Программа повышения конкурентоспособности

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

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

ПРИМЕНЕНИЕ СИНГУЛЯРНОГО СПЕКТРАЛЬНОГО АНАЛИЗА ДЛЯ ДЕКОМПОЗИЦИИ ВРЕМЕННОГО РЯДА

Методические указания к выполнению
лабораторной работы № 5

Екатеринбург
2017

Содержание

Цель изучения материала

Целью данной лабораторной работы является реализация алгоритма SSA средствами языка MATLAB, а также его апробация на модельных ВР с целью научиться и овладеть навыками по выбору его параметров, метода группировки компонент и построения графиков, помогающих проводить методику анализа ВР методом «Гусеница».

Перечень компетенций, формирующихся или получающих приращение в результате прослушивания лекции

Способность анализировать профессиональную информацию, выделять в ней главное, структурировать, оформлять и представлять в виде аналитических обзоров с обоснованными выводами и рекомендациями.

Умение разрабатывать стратегии проектирования, определением целей проектирования, критериев эффективности, ограничений применимости.

Умение проводить разработку и исследование теоретических и экспериментальных моделей объектов профессиональной деятельности в областях науки, техники.

Список литературы

1. Голяндина Н.Э. Метод «Гусеница»-SSA: анализ временных рядов: Учеб. пособие. — СПб. — 2004. —76 с.

2. Golyandina, N., and A. Zhigljavsky Singular Spectrum Analysis for time series. — Springer Briefs in Statistics. — Springer. — 2013. — 120 с. ISBN 978-3-642-34912-6.

3. Golyandina N., Nekrutkin V., Zhigljavsky A. A. Analysis of Time Series Struc ture. SSA and Related Techniques. — Monographs on Statistics and Applied Probability 90. — Florida 33431: Chapman and Hall/CRC. — 2000. —296 с.

4. Jolliffe I. T. Principal Component Analysis. Springer Series in Statistics. 2nd edition. — Springer, NY, 2002. — P. 487. ISBN: 978-0-387-95442-4.

5. Ефимов В. М., Галактионов Ю. К., Шушпанова Н. Ф. Анализ и прогноз временных рядов методом главных компонент. — Новосибирск: Наука, Сибирское отделение. — 1988. — С. 70.

1. Введение

Метод сингулярного спектрального анализа SSA относится к адаптивным методам, и, потому, весьма эффективен для анализа и декомпозиции 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*. Поэтому, что избежать ситуации нехватки памяти, следует ограничить выбор длины окна значениями, не превосходящими 1000.

2. Задание на лабораторную работу

1) К сожалению, на этот раз в MATLAB нет встроенных функций, реализующих сразу метод сингулярного спектрального анализа. Но мы его можем довольно быстро реализовать средствами MATLAB.
2) Нам потребуется написать несколько функций MATLAB. Для начала напишем функцию, которая будет реализовывать **этап разложения**.
3) Назовем эту функцию **SSA_modes(F, L)**, ее входным параметрами должны быть исходный временной ряд *F* и длина окна *L*. Длину ряда *N* мы можем посчитать и внутри самой функции. На выходе метода мы получаем массив собственных чисел **lambda**, массив собственных векторов **U** и массив траекторных векторов **V**. Все вместе это называется **собственными тройками** компонент.
4) Внутри этой функции реализуются два шага алгоритма SSA – вложение и непосредственно разложение.
5) Сначала определим число $K = N - L + 1$.

6) Затем построим траекторную матрицу $X_i = \left(f_{i-1}, ..., f_{i+L-2} \right)^T, 1 \le i \le K$.

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

7) Сделать это придется в цикле. Самым оптимальным решением будет использовать всего один цикл, а внутри него обеспечивать «вложение» по столбцам матрицы. Но в целом реализация остается на усмотрение студентов.
8) Прежде чем переходить дальше, проверьте полученную матрицу на правильность построения для малых значений длины окна *L*.
9) Теперь в этой функции можно реализовать второй шаг метода SSA – это шаг сингулярного разложения.
10) Есть два способа построения такого разложения. Мы реализуем их оба, чтобы убедиться в их правильности, но для конечного варианта модели оставим только один.
11) Первый способ проще – для разложения мы используем функцию MATLAB **[*U*,*A*,*V*]=svd(*X*);** через которую сразу же получим нужное **сингулярное разложение**. SVD = *Singular Value Decomposition*. Здесь матрица ***A*** – это диагональная матрица собственных чисел, откуда можно найти эти собственные числа просто взяв эти диагональные элементы: **lambda=diag(A);**
12) Нужно еще вспомнить, что для нас важно, чтобы собственные числа были упорядочены в виде $\left(\lambda_1 \ge \lambda_1 \ge ... \ge \lambda_L \ge 0 \right)$. К счастью, функция **[*U*,*A*,*V*]=svd(*X*);** в последних версиях MATLAB уже производит эту сортировку автоматически.

13) Таким образом, можно реализовать шаг сингулярного разложения всего в две строки. Но для изучения и понимания метода мы все-таки реализуем его в лоб, по формулам.

14) Итак, для разложения мы вычисляем сначала матрицу $S = XX^T$.

15) Теперь ищем **собственные числа** и **собственные вектора** этой матрицы с помощью функции **[U,A]=eig(S);**

16) Аналогично вычисляем **lambda=diag(A);**

17) Но собственные числа теперь **не отсортированы** в виде $(\lambda_1 \geq \lambda_1 \geq ... \geq \lambda_L \geq 0)$. Для сортировки массивов в MATLAB есть функция **sort**, которая получает массив и метод сортировки. Нам нужно отсортировать в виде **sort(lambda, 'descend').**

18) Проблема в том, что у нас не отсортированы и собственные вектора. Поэтому мы не просто сортируем собственные числа, а еще и запоминаем происходящие перестановки.

19) В связи с этим итоговая функция сортировки будет выглядеть следующим образом: **[lambda, i]=sort(lambda , 'descend');**

20) В это выражении: *lambda* – массив собственных чисел, *i* – массив перестановок, сортируем по принципу $(\lambda_1 \geq \lambda_1 \geq ... \geq \lambda_L \geq 0)$.

21) Далее просто в матрице ***U*** нужно подставить эти перестановки в столбцы: **U=U(:,i);**

22) Стоп, мы забыли про траекторные вектора $V = X^T U$? А нет, не забыли – вот же они: **V=(X')*U;**

23) На этом реализацию функции **SSA_modes(F, L)** можно считать законченной. Она вернет три массива ***U, lambda, V***.

24) Реализуйте оба способа построения сингулярного разложения и убедитесь в том, что они совпадают. Результаты совпадения продемонстрируйте в отчете.

25) Далее нам нужна функция, которая будет реализовывать этап восстановления ряда. Пусть это будет функция **SSA_group()**, у которой входными параметрами являются та же самая длина окна ***L***, массив собственных векторов ***U***, массив собственных отсортированных значений ***lambda***, массив траекторных векторов ***V***, и массив группировки компонент *I*. Выходной параметр всего один – это массив, который содержит отсчеты восстановленного ряда.

26) При вызове этой функции мы, очевидно, передаем ей уже найденные величины ***L, U, lambda, V***. Новым параметром будет только ***I***.

27) Пусть группировку *I* мы будем задавать в виде **массива номеров компонент**, которые мы хотим группировать вместе. Тогда шаг группировки выглядит всего одной строчкой: **y_zv=U(:,I)*V(I,:)**;

28) Эта строка кода соответствует $y^*_{ij} = y_{ij}$, $X = X_{I_1} + ... + X_{I_m}$.

29) Далее выделяем место под наш новый ряд: **G=zeros(1,N);** Восстановленный ряд будет иметь название ***G***. Вычисляем снова **K=N-L+1;** и при необходимости меняем их местами:

Lp=min(L,K); Kp=max(L,K); $L^* = \min(L,K), K^* = \max(L,K)$

30) Далее идет этап **диагонального усреднения**. Здесь Вы должны самостоятельно по формулам ниже построить данную процедуру. К сожалению, здесь нет красивого решения без циклов с помощью векторных преобразований, поэтому задачу надо решать в лоб.

$$g_k = \begin{cases} \dfrac{1}{k+1}\sum_{m=1}^{k+1} y^*_{m,k-m+2} & 0 \le k < L^* - 1, \\ \dfrac{1}{L^*}\sum_{m=1}^{L^*} y^*_{m,k-m+2} & L^* - 1 \le k < K^*, \\ \dfrac{1}{N-k}\sum_{m=k-K^*+2}^{N-K^*+1} y^*_{m,k-m+2} & K^* \le k < N. \end{cases}$$

31) *Подсказка*: Вам понадобятся три цикла: по количеству строк в формуле выше.

32) Итак, у нас уже есть две функции, которых достаточно для реализации метода.

33) Постройте в среде MATLAB временной ряд, состоящий из отсчетов двух гармоник с шумом:

f1=10; f2=45; t=0:0.001:4;

F= sin(2*pi*f1*t)+sin(2*pi*f2*t)+0.1*randn(1,length(t));

34) Исходный ВР выглядит следующим образом:

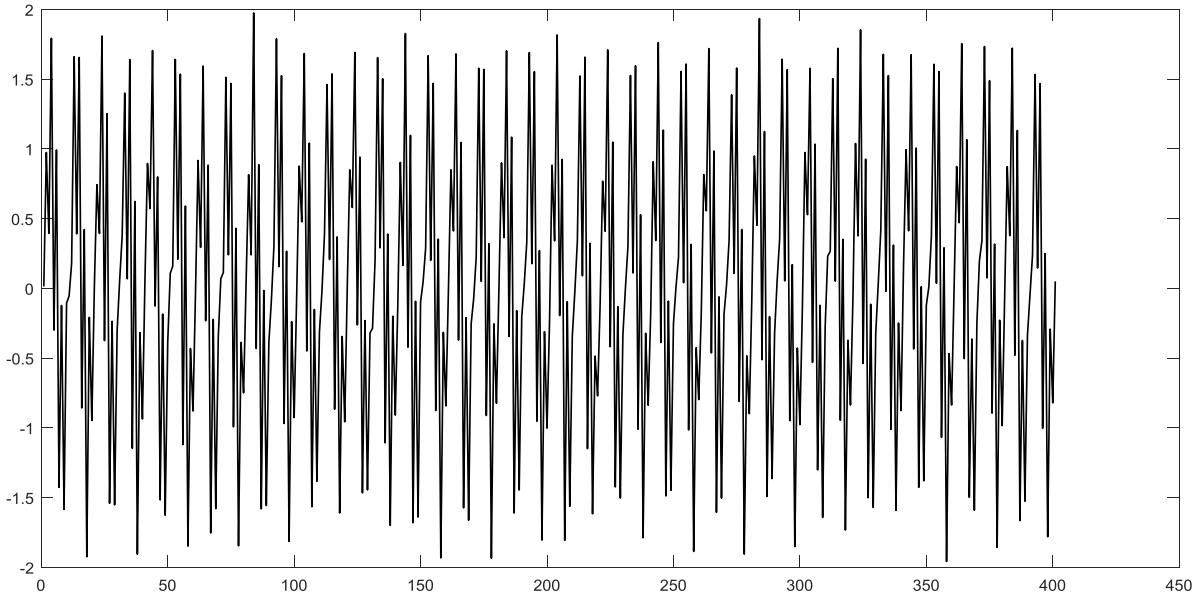

Рисунок 1 – ВР из двух гармоник с шумом

35) Давайте подумаем, какие группировки нам необходимо произвести, и какую длину окна нужно выбрать. У нас здесь две гармонических компоненты, тогда на основе методики приведенной в лекции 6, нам потребуется на каждую гармонику по два собственных числа. Так как мощность шума мала, то можно ожидать выделить компоненты **в первых 4 собственных тройка**х.

36) Как выбрать длину окна? В самом простом случае, надо попробовать сначала длину окна в половину ВР. Но мы обойдемся для скорости и четвертью от числа отсчетов, то есть возьмем $L = 100$.

37) Определяем еще способ группировки. Для ее нахождения используем методику, приведенную в лекции 6: ищем близкие собственные тройки. Ими оказываются группировки вида [1 3] и [2 4].

38) Вызываем наши реализованные функции.

39) Сначала функцию разбиения **[lambda,U,V]=SSA_modes(F,100);**

40) Затем функцию восстановления для первой гармоники

G1=SSA_group(100,lambda,U,V,[1 3]);

И для второй гармоники

G2=SSA_group(100,lambda,U,V,[2 4]);

41) Полученные в результате гармоники приведены на рисунке 2.

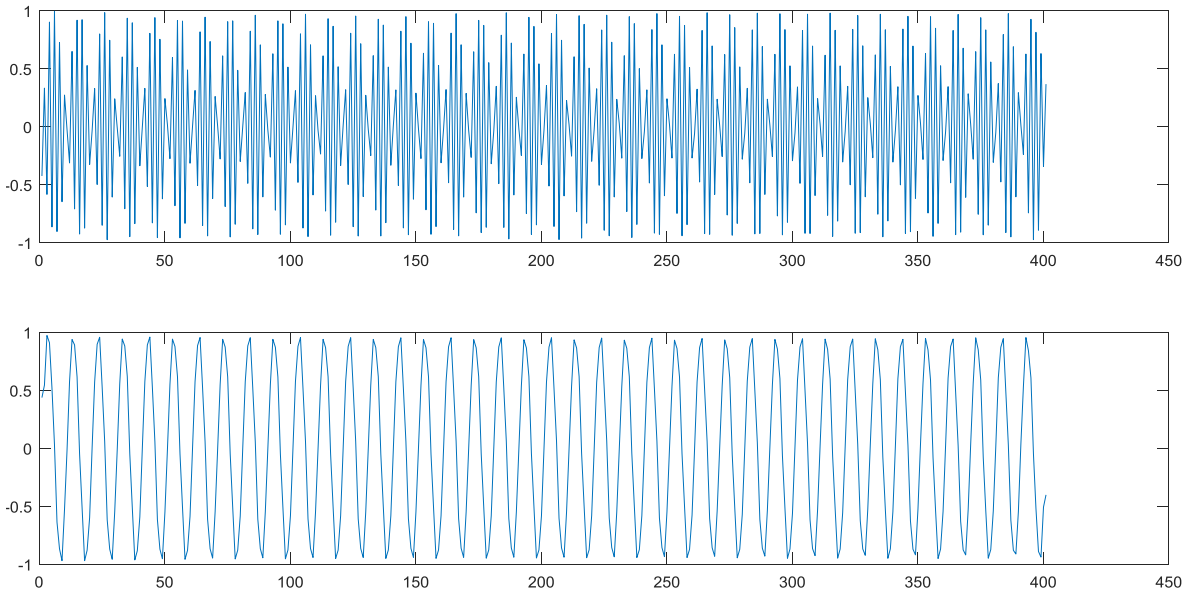

Рисунок 2 – Выделенные из ВР гармоники

42) Первая гармоника имеет ну очень низкую точность, а вот вторая выглядит неплохо.

43) **Самостоятельное задание**: попробуйте подобрать такую **длину окна** и такой **способ группировки**, который обеспечит хорошее выделение **первой гармоники**.

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

F = exp(-pi*t) + 0.5*randn(1,length(t));

45) Используя метод SSA и приведенные выше функции найдите для него такую **длину окна** и такую **группировку**, которая позволит выделить этот экспоненциальный тренд, как на рисунке 3.

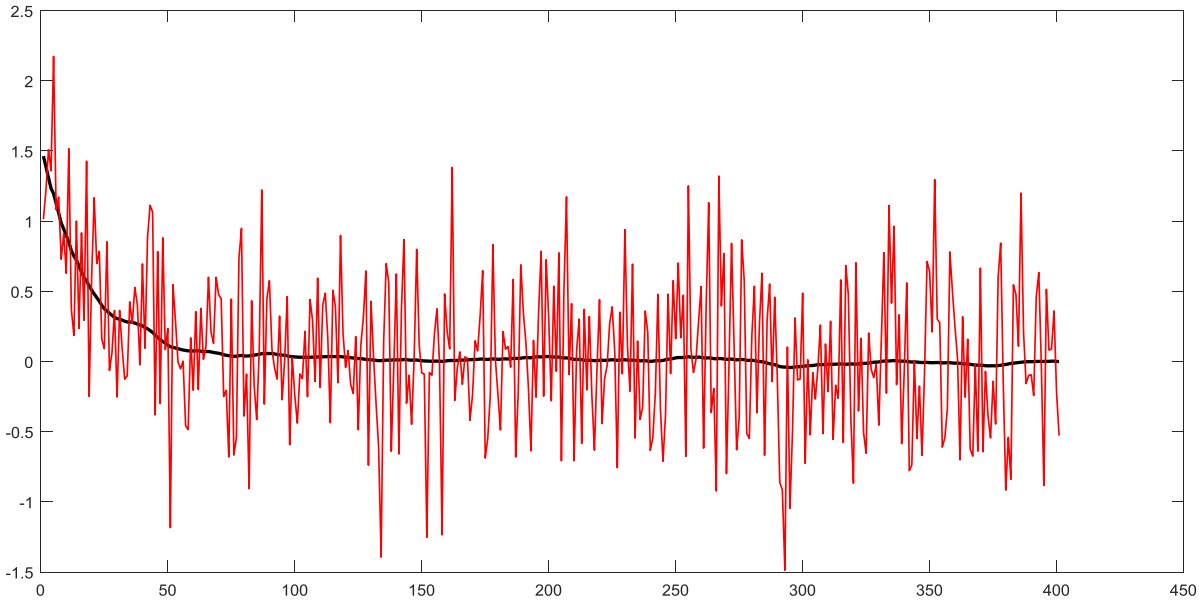

Рисунок 3 – Тренд, полученный методом SSA

46) Для более точного поиска длины окна и метода группировки напишите в MATLAB функцию, которая строит графики разности соседних собственных чисел, для реализации методики, описанной в **лекции 6 на стр. 15-16**.

47) **Самостоятельно** смоделируйте ВР из **4 гармоник с шумом**, и разделите его на компоненты с помощью метода SSA.

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

3. Требования к оформлению отчета

Отчет должен обязательно содержать: постановку задачи, результаты выполнения пунктов с 1 по 47, графики соответствующих зависимостей с пояснениями, объяснения, которые требовались в ходе работы, заключение. Обязательно не забудьте выполнить пункты **43**, **46** и **47**, так как они наиболее сложны и требовательны по объему выполненной работы. Также весь код функций и сценариев добавляется в приложение в конце. У отчета должен быть оформлен грамотный титульный лист с указанием названия дисциплины, номера работы, фамилии преподавателя и ученика.