственные тройки нового ряда **группируются и усредняются** на основе метода группировки ***I***.

90. В результате **усреднения** будет получен новый временной ряд, для которого первые ***N*** отсчетов совпадают с ВР, а последний отсчет является **пред-прогнозом.**

91. Для получения точного прогноза, новый отсчет ряда приравнивается этому приближению, после чего **предыдущие шаги повторяются** до тех пор, пока **значение не перестанет изменяться** с увеличением числа шагов.

92. Полученный в результате отсчет принимается за **первую точку прогноза**. Для продолжения прогноза, новый ряд длины *N*+1 становится ВР для прогнозирования, и алгоритм **повторяется вновь.** На протяжении всего

алгоритма прогноза нет необходимости заново искать необходимую группировку компонент.

93. Постройте ретроспективный прогноз данным методом для ряда **Fort**.

94. Не забудьте в отчет-тетрадь добавить необходимые рисунки и таблицы результатов.

95. Сформулировать и оформить выводы по полученным результатам и лабораторной работе.

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. Код должен сопровождаться пояснениями, а результаты описанием того, что вы наблюдаете. Не забывайте подписывать и оформлять полученные графические результаты.