ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

https://doi.org/10.31649/1997-9266-2022-163-4-41-55

УДК 517.98

О. Б. Мокін[1]
В. Б. Мокін[1]
Б. І. Мокін[1]

АЛГОРИТМ МЕТОДУ ІДЕНТИФІКАЦІЇ МОДЕЛІ АВТОРЕГРЕСІЇ – КОВЗНОГО СЕРЕДНЬОГО, ЯКИЙ УЗАГАЛЬНЮЄ МЕТОДИКУ ЮЛА–УОКЕРА, ТА ЙОГО ПРОГРАМНА PYTHON-РЕАЛІЗАЦІЯ

[1]Вінницький національний технічний університет

Для практичної реалізації розробленого цими ж авторами нового методу ідентифікації математичної моделі авторегресії – ковзного середнього АРКС $\left(n_{ap},n_{kc}\right)$ прогнозування стаціонарних часових рядів з довільними значеннями порядків n_{ap},n_{kc}, який узагальнює відомий метод Юла–Уокера та вже опублікований в попередніх роботах цих же авторів, запропоновано та деталізовано 11-етапний алгоритм його практичної реалізації. Алгоритм реалізовано за умови, доведеної авторами у попередніх публікаціях, що оптимальною структурою моделі АРКС $\left(n_{ap},n_{kc}\right)$ є структура АРКС(3,3). Характерною особливістю цього алгоритму є те, що параметри авторегресійної складової моделі АРКС(3,3) визначаються з використанням четвертої, п'ятої та шостої автоковаріацій, що суттєво відрізняє його від традиційного алгоритму ідентифікації цього класу моделей за методикою Юла–Уокера, в якому використовуються лише автоковаріації першого, другого та третього порядків. Інша характерна особливість цього алгоритму полягає в тому, що для визначення параметрів ковзного середнього застосовується пряма процедура, яка не вимагає поновлення процедури мінімізації суми квадратів відхилень при переході до інших значень порядків авторегресії та ковзного середнього, як того вимагає процедура обчислення значень параметрів складової ковзного середнього в моделі за будь-яким з класичних методів ідентифікації цього класу моделей. Створено програму Python-реалізації запропонованого алгоритму ідентифікації та продемонстровано її ефективність у розв'язанні задачі ідентифікації математичної моделі класу АРКС(3,3) для конкретного часового ряду, заданого його експериментальною реалізацією. Визначено умови, яким повинна відповідати експериментальна реалізація часового ряду, з використанням якої здійснюється авторська ідентифікація математичної моделі цього часового ряду, щоб прогнозування його наступних значень здійснювалось точніше, ніж з використанням математичних моделей цього ж класу, ідентифікація яких здійснювалась традиційно.

Ключові слова: стаціонарний часовий ряд, модель авторегресії – ковзного середнього, АРКС, ідентифікація моделі, узагальнення методу Юла–Уокера, програма Python-реалізації алгоритму ідентифікації моделі.

Вступ

В роботі [1] авторами цієї статті запропоновано новий метод ідентифікації моделі авторегресії – ковзного середнього АРКС $\left(n_{ap},n_{kc}\right)$ для прогнозування стаціонарних часових рядів з довільними значеннями порядків n_{ap},n_{kc}, який узагальнює методику Юла–Уокера, детально викладену в роботах [2], [3].

В роботі [4] авторами статті доведено, що оптимальною структурою моделі АРКС $\left(n_{ap},n_{kc}\right)$ є структура АРКС(3,3).

Але ні в роботі [1], ні в роботі [4] не подано ні робочого алгоритму реалізації запропонованого

нами нового методу ідентифікації моделі АРКС$\left(n_{ap}, n_{kc}\right)$, ні прикладу реалізації цього методу в задачі ідентифікації моделі конкретного часового ряду.

А тому *метою цієї публікації* є розкриття та деталізація алгоритму методу ідентифікації моделі АРКС$\left(n_{ap}, n_{kc}\right)$ в її оптимальному варіанті АРКС(3,3) та програмна Python-реалізація цього алгоритму в задачі прогнозування конкретного часового ряду, яка здійснена з використанням інформації, почерпнутої з робіт [5], [6].

Алгоритмізація узагальненого методу ідентифікації моделі АРКС(3,3)

Отже, використовуючи висновок, отриманий в нашій роботі [4], в якості першої вихідної передумови побудови алгоритму узагальненого методу ідентифікації моделі АРКС(3,3), представимо оптимальну еквівалентну математичну модель АРКС(3,3) стаціонарного часового ряду $l_t, t \in [0, 1, 2, \ldots, N-1]$ у вигляді

$$m_t = g_1 m_{t-1} + g_2 m_{t-2} + g_3 m_{t-3} + a_t - p_1 a_{t-1} - p_2 a_{t-2} - p_3 a_{t-3}; \tag{1}$$

$$m_t = l_t - b; \tag{2}$$

$$b = \frac{1}{N} \sum_{t=0}^{N-1} l_t, \tag{3}$$

де a_t — імпульс стаціонарного білого шуму, згенерований в заданому діапазону значень, g_1, g_2, g_3 — параметри авторегресійної складової моделі АРКС(3,3), а p_1, p_2, p_3 — параметри складової ковзного середнього цієї моделі.

Як другу вихідну передумову створюваного алгоритму використано отриману у нашій роботі [1] систему рівнянь

$$\begin{cases} \left\{q_0 = g_1 q_1 + g_2 q_2 + g_3 q_3 + \sigma_a^2 - p_1\left(g_1 - p_1\right)\sigma_a^2 - p_2\left(g_2 - p_2\right)\sigma_a^2 - p_3\left(g_3 - p_3\right)\sigma_a^2\right\}, \\ \left\{q_1 = g_1 q_0 + g_2 q_1 + g_3 q_2 - p_1\sigma_a^2 - p_2(g_1 - p_1)\sigma_a^2 - p_3(g_2 - p_2)\sigma_a^2\right\}, \\ \left\{q_2 = g_1 q_1 + g_2 q_0 + g_3 q_1 - p_2\sigma_a^2 - p_3(g_1 - p_1)\sigma_a^2\right\}, \\ \left\{q_3 = g_1 q_2 + g_2 q_1 + g_3 q_0 - p_3\sigma_a^2\right\}, \end{cases} \tag{4}$$

$$\begin{cases} q_4 = g_1 q_3 + g_2 q_2 + g_3 q_1, \\ q_5 = g_1 q_4 + g_2 q_3 + g_3 q_2, \\ q_6 = g_1 q_5 + g_2 q_4 + g_3 q_3, \end{cases} \tag{5}$$

яку для зручності реалізації в Python-програмі представлено у вигляді (4)—(5) і в якій σ_a^2 — дисперсія стаціонарного білого шуму, а q_i, $i = 0, 1, 2, 3, 4, 5, 6$ — дисперсія та автоковаріації центрованого стаціонарного часового ряду m_t, що обчислюються за виразом

$$q_i = \frac{1}{(N-1-i)} \sum_{t=0}^{N-1} m_t m_{(t+i)}, \quad i = 0, 1, 2, 3, 4, 5, 6. \tag{6}$$

Як третю вихідну передумову створюваного алгоритму використаємо систему рівнянь (5), яка в матричній формі матиме вигляд

$$Mg = q, \tag{7}$$

де

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}; \quad M = \begin{bmatrix} q_3 & q_2 & q_1 \\ q_4 & q_3 & q_2 \\ q_5 & q_4 & q_3 \end{bmatrix}; \quad q = \begin{bmatrix} q_4 \\ q_5 \\ q_6 \end{bmatrix}, \tag{8}$$

та розв’язок цієї системи у вигляді

$$g = M^{-1} q, \tag{9}$$

у якому M^{-1} — матриця, обернена до матриці M, заданої виразом (8).

Як четверту вихідну передумову створюваного алгоритму використаємо взяту з роботи [1] систему проміжних параметрів А, В, С, Д, які визначаються з системи рівнянь (4), та вирази для яких мають вигляд

$$\{q_0 - g_1 q_1 - g_2 q_2 - g_3 q_3 = \text{А}\}; \tag{10}$$

$$\{q_1 - g_1 q_0 - g_2 q_1 - g_3 q_2 = \text{В}\}; \tag{11}$$

$$\{q_2 - g_1 q_1 - g_2 q_0 - g_3 q_1 = \text{С}\}; \tag{12}$$

$$\{q_3 - g_1 q_2 - g_2 q_1 - g_3 q_0 = \text{Д}\}. \tag{13}$$

Цих чотирьох вихідних передумов уже достатньо для деталізації алгоритму узагальненого методу ідентифікації моделі АРКС(3,3), який включатиме в себе такі етапи:

1 етап — для експериментально визначеного відрізку часового ряду l_t за виразом (3) визначаємо його середнє значення b;

2 етап — використовуючи вираз (2), центруємо експериментально визначений відрізок часового ряду, трансформуючи його у центрований часовий ряд m_t;

3 етап — використовуючи вираз (6), обчислюємо дисперсію q_0 та перші шість автоковаріацій q_i, $i = 0, 1, 2, 3, 4, 5, 6$ центрованого часового ряду m_t;

4 етап — підставляючи у вирази (8) числові значення обчислених на попередньому етапі відповідних автоковаріацій, формуємо матриці **M** та **q**;

5 етап — обчислюємо матрицю $\mathbf{M}^{-1}$, обернену до матриці **M**;

6 етап — використовуючи вираз (9), розв'язуємо матричне рівняння та знаходимо числові значення параметрів g_1, g_2, g_3 авторегресійної складової моделі АРКС(3,3);

7 етап — підставляючи у вирази (10)—(13) числові значення параметрів авторегресійної складової моделі АРКС(3,3) та відповідних автоковаріацій, знаходимо числові значення проміжних параметрів А, В, С, Д;

8 етап — з четвертого рівняння системи (4) після підстановки в нього виразу (13) знаходимо, що

$$\sigma_a^2 = -\frac{\text{Д}}{p_3}, \tag{14}$$

з третього рівняння цієї ж системи (4) після підстановки в нього виразів (12) та (14) знаходимо, що

$$p_3 = \frac{\text{Д}p_2}{\text{С} - \text{Д}(g_1 - p_1)}. \tag{15}$$

А підставляючи вирази (14) та (15) у перші два рівняння системи (4), отримуємо нову систему рівнянь

$$\begin{cases} f_1(p_1, p_2) = 0, \\ f_2(p_1, p_2) = 0, \end{cases} \tag{16}$$

в яких

$$\begin{aligned} f_1(p_1, p_2) = {} & Ap_2\big(C - Д(g_1 - p_1)\big) - \big(C - Д(g_1 - p_1)\big)^2\big(-1 + p_1(g_1 - p_1) + p_2(g_2 - p_2)\big) - \\ & - \text{Д}p_2\big(g_3\big(\text{С} - \text{Д}(g_1 - p_1)\big) - \text{Д}p_2\big); \end{aligned} \tag{17}$$

$$f_2(p_1, p_2) = \text{В}p_2 - \big(\text{С} - \text{Д}(g_1 - p_1)\big)\big(p_1 + p_2(g_1 - p_1)\big) - \text{Д}p_2(g_2 - p_2); \tag{18}$$

9 етап — розв'язуючи систему нелінійних рівнянь (16), знаходимо числові значення параметрів p_1, p_2 складової ковзного середнього в моделі АРКС(3,3);

10 етап — з виразу (15) знаходимо числове значення параметра p_3 складової ковзного середнього в моделі АРКС(3,3);

11 етап — з виразу (14) знаходимо числове значення дисперсії σ_a^2 білого шуму, на основі середньоквадратичного значення із якої формується діапазон генерації імпульсів білого шуму, які на кожному кроці прогнозування підмішуються до складової ковзного середнього в моделі АРКС(3,3). Але, якщо в процесі визначення параметрів p_1, p_2, p_3 виявиться, що $p_3 = 0$, то дисперсію білого шу-

му слід визначати з першого рівняння системи (4) з урахування виразу (10), тобто використовуючи вираз

$$\sigma_a^2 = \frac{A}{1 - p_1(g_1 - p_1) - p_2(g_2 - p_2) - p_3(g_3 - p_3)}. \quad (19)$$

Програма Python-реалізації алгоритму узагальненого методу ідентифікації моделі АРКС(3,3)

```
In [1]: import numpy as np                  # Виклик ППП numpy як np
In [2]: L=[5.,8.,3.,4.,6.,3.,2.,7.,5.,4.,3.,6.,4.,5.,   # Внесення списку з N
3.,8.,6.,4.,3.,5.,4.,2.,7.,4.,5.,3.,6.,3.,4.,5.]       ….значень часового ряду
In [3]: N=30                                # Фіксація N=30
In [4]: def fun (x):                        # Формування функції, яка
            return np.sum(x)                ….визначає суму членів
                                            ….числового масиву x в numpy
In [5]: fun(L)                              # Використання сформованої
Out[5]: 137.0                               ….функції для визначення суми
                                            ….членів часового ряду,
                                            ….заданого списком L
In [6]: b=_/N;b                             # Обчислення середнього
Out[6]: 4.566666666666666                   …значення b часового ряду
In [7]: b=b.round(3);b                      ….та залишення в ньому лише
Out[7]: 4.567                               …трьох знаків після коми
In [8]: L1=L-b                              # Центрування часового ряду
                                            ….шляхом віднімання від
                                            ….кожного його члена
                                              середнього значення
                                            ….цього ряду
In [9]: def fun (x):                        # Формування функції, яка
            return np.dot(x,x)              ….визначає скалярний добуток
                                            ….масиву x з самим собою
                                            …. в numpy
In [10]: fun(L1)                            # Використання сформованої
Out[10]:                                    ….функції для визначення
77.36667                                    ….скалярного добутку
                                            ….центрованого часового ряду,
                                            ….заданого списком L1, з самим
                                            ….собою
In [11]: q0=_/N; q0                         # Обчислення дисперсії q0
Out[11]:                                    ….часового ряду та залишення
 2.5788889999999998                         ….в ній лише трьох знаків
In [12]: q0=q0.round(3); q0                 ….після коми
Out[12]:
 2.579
In [13]: L2=L1[:-1]                         # Вилучення зі списку L1
                                            ….центрованого часового ряду
                                            ….останнього члена та
                                            ….трансформація його у
                                            ….список L2
In [14]: L5=L1[1:]                          # Вилучення зі списку L1
                                            ….центрованого часового ряду
                                            ….першого члена та
                                            ….трансформація його у
```

```
                                                    ….список L5
In [15]: def fun (x,y):                             # Формування функції, яка
             return np.dot(x,y)                     ….визначає скалярний добуток
                                                    ….масиву x з масивом y
In [16]: fun(L2,L5)                                 # Використання сформованої
Out[16]:                                            ….функції для визначення
 -22.820818999999993                                ….скалярного добутку
                                                    ….центрованого часового ряду,
                                                    ….укороченого зліва і справа
                                                    ….на одну одиницю у формі
                                                    ….списків L2 та L5
In [17]: q1=_/(N-1); q1                             # Обчислення автоковаріації q1
Out[17]:                                            ….часового ряду та залишення
 -0.786924793103448                                 ….в ній лише трьох знаків
In [18]: q1=q1.round(3);q1                          ….після коми
Out[18]:
 -0.787
In [19]: L3=L2[:-1]                                 # Вилучення зі списку L2
                                                    ….укороченого на одиницю
                                                    ….справа часового ряду
                                                    ….останнього члена та
                                                    ….трансформація його у
                                                    ….список L3
In [20]: L6=L5[1:]                                  # Вилучення зі списку L5
                                                    ….укороченого на одиницю
                                                    ….зліва часового ряду
                                                    ….першого члена та
                                                    ….трансформація його у
                                                    ….список L6
In [21]: fun(L3,L6)                                 # Використання сформованої
Out[21]:                                            ….функції для визначення
-14.874308000000001                                 ….скалярного добутку
                                                    ….центрованого часового ряду,
                                                    ….укороченого зліва і справа
                                                    ….на дві одиниці у формі
                                                    ….списків L3 та L6
In [22]: q2=_/(N-2);q2                              # Обчислення автоковаріації q2
Out[22]:                                            ….часового ряду та залишення
 -0.5312252857142857                                ….в ній лише трьох знаків
In [23]: q2=q2.round(3);q2                          ….після коми
Out[23]:
-0.531
In [24]: L4=L3[:-1]                                 # Вилучення зі списку L3
                                                    ….укороченого на дві одиниці
                                                    ….справа часового ряду
                                                    ….останнього члена та
                                                    ….трансформація його у
                                                    ….список L4
In [25]: L7=L6[1:]                                  # Вилучення зі списку L6
                                                    ….укороченого на дві одиниці
                                                    ….зліва часового ряду
                                                    ….першого члена та
```

```
                                                ….трансформація його у
                                                ….список L7
In [26]: fun(L4,L7)                             # Використання сформованої
Out[26]:                                        ….функції для визначення
2.6702030000000008                              ….скалярного добутку
                                                ….центрованого часового ряду,
                                                ….укороченого зліва і справа
                                                ….на три одиниці у формі
                                                ….списків L4 та L7
In [27]: q3=_/(N-3); q3                         # Обчислення автоковаріації q3
Out[27]:                                        ….часового ряду та залишення
0.09889640740740743                             ….в ній лише трьох знаків
In [28]: q3=q3.round(3); q3                     ….після коми
Out[28]:
0.099
In [29]: L41=L4[:-1]                            # Вилучення зі списку L4
                                                ….укороченого на три одиниці
                                                ….справа часового ряду
                                                ….останнього члена та
                                                ….трансформація його у
                                                ….список L41
In [30]: L71=L7[1:]                             # Вилучення зі списку L7
                                                ….укороченого на три одиниці
                                                ….зліва часового ряду
                                                ….першого члена та
                                                ….трансформація його у
                                                ….список L71
In [31]: fun(L41,L71)                           # Використання сформованої
Out[31]:                                        ….функції для визначення
20.482714                                       ….скалярного добутку
                                                ….центрованого часового ряду,
                                                ….укороченого зліва і справа
                                                ….на чотири одиниці у формі
                                                ….списків L41 та L71
In [32]: q4=_/(N-4); q4                         # Обчислення автоковаріації q4
Out[32]:                                        ….часового ряду та залишення
0.7877966923076923                              ….в ній лише трьох знаків
In [33]: q4=q4.round(3); q4                     ….після коми
Out[33]:
0.788
In [34]: L31=L41[:-1]                           # Вилучення зі списку L41
                                                ….укороченого на чотири
                                                ….одиниці справа часового
                                                ….ряду останнього члена
                                                ….та трансформація його у
                                                ….список L31
In [35]: L61=L71[1:]                            # Вилучення зі списку L71
                                                ….укороченого на чотири
                                                ….одиниці зліва часового
                                                ….ряду першого члена
                                                ….та трансформація його у
                                                ….список L61
```

```
In [36]: fun(L31,L61)                    # Використання сформованої
Out[36]:                                 ….функції для визначення
-32.271775                               ….скалярного добутку
                                         ….центрованого часового ряду,
                                         ….укороченого зліва і справа
                                         ….на п’ять одиниць у формі
                                         ….списків L31 та L61
In [37]: q5=_/(N-5); q5                  # Обчислення автоковаріації q5
Out[37]:                                 ….часового ряду та залишення
-1.2908709999999999                      ….в ній лише трьох знаків
In [38]: q5=q5.round(3); q5              ….після коми
Out[38]:
-1.291
In [39]: L21=L31[:-1]                    # Вилучення зі списку L31
                                         ….укороченого на п’ять
                                         ….одиниць справа часового
                                         ….ряду останнього члена
                                         ….та трансформація його у
                                         ….список L21
In [40]: L51=L61[1:]                     # Вилучення зі списку L61
                                         ….укороченого на п’ять
                                         ….одиниць зліва часового
                                         ….ряду першого члена
                                         ….та трансформація його у
                                         ….список L51
In [41]: fun(L21,L51)                    # Використання сформованої
Out[41]:                                 ….функції для визначення
-3.593264000000005                       ….скалярного добутку
                                         ….центрованого часового ряду,
                                         ….укороченого зліва і справа
                                         ….на шість одиниць у формі
                                         ….списків L21 та L51
In [42]: q6=_/(N-6); q6                  # Обчислення автоковаріації q6
Out[42]:                                 ….часового ряду та залишення
-0.14971933333333354                     ….в ній лише трьох знаків
In [43]: q6=q6.round(3); q6              ….після коми
Out[43]:
-0.15
In [44]: L9=[q0,q1,q2,q3,q4,q5,q6];L9    # Формування списку L9
Out[44]:                                 ….значень автоковаріацій
[2.579, -0.787, -0.531, 0.099, 0.788,
 -1.291, -0.15]
In [45]: import sympy                    # Виклик ППП sympy
In [46]: from sympy import*              ….та усіх його функцій
In [47]: q,q0,q1,q2,q3,q4,q5,q6 =\
 symbols('q q0 q1 q2 q3 q4 q5 q6')       # Символізація автокореляцій
In [48]: M = symbols('M')                # Символізація матриці системи
In [49]: M = Matrix([[q3,q2,q1],[q4,q3,q2],  ….рівнянь для визначення
[q5,q4,q3]]);M                           ….параметрів авторегресії та
Out[49]:                                 ….її представлення у загальному
Matrix([                                 ….вигляді
[q3, q2, q1],
```

```
[q4, q3, q2],
[q5, q4, q3]])
In [50]: g,g1,g2,g3=symbols('g g1 g2 g3')          # Символізація параметрів
In [51]: g = Matrix([g1,g2,g3]);g                  ….авторегресії та форматування
Out[51]:                                           ….матриці-стовпця з них у
Matrix([                                           ….загальному вигляді
[g1],
[g2],
[g3]])
In [52]: M =M.subs([(q1,L9[1]),(q2,L9[2]),         # Ідентифікація системної
(q3,L9[3]),(q4,L9[4]),(q5,L9[5])]);M               ….матриці M
Out[52]:
Matrix([
[ 0.099, -0.531, -0.787],
[ 0.788,  0.099, -0.531],
[-1.291,  0.788,  0.099]])
In [53]: q=Matrix([q4,q5,q6]); q                   # Форматування матриці-
Out[53]:                                           ….стовпця автоковаріацій в
Matrix([                                           ….загальному вигляді
[q4],
[q5],
[q6]])
In [54]: q=q.subs([(q4,L9[4]),(q5,L9[5]),          # Ідентифікація матриці
(q6,L9[6])]); q                                    ….автоковаріацій шляхом
Out[54]:                                           ….підставлення в неї
Matrix([                                           ….їх числових значень
[ 0.788],
[-1.291],
[ -0.15]])
In [55]: M1=simplify(M.inv());M1                   # Обчислення матриці
Out[55]:                                           ….оберненої до системної
Matrix([
[-0.492522254750502,   0.65280312404595,
 -0.413904602224703],[-0.698718915489662,
   1.15728680936142,   0.65280312404595],
[-0.861168944212565, -0.698718915489662,
 -0.492522254750502]])
In [56]: g=M1*q; g                                 # Розв’язання системи рівнянь
Out[56]:                                           ….M g=q для визначення
[-1.16879067955301],                               ….параметрів авторегресійної
[-2.14256824489834],                               ….складової моделі АРКС(3,3)
[0.297323330070228]])
In [57]: g=g.evalf(3); g                           # Скорочення до трьох знаків
Out[57]:                                           ….після коми в числових
Matrix([                                           ….значеннях параметрів
[-1.17],
[-2.14],
[0.297]])
In [58]: g1=g[0,0]; g1                             # Ідентифікація параметра g1
Out[58]:
-1.17
In [59]: g2=g[1,0]; g2                             # Ідентифікація параметра g2
```

```
Out[59]:
-2.14
In [60]: g3=g[2,0]; g3                          # Ідентифікація параметра g3
Out[60]:
0.297
In [61]: A,B,C,D = symbols ('A B C D')          # Оголошення символьними
                                                ….A,B,C,D
In [62]: p,p0,p1,p2,p3 = symbols (‘p p0 p1 p2 p3’)  # Оголошення символьними
                                                ….p,p0,p1,p2,p3
In [63]: A=q0-g1*q1-g2*q2-g3*q3;A               # Перший етап обчислення
Out[63]:                                        ….виразу A
q0 + 1.17*q1 + 2.14*q2 - 0.297*q3
In [64]: A = A.subs([(q0,L9[0]),(q1,L9[1]),     # Другий етап обчислення
(q2,L9[2]),(q3,L9[3])]);A                       ….виразу A
Out[64]:
0.492090270996094
In [65]: A=A.evalf(3);A                         # Обмеження до трьох числа
Out[65]:                                        ….знаків після коми у
0.492                                           ….значенні виразу A
In [66]: B =q1-g1*q0-g2*q1-g3*q2;B              # Перший етап обчислення
Out[66]:                                        ….виразу B
1.17*q0 + 3.14*q1 - 0.297*q2
In [67]: B = B.subs([(q0,L9[0]),(q1,L9[1]),     # Другий етап обчислення
(q2,L9[2])]);B                                  ….виразу B
Out[67]:
0.698738830566406
In [68]: B=B.evalf(3);B                         # Обмеження до трьох числа
Out[68]:                                        ….знаків після коми у
0.699                                           ….значенні виразу B
In [69]: C = q2-g1*q1-g2*q0-g3*q1;C             # Перший етап обчислення
Out[69]:                                        ….виразу C
2.14*q0 + 0.871*q1 + q2
In [70]: C = C.subs([(q0,L9[0]),(q1,L9[1]),     # Другий етап обчислення
(q2,L9[2])]);C                                  ….виразу C
Out[70]:
4.30896606445313
In [71]: C=C.evalf(3);C                         # Обмеження до трьох числа
Out[71]:                                        ….знаків у
4.31                                            ….значенні виразу C
In [72]: D = q3-g1*q2-g2*q1-g3*q0;D             # Перший етап обчислення
Out[72]:                                        ….виразу D
-0.297*q0 + 2.14*q1 + 1.17*q2 + q3
In [73]: D = D.subs([(q0,L9[0]),(q1,L9[1]),     # Другий етап обчислення
(q2,L9[2]),(q3,L9[3])]);D                       ….виразу D
Out[73]:
-2.97453179931641
In [74]: D=D.evalf(3);D                         # Обмеження до трьох числа
Out[74]:                                        ….знаків у
-2.97                                           ….значенні виразу D
In [75]: p,p1,p2,p3 = symbols ('p p1 p2 p3')    # Оголошення символьними
                                                ….p,p1,p2,p3
In [76]: f1 = Function ('f1')(p1,p2)            # Символізація функції f1
```

```
In [77]: f2 = Function ('f2')(p1,p2)                    # Символізація функції f2
In [78]: f1=-A*p2/(C-D*(g1-p1))-1+p1*(g1-p1)+\          # Внесення та ідентифікація
p2*(g2-p2)+(D*p2/(C-D*(g1-p1)))*\                       ….функції f1
(g3-D*p2/(C-D*(g1-p1)));f1
Out[78]:
p1*(-p1 - 1.17) + p2*(-p2 - 2.14) + \
0.492*p2/(2.97*p1 - 0.832) - 2.97*p2*\
(2.97*p2/(0.832 - 2.97*p1) +\
 0.297)/(0.832 - 2.97*p1) - 1
In [79]: f2=expand(f1);f2                               # Спрощення функції f1
Out[79]:                                                ….до вигляду f2
(0.492*p2+(1+p1*(1.17+p1)+p2*(2.14+\
p2))*(4.31-2.97*(1.17+p1)))*(4.31-2.97*(1.17+p1))+\
2.97*p2*(0.297*(4.31-2.97*(1.17+p1))+2.97*p2)
In [80]: f11=Function('f11')(p1,p2)                     # Символізація функції f11
In [81]: f22=Function('f22')(p1,p2)                     # Символізація функції f22
In [82]: f11=-B*p2/(C-D*(g1-p1))+p1+p2*\                # Внесення та ідентифікація
(g1-p1)+D*p2*(g2-p2)/(C-D*(g1-p1));f11                  ….функції f11
Out[82]:
p1 + p2*(-p1 - 1.17) + 0.699*p2/(2.97*p1 -\
 0.832) - 2.97*p2*(-p2 - 2.14)/(0.832 - 2.97*p1)
In [83]: f22=expand(f11);f22                            # Спрощення функції f11
Out[83]:                                                ….до вигляду f22
0.699*p2-(p1-p2*(1.17+p1))*(4.31-2.97*(1.17+p1))-\
2.97*p2*(2.14+p2)
In [84]: L15= solve([(0.492*p2+(1+p1*(1.17+p1)+p2*(2.14+\ # Розв’язання системи рівнянь
p2))*(4.31-2.97*(1.17+p1)))*(4.31-2.97*(1.17+p1))+\     ….f2(p1,p2)=0, f22(p1,p2)=0 та
2.97*p2*(0.297*(4.31-2.97*(1.17+p1))+2.97*p2),0.699*p2-\ ….отримання коренів у вигляді
(p1-p2*(1.17+p1))*(4.31-2.97*(1.17+p1))-\               ….списку L15
2.97*p2*(2.14+p2)],p1,p2);L15
Out[84]:
[(0.281178451178451, 0.0)]
In [85]: p1=L15[0][0];p1                                # Визначення параметра p1 та
Out[85]:                                                ….та залишення в його
0.280193236714976                                       ….числовому значенні
In [86]: p1=p1.evalf(3);p1                              ….лише трьох знаків після коми
Out[86]:
0.280
In [87]: p2=L15[0][1];p2                                # Визначення параметра p2
Out[87]:
0.0
In [88]: p3=D*p2/(C-D*(g1-p1));p3                       # Визначення параметра p3

Out[88]:
0.0
In [89]: p0=1
In [90]: ska,skv = symbols('ska skv')                   # Символізація дисперсії і СКВ
                                                        ….білого шуму
In [91]: ska=A/(1-p1*(g1-p1));ska                       # Обчислення дисперсії
Out[91]:                                                ….білого шуму
0.350
In [92]: skv=ska**(0.5);skv                             # Обчислення СКВ
```

```
Out[92]:
0.591587344301330
In [93]: skv=skv.evalf(3);skv
Out[93]:
0.592
In [94]: a11,a22 = symbols('a11 a22')
In [95]: a11=-2*skv;a11
Out[95]:
-1.18
In [96]: a22=2*skv;a22
Out[96]:
1.18
In [97]: p=[p0,p1];p
Out[97]:
[1.0, 0.280]
In [98]: import random as rnd

In [99]: m = symbols('m:5');m
Out[99]:
(m0, m1, m2, m3, m4)
In [100]: l = symbols ('l:5');l
Out[100]:
 (l0, l1, l2, l3, l4)
In [101]: m=list(m);m
Out[101]:
 [m0, m1, m2, m3, m4]
In [102]: l=list(l);l
Out[102]:
 [l0, l1, l2, l3, l4]
In [103]: d = symbols ('d:5'); d
Out[103]:
 (d0, d1, d2, d3, d4)
In [104]: d = list (d); d
Out[104]:
[d0,d1,d2,d3,d4]
In [105]: d0 = rnd.uniform (-1.18, 1.18); d0
Out[105]:
-0.9786481619579281
In [106]: m0= g1*L1[29]+g2*L1[28]+\
g3*L1[27]+ d0; m0
Out[106]:
-0.735726592133709
In [107]: L1 = np.append(L1,[m0])

In [108]: l0=b+m0; l0
Out[108]:
3.83127340786629
In [109]: L = np.append(L,[l0])

In [110]: d1 = rnd.uniform (-1.18, 1.18); d1
Out[110]:
-0.9602434653342734
```

….білого шуму
….та залишення в його
….числовому значенні
….лише трьох знаків після коми

Символізація та обчислення
….границь ***a11,a22*** діапазону
…. генерації імпульсів
….«нормального» білого шуму
…. з довірчою ймовірністю ***0,95***
….для «підмішування» в модель
….***АРКС(3,1)***
Формування списку параметрів
….складової ковзного
….середнього в моделі ***АРКС(3,1)***
Виклик модуля генерації
….імпульсів білого шуму
Індексна символізація
….параметра ***m***

Індексна символізація
….параметра ***l***

Створення списку для
….параметра ***m***

Створення списку для
….параметра ***l***

Індексна символізація
….параметра ***d***

Створення списку для
….параметра ***d***

Генерація випадкового
….імпульсу ***d0*** для першої
….ітерації прогнозу
Обчислення ***N+1*** -го члена
….списку ***L1*** з використанням
….моделі ***АРКС(3,1)***

Додавання в кінець списку
….***L1*** нового члена ***m0***
Обчислення ***N+1*** -го члена
….списку ***L***

Додавання в кінець списку
….***L*** нового члена ***l0***
Генерація випадкового
….імпульсу ***d1*** для другої
….ітерації прогнозу

```
In [111]: m1= g1*L1[30]+g2*L1[29]+\
g3*L1[28]+ d1-p1*d0; m1
Out[111]:
-0.922476284076637
In [112]: L1 = np.append(L1,[m1])

In [113]: l1=b+m1; l1
Out[113]:
3.64452371592336
In [114]: L=np.append(L,[l1])

In [115]: d2 = rnd.uniform (-1.18, 1.18); d2
Out[115]:
0.1282150195429812
In [116]: m2= g1*L1[31]+g2*L1[30]+\
g3*L1[29]+ d2-p1*d1; m2
Out[116]:
3.18046983687152
In [117]: L1 = np.append(L1,[m2])

In [118]: l2=b+m2; l2
Out[118]:
7.74746983687152
In [119]: L=np.append(L,[l2])

In [120]: d3 = rnd.uniform(-1.18, 1.18); d3
Out[120]:
-1.133835327586167
In [121]: m3= g1*L1[32]+g2*L1[31]+\
g3*L1[30]+ d3-p1*d2; m3
Out[121]:
-3.12903725295206
In [122]: L1 = np.append(L1,[m3])

In [123]: l3=b+m3; l3
Out[123]:
1.43796274704794
In [124]: L=np.append(L,[l3])

In [125]: d4 = rnd.uniform(-1.18, 1.18); d4
Out[125]:
0.13261500744833254
In [126]: m4= g1*L1[32]+g2*L1[31]+\
g3*L1[30]+ d4-p1*d3; m4
Out[126]:
-1.50894475826974
In [127]: L1 = np.append(L1,[m4])

In [128]: l4=b+m4; l4
Out[128]:
3.05805524173026
In [129]: L=np.append(L,[l4])
```

Обчислення ***N+2*** -го члена
….списку ***L1*** з використанням
….моделі ***АРКС(3,1)***

Додавання в кінець списку
….***L1*** нового члена ***m1***
Обчислення ***N+2*** -го члена
….списку ***L***

Додавання в кінець списку
….***L*** нового члена ***l1***
Генерація випадкового
….імпульсу ***d2*** для третьої
….ітерації прогнозу
Обчислення ***N+3*** -го члена
….списку ***L1*** з використанням
….моделі ***АРКС(3,1)***

Додавання в кінець списку
….***L1*** нового члена ***m2***
Обчислення ***N+3*** -го члена
….списку ***L***

Додавання в кінець списку
….***L*** нового члена ***l2***
Генерація випадкового
….імпульсу ***d3*** для четвертої
….ітерації прогнозу
Обчислення ***N+4*** -го члена
….списку ***L1*** з використанням
….моделі ***АРКС(3,1)***

Додавання в кінець списку
….***L1*** нового члена ***m3***
Обчислення ***N+4*** -го члена
….списку ***L***

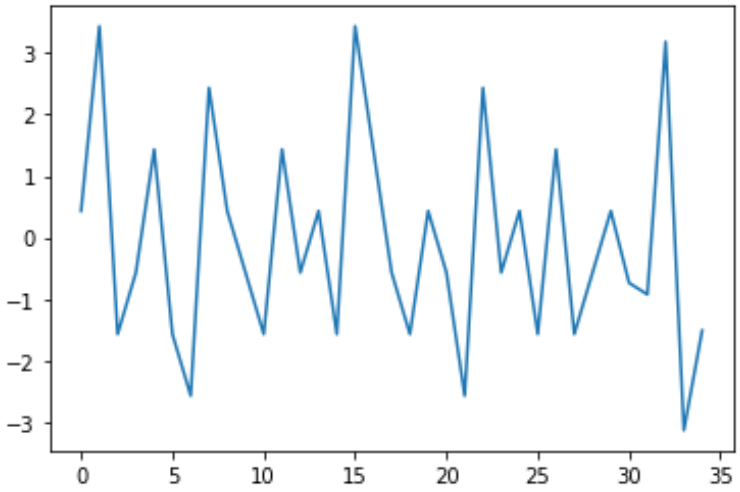
Додавання в кінець списку
….***L*** нового члена ***l3***
Генерація випадкового
….імпульсу ***d4*** для п’ятої
….ітерації прогнозу
Обчислення ***N+5*** -го члена
….списку ***L1*** з використанням
….моделі ***АРКС(3,1)***

Додавання в кінець списку
….***L1*** нового члена ***m4***
Обчислення ***N+5*** -го члена
….списку ***L***

Додавання в кінець списку

```
                                        ….L нового члена l4
In [130]: import matplotlib             # Виклик ППП matplotlib
In [131]: import matplotlib.pyplot as plt  # Виклик модуля pyplot як plt
In [132]: plt.plot(L1)                  # Побудова графіка часового
Out[132]: [<matplotlib.                 ….ряду з прогнозними членами
lines.Line2D at 0x2066cb6ce50>]         ….включно
```

***Кінець програми Python-реалізації алгоритму узагальненого методу ідентифікації моделі* АРКС(3,3).**



Графік часового ряду m_t, який в діапазоні $t \in [0,30]$ заповнений експериментально отриманими значеннями, а за межами цього діапазону заповнений прогнозними значеннями, отриманими з використанням моделі АРКС (3,1), ідентифікованої з використанням експериментально отриманих значень

Попередня оцінка отриманих результатів

Здійснивши ідентифікацію математичної моделі певного об'єкта, потрібно у першу чергу після цього оцінити адекватність отриманої моделі, але для цього потрібно було б спочатку здійснити прогнозування подальших значень часового ряду, реалізованих як з використанням нашої математичної моделі, так і з використанням інших математичних моделей цього класу, ідентифікованих класичними методами, що збільшило б обсяг цієї статті, як мінімум вдвічі. Тому у цій статті ми питанням оцінки адекватності математичної моделі, синтезованої за нашим методом, займатись не будемо, а лише в нашій наступній статті. А у цій статті звернемо увагу на три аспекти, які певною мірою теж можна віднести до питання оцінки адекватності синтезованої математичної моделі за нашим методом.

І як перший з цих аспектів, який працює на підвищення ступеня адекватності моделі, синтезованої за нашим методом, повторимо уже висловлене нами раніше у роботі [1] твердження про те, що на відміну від алгоритму ідентифікації моделі АРКС$\left(n_{ap}, n_{kc}\right)$ за методикою, викладеною в роботах [2], [3], де і параметри авторегресії і параметри ковзного середнього пропонується визначати, використовуючи ті самі значення автоковаріацій та застосовуючи процедуру мінімізації суми квадратів відхилень для пошуку оцінок параметрів ковзного середнього за визначених попередньо на тих же значеннях автоковаріацій за методикою Юла–Уокера параметрах авторегресії, у запропонованому нами удосконаленому методі, по-перше, параметри авторегресії розраховуються з використанням одного набору автоковаріацій, а параметри ковзного середнього розраховуються з використанням іншого набору автоковаріацій, що відповідає кібернетичному принципу використання «свіжих точок» при розширенні набору параметрів, оцінки яких знаходяться. По-друге, для визначення параметрів ковзного середнього застосовується пряма процедура, яка не вимагає поновлення процедури мінімізації суми квадратів відхилень при переході до інших значень порядків авторегресії та ковзного середнього.

Що ж до другого аспекту, то він грає роль попереджувального, оскільки акцентує увагу на тому, що для розрахунку параметрів авторегресійної складової моделі АРКС(3,3) у базовому класичному ме-

тоді ідентифікації Юла–Уокера використовуються перша, друга та третя автоковаріації часового ряду, що розраховуються з використанням експериментально визначеної реалізації цього часового ряду певної довжини, а у запропонованому методі для розрахунку цих же параметрів використовуються четверта, п'ята та шоста автоковаріації часового ряду, для отримання оцінок яких з похибкою, що не перевищуватиме похибку розрахунку перших трьох автоковаріацій, потрібно мати експериментально визначену реалізацію часового ряду вдвоє довшу, оскільки в розрахунку автоковаріації необхідно скорочувати цю реалізацію з обох боків на кількість членів, що дорівнює порядку автоковаріації. До того ж, слід пам'ятати один з основних постулатів математичної статистики, що з прийнятною для практичних розрахунків точністю обчислюються лише автоковаріації на відрізку значень незалежної змінної, що не перевищує 10 % від довжини реалізації, а похибка обчислення оцінок автоковаріацій за межами цього 10-процентного відрізку неухильно і стрімко зростає. Отже, на вищу точність прогнозування наступних значень часового ряду за використання моделі АРКС(3,3), ідентифікованої за нашим методом, можна розраховувати лише за використання в процесі ідентифікації цієї моделі експериментальної реалізації часового ряду, що містить в собі не менше 60 його значень.

Третій же аспект стосуватиметься того, що наш метод автоматично скорочує кількість параметрів ковзного середнього в разі, якщо ми планували їх використати в кількості, більшій від доцільної. Про це свідчить той факт, що в алгоритм закладено обчислення трьох параметрів складової ковзного середнього, а в результаті розрахунків отримано по факту, що для авторегресійного моделювання заданого часового ряду виявилось достатньо лише одного параметра для цієї складової моделі, оскільки інші в обчисленні отримали нульові значення. Тобто стартував алгоритм з процедури визначення параметрів моделі АРКС(3,3), а на фініші ми отримали модель АРКС(3,1).

Висновки

1. Запропоновано та деталізовано 11-етапний алгоритм реалізації методу ідентифікації математичної моделі класу АРКС(3,3), розробленого цими ж авторами в їх роботах, опублікованих раніше.

2. Створено програму Python-реалізації запропонованого алгоритму ідентифікації та продемонстровано її ефективність у розв'язанні задачі ідентифікації математичної моделі класу АРКС(3,3) для конкретного часового ряду, заданого його експериментальною реалізацією.

3. Визначено умови, яким повинна відповідати експериментальна реалізація часового ряду, за використання яких здійснюється авторська ідентифікація математичної моделі цього часового ряду, щоб прогнозування його наступних значень здійснювалось точніше, ніж з використанням математичних моделей цього ж класу, ідентифікація яких здійснювалась традиційними методами.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] О. Б. Мокін, В. Б. Мокін, і Б. І. Мокін, «Метод ідентифікації моделі авторегресії – ковзного середнього АРКС(p,q) з довільними значеннями порядків *p, q*, який узагальнює методику Юла–Уокера,» *Наукові праці Вінницького національного технічного університету*, № 2, с. 1-6, 2014. [Електронний ресурс]. Режим доступу: http://praci.vntu.edu.ua/article/view/3626/5339 .

[2] Дж. Бокс, і Г. Дженкинс, «Анализ временных рядов,» *Прогноз и управление*, вып. 1. М.: Мир, 1974, 408 с.

[3] Дж. Бокс, і Г. Дженкинс, «Анализ временных рядов,» *Прогноз и управление*, вып. 2. М.: Мир, 1974, 197 с.

[4] О. Б. Мокін, В. Б. Мокін, Б. І. Мокін, і І. О. Чернова, «До питання вибору оптимальної математичної моделі стаціонарного часового ряду,» *Вісник Вінницького політехнічного інституту*, № 4, с. 7-15, 2018.

[5] *Python*. [Електронний ресурс]. Режим доступу: https://www.python.org .

[6] П. Г. Доля, *Введение в научный Python*. Харков: ХНУ им. Каразина, 2016, 265 с.

Рекомендована кафедрою системного аналізу та інформаційних технологій ВНТУ

Стаття надійшла до редакції 15.08.2022

Мокін Олександр Борисович — д-р, техн. наук, професор, професор кафедри системного аналізу та інформаційних технологій, e-mail: abmokin@gmail.com ;

Мокін Віталій Борисович — д-р, техн. наук, професор, завідувач кафедри системного аналізу та інформаційних технологій, e-mail: vbmokin@gmail.com ;

Мокін Борис Іванович — академік НАПН України, д-р, техн. наук, професор, професор кафедри системного аналізу та інформаційних технологій, e-mail: borys.mokin@gmail.com

O. B. Mokin[1]
V. B. Mokin[1]
B. I. Mokin[1]

An Algorithm of Identification Method of Autoregressive – Moving-Average Model, Generalizing the Yule–Walker Method, and its Implementation on Python

[1]Vinnytsia National Technical University

The paper presents a detailed 11-step algorithm for practical implementation of a new identification method of autoregressive–moving-average model ARMA (n_{ap}, n_{kc}) *of prediction of stationary time series with arbitrary values of the orders* n_{ap}, n_{kc}. *The new method, published in the previous authors' works, is a generalization of the well-known Yule–Walker method. The algorithm is implemented under the condition, proven by the authors in the previous publications, that the optimal structure of the ARMA* (n_{ap}, n_{kc}) *model is the ARMA(3,3). A feature of this algorithm is that the parameters of the autoregressive component of the ARMA(3,3) model are determined using the fourth, fifth, and sixth autocovariances, which significantly distinguishes it from the traditional algorithm for identifying this class of models using the Yule–Walker method, which uses only autocovariances of the first, second, and third orders. Another feature of the algorithm is a straightforward procedure of determining the parameters of the moving average that does not require renewing the minimizing the residual sum of squares procedure when moving to other orders of the autoregressive and the moving average components, unlike the traditional approaches. The article presents a Python program implementation of the proposed identification algorithm and a demonstration of its effectiveness in solving the problem of identifying the ARMA(3,3) model for a specific time series given by an experimental implementation. The paper also determines the conditions for the experimental implementation of the time series to provide more accurate forecasting compared to the traditional approach.*

Keywords: a stationary time series, autoregressive – moving-average model, ARMA, model identification, generalization of the Yule–Walker method, Python implementation of the model identification algorithm.

Mokin Oleksandr B. — Dr. Sc. (Eng.), Professor, Professor of the Chair of Systems Analysis and Information Technology, e-mail: abmokin@gmail.com ;

Mokin Vitalii B. — Dr. Sc. (Eng.), Professor, Head of the Chair of Systems Analysis and Information Technology, e-mail: vbmokin@gmail.com ;

Mokin Borys I. — Academician of NAPS of Ukraine, Dr. Sc. (Eng.), Professor of the Chair of Systems Analysis and Information Technology, e-mail: borys.mokin@gmail.com