

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

МЕТОД SSA – «ГУСЕНИЦА»

**Методические указания к выполнению
практического задания № 7**

Екатеринбург

2020

Содержание

Введение.....	3
1. Задание на лабораторную работу	3
2. Требования к оформлению отчета.....	15

Введение

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  
%matplotlib inline
```

- 2) 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.

- 3) 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 .

- 4) Внутри функции нам понадобится определить размерность траекторной X матрицы $L \times K$.

$N = \text{len}(F)$

$K = N - L + 1$

$X = \text{np.empty}((L, K))$

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

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

- 6) Прежде чем переходить дальше, проверьте полученную матрицу на правильность построения для малых значений длины окна L .
- 7) Теперь в этой функции можно реализовать второй шаг метода SSA – это шаг сингулярного разложения. Сначала нужно создать полную матрицу $S = XX^T$.

$S = \text{np.dot}(X, X.T)$

- 8) Для разложения мы используем функцию ниже, через которую сразу же получим нужное **сингулярное разложение** $SVD = \text{Singular Value Decomposition}$.

$U, A, _ = \text{np.linalg.svd}(S)$

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

- 9) Еще нужна матрица траекторных векторов $V = X^T U$. Рассчитайте данную матрицу самостоятельно.

- 10) В результате выполнения своей работы функция **SSA_modes(F, L)** должна вернуть **три** значения: массив собственных чисел **A**, матрицу собственных векторов **U** и матрицу траекторных векторов **V**.
- 11) 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(A) # собственные числа
print(U) # собственные вектора
print(V) # траекторные вектора
```

- 12) При правильной реализации функции должны получиться следующие результаты:

```
Для 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]]
```

- 13) 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.
- 14) При вызове этой функции мы, очевидно, передаем ей уже найденные величины **A**, **U**, **V**, $N=\text{len}(F)$. Новым параметром будет только **I**. Длину окна **L** можно найти из длины массива собственных чисел: $L = \text{len}(A)$. Еще нам понадобится $K = N - L + 1$.
- 15) 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  $Z = X_{I_1} + \dots + X_{I_m}$
- 16) Далее идет этап **диагонального усреднения**. Нам потребуется восстановить временной ряд  $G = g_0, \dots, g_{N-1}$  той же длины, что и исходный ряд: **G = np.zeros(N)**. При этом еще понадобится  $L^* = \min(L, K)$ ,  $K^* = \max(L, K)$ .

- 17) Далее Вы должны самостоятельно по формулам ниже построить процедуру **диагонального усреднения**.

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

- 18) 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)
```

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

- 19) 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.
- 20) 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.

21) 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 из пункта 20 сохраняется.

22) Теперь применим готовый метод SSA к некоторым периодическим временным рядам.

23) 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.

24) 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  $F$ .



- 25) 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))
```

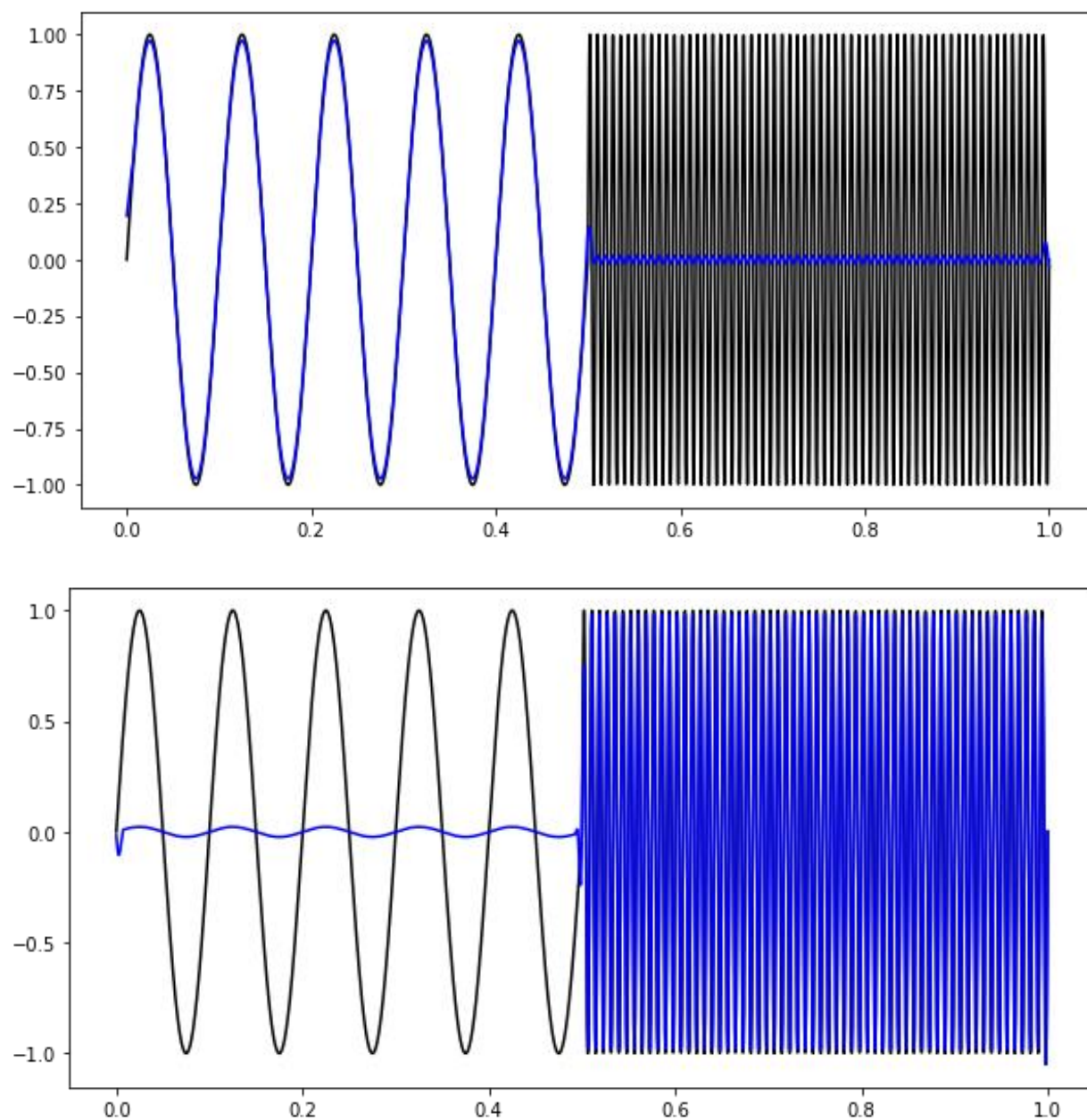
- 26) Используя метод SSA выделите этот экспоненциальный тренд  **$\text{np.exp}(-0.4 \cdot \pi \cdot t)$** . Длину окна и метод группировки определите сами. Постройте на графиках исходный тренд и тот, что был получен Вами с помощью метода SSA.

- 27) 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)
plt.show()
```

- 28) 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.

29) Например, должен получиться результат как на рисунках:



- 30) Смоделируйте временной ряд из **4 гармоник с шумом**, и разделите его на компоненты с помощью метода SSA:

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

```
t = np.linspace(0,1,1024)
f1 = 10
f2 = 40
f3 = 100
f4 = 150
F=2.0*np.sin(2*np.pi*f1*t)+1.5*np.sin(2*np.pi*f2*t)
 +0.8*np.sin(2*np.pi*f3*t)
 +0.5*np.sin(2*np.pi*f4*t)+0.2*rand.randn(len(t))
plt.figure(figsize = (10, 15))
plt.subplot(5,1,1)
plt.plot(t, F, 'k')
plt.subplot(5,1,2)
plt.plot(t, 2.0*np.sin(2*np.pi*f1*t), 'b')
plt.subplot(5,1,3)
plt.plot(t, 1.5*np.sin(2*np.pi*f2*t), 'r')
plt.subplot(5,1,4)
plt.plot(t, 0.8*np.sin(2*np.pi*f3*t), 'g')
plt.subplot(5,1,5)
plt.plot(t, 0.5*np.sin(2*np.pi*f4*t), 'm')
plt.show()
```

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

32) 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])`**

33) Нам надо вычислить ряд весовых коэффициентов:

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

для чего потребуется следующая последовательность команд:

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

**`R = R / (1 - vu * vu)`**

34) Пусть  $BP$ , 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

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

- 35) 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?
- 36) 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.
- 37) Загрузите из mat-файла **Fort.mat** ряд, содержащий отсчеты некоторого реального ВР, всего 174 отсчета в вектор-строке:
- ```
file = h5py.File('Fort.mat','r')
data = file.get('Fort')
fort = np.array(data)
F = np.ravel(fort)
plt.figure(figsize = (10, 5))
plt.plot(F)
plt.show()
```
- 38) 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.

- 39) Теперь самостоятельно реализуйте метод прогноза на основе SSA с итерационной аппроксимацией (название в лекциях – стохастический SSA-прогноз).
- 40) Проведем *частичный* SSA-анализ заданного ВР: **без диагонального усреднения** (последний этап).
- 41) К ВР добавляется всего **один новый случайный отсчет** из диапазона уже имевшихся уровней ряда:

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

- 42) 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.
- 43) Полученные собственные тройки нового ряда **группируются и усредняются** на основе метода группировки I .
- 44) В результате **усреднения** будет получен новый временной ряд, для которого первые N отсчетов совпадают с ВР, а последний отсчет является **пред-прогнозом**.
- 45) 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.
- 46) 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.
- 47) Постройте ретроспективный прогноз данным методом для ряда **Fort**.

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.