Міністерство освіти і науки України
Вінницький національний технічний університет

Б. І. Мокін, В. Б. Мокін, О. Б. Мокін

НАВЧАЛЬНИЙ ПОСІБНИК
для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python

Частина 2

Вінниця
ВНТУ
2023

УДК 517.98
М74

Рекомендовано до друку Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України як навчальний посібник для студентів та аспірантів закладів вищої освіти, що спеціалізуються в галузі інформаційних технологій (протокол № 7 від «28» лютого 2023 р.).

Рецензенти:
В. Я. Данилов, д.т.н. проф. (НТУУ «КПІ ім. Сікорського»)
В. І. Клочко, д. пед. н. професор (ВНТУ)
О. С. Макаренко, д. ф-м. н. професор (НТУУ «КПІ ім. Сікорського»)

Мокін, Б. І.

М74 Навчальний посібник для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python. Частина 2 : навчальний посібник / Б. І. Мокін, В. Б. Мокін, О. Б. Мокін. – Вінниця : ВНТУ, 2023. – 139 с.

ISBN 978-966-641-926-5

В навчальному посібнику викладено способи розв'язання задач з функціонального аналізу, адаптованого до прикладних проблем в галузі інформаційних технологій, згідно зі змістом однойменного навчального посібника цих же авторів, а також викладені основи програмування мовою Python і програми реалізації способів розв'язання даного класу задач цією мовою. Частина 2 є продовженням частини І і охоплює задачі з дослідження та використання різних класів операторів.

Навчальний посібник рекомендується для студентів та аспірантів, що спеціалізуються в ІТ-галузі за спеціальностями 124 «Системний аналіз» та 126 «Інформаційні системи та технології»

УДК 517.98

ISBN 978-966-641-926-5

ЗМІСТ

ВСТУП

Оскільки друга частина нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python» є органічним продовженням нашої першої частини цього ж навчального посібника [1], то, з метою їхнього пов'язання між собою, ми почнемо вступ до другої частини з цитування першого абзацу вступу до частини першої, який звучить так [1]: «У 2020 році ми опублікували навчальний посібник «Функціональний аналіз, адаптований до прикладних задач в галузі інформаційних технологій» [2], з використанням якого студенти, що навчаються у ВНТУ за спеціальністю 124 «Системний аналіз», опановують навчальну дисципліну «Функціональний аналіз», яка, згідно з освітньо-професійною програмою цієї спеціальності, входить до переліку обов'язкових.

Уже після першого року використання в навчальному процесі нашого навчального посібника з функціонального аналізу ми зрозуміли, що студентам важко засвоювати практичні навички з розв'язання прикладних задач з цієї навчальної дисципліни, користуючись лише навчальним посібником, в якому викладені її теоретичні основи, особливо ж в умовах реалізації навчального процесу в онлайн-режимі, зумовленому карантинними обмеженнями щодо перебування студентів групами в аудиторіях, викликаними світовою пандемією коронавірусної хвороби. А тому в нашому авторському колективі виникла задумка написати з функціонального аналізу ще один навчальний посібник, в якому викласти практичні способи розв'язання задач з функціонального аналізу, характерних для ІТ-галузі, аби цей другий навчальний посібник доповнив перший практикою застосування теоретичних основ функціонального аналізу, викладених у першому навчальному посібнику, при розв'язанні прикладних задач, характерних для ІТ-галузі. Особливо важливим аспектом цього другого навчального посібника, на наш погляд, мало стати доведення етапу розв'язання прикладних задач функціонального аналізу до застосування сучасної мови програмування Python, усі дистрибутиви якої викладені в інтернеті для вільного доступу у вигляді програмного середовища Anaconda [3]. Саме ця задумка і реалізована нами у навчальному посібнику, зміст якого подається нижче. І оскільки цей навчальний посібник покликаний доповнювати практикою попередній, присвячений теорії функціонального аналізу, то і розділи його збігаються за назвами і змістом з відповідними розділами свого теоретичного попередника.

Характерною особливістю нашого навчального посібника, вміст якого подається нижче, є те, що ми спочатку демонструємо способи і алгоритми «ручного» розв'язання прикладних задач функціонального аналізу на конкретних простих прикладах, а уже потім показуємо, як ці задачі розв'язувати за допомогою мови Python з використанням дистрибутивів і підпрограм, що допускають їх комп'ютерну реалізацію в консолі Spyder.

Такий підхід зумовлений тим, що, як показала практика роботи зі студентами під час вивчення функціонального аналізу, в разі, якщо для розв'язання задачі використати готову програму, написану мовою Python, то студенти, використовуючи цю програму, отримують розв'язок задачі, не розуміючи суті закладених в програму алгоритмів, що, по-перше, надовго не залишається в пам'яті, а по-друге, не сприяє осмисленому застосуванню цих же алгоритмів при розв'язанні інших задач, умови яких в чомусь відрізняються.»

Але, оскільки у цьому навчальному посібнику ми використовуватимемо мову програмування Python, то спочатку, з посиланням на популярний [4] та науковий [5] варіанти її викладення, здійснимо екскурс в історію створення цієї мови та в загальних рисах розглянемо основні дистрибутиви, що входять до її структури, використавши для цього матеріал, уже викладений нами у вступі до першої частини навчального посібника [1].

Цитувати історію створення мови Python ми не будемо, бо ті, кого вона цікавить, можуть ознайомитись з цією історією, прочитавши вступ до першої частини навчального посібника [1], а ось інформацію про основні дистрибутиви цієї мови, аби їх пригадати, ми наведемо знову ж таки зі вступу до першої частини цього навчального посібника [1]:

«…сконцентруємо увагу на 64-бітовому інтегрованому середовищі Anaconda, на яке ви можете зустріти посилання в різних літературних джерелах і як на дистрибутив, і як на пакет прикладних програм (ППП) мовою Python. Встановлювати на своєму комп'ютері дистрибутив Anaconda вигідно ще й тому, що одночасно та автоматично на вашому комп'ютері встановлюється і бібліотека пакетів прикладних програм, які використовуються для наукових досліджень і носять назви: numpy, sympy, sciру, matplotlib.

ППП numpy (або NumPy – числовий Пайтон) – це пакет програм, який використовується для числових розрахунків.

ППП sympy (або SymPy - символьний Пайтон) – це пакет програм, який використовується для всіляких перетворень виразів, заданих у символьній формі (зокрема і для взяття похідних та невизначених інтегралів від складних функцій).

ППП sciру (або SciPy – науковий Пайтон) – це пакет програм, який зручно застосовувати при обробленні результатів наукових досліджень у сукупності з пакетами sympy та numpy.

ППП matplotlib – це пакет програм для одно-, дво- і тривимірних графічних зображень результатів розрахунків, виконаних з застосуванням пакетів numpy, sympy та sciру, тобто, це графічний редактор, пристосований до роботи з програмами, написаними мовою Python.

Зміст усіх цих ППП та порядок їх застосування ми будемо розкривати в процесі їх використання для розв'язання задач функціонального аналізу, а у цій вступній частині навчального посібника згадаємо ще лише про те, що реалізовувати ці ППП в разі, якщо ви встановили на своєму комп'ютері дистрибутив Anaconda, можна за допомогою інтерпретаторів (або, що одне і те ж, консолей) IPython, IPython Notebook, який ще називають Jupyter Notebook, та Spyder, головне вікно останнього з яких на екрані розділене на три частини, причому ліва частина призначена для набору і редагування програм у вигляді файлів, які можна або виконувати або відправляти в пам'ять з можливістю виклику у подальшому, права нижня частина призначена для набору програм за допомогою командного рядка, їх тестування та виведення результатів їх дії і результатів тестування на екран, а права верхня частина призначена для виведення на екран рисунків, що супроводжують обчислення.

Ми в нашому навчальному посібнику будемо використовувати саме консоль Spyder, назва якої є абревіатурою від Scientific PYthon Development EnviRonment, що перекладається як "Наукового Пайтону середовище розвитку», і яка є найбільш узагальненою, оскільки інтегрує в собі основні риси інших двох інтерпретаторів, згаданих вище» – кінець цитати зі вступу до першої частини навчального посібника [1].

На цьому ми у вступі до другої частини нашого навчального посібника поставимо крапку і приступимо до конкретизації її змісту, зберігши ту ж послідовність розділів, що і в базовому навчальному посібнику [2] та розпочавши нумерацію розділів і програм у другій частині у послідовностях, що доповнюють їх до послідовностей, використаних у частині першій [1]. Єдине, в чому ми відступимо від вищевикладеного, це те, що восьмий розділ базового навчального посібника [2] ми у другій частині, до написання якої приступаємо, розіб'ємо на два розділи, у першому з яких, тобто у «Розділі 8», викладемо задачі і програми, пов'язані з неперервними операторами, а у другому з них, тобто у «Розділі 9», викладемо задачі і програми, пов'язані з дискретними операторами.

Розділ 6 ОПЕРАТОРИ ТА ЇХ ОСНОВНІ ХАРАКТЕРИСТИКИ (В ПРИКЛАДАХ І ПРОГРАМАХ)

6.1 Загальні характеристики операторів, резольвента і спектр та група

У кінці однойменного підрозділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. Дайте означення оператора.
2. Який оператор є лінійним?
3. Що собою являє норма оператора?
4. Як визначити обернений оператор?
5. Що є резольвентою неоднорідного операторного рівняння?
6. Які значення оператора є регулярними, а які складають його спектр?
7. За яких умов неоднорідне операторне рівняння має коректний розв'язок?
8. Як визначити власні числа та власні вектори оператора?
9. Що розуміють під групою операторів, і які властивості такої групи ви знаєте?

Тож із відповідей на ці питання ми і розпочнемо викладення змісту другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих питань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2], у вигляді розлогої, але з пропусками та деякими перефразуваннями відфільтрованої цитати. Отже,

Оператор A, який здійснює перетворення множини X у множину Y, ***називається лінійним, якщо:***

$$A\cdot\left(x_1+x_2\right)=A\cdot x_1+A\cdot x_2,\ \forall\, x_1,x_2\in X\,, \tag{6.1}$$

$$A\cdot x_n\to A\cdot x_0,\quad \{x_n\}\subset \mathrm{X},\quad x_0\in \mathrm{X},\quad x_n\to x_0, \tag{6.2}$$

$$A\cdot 0=0,\ 0\in X\,; \tag{6.3}$$

$$A\left(-x\right)=-A\cdot x,\ \forall\, x\in X\,\cdot \tag{6.4}$$

Лінійний оператор, для кого виконується співвідношення

$$A\cdot\left(t\cdot x\right)=t\cdot A\cdot x,\ \forall\, x\in X, t-\text{скаляр}, \tag{6.5}$$

називається ***однорідним***.

Число

$$K_0=\sup_{\|x\|\le 1}\|A\cdot x\|=\|A\|=\sup_{x\neq 0}\frac{\|A\cdot x\|}{\|x\|}. \tag{6.6}$$

називають нормою оператора A ***і позначають***

$$K_0=\|A\|\cdot \tag{6.7}$$

З нерівності трикутника для норми випливає, що для $\forall\, x\in S$ справедливо

$$\|A\cdot x\|\le\|A\|\cdot\|x\|. \tag{6.8}$$

Отже, ***множина лінійних операторів A, які перетворюють множину X у множину*** Y, що символічно позначається як

$$A:\left(X\to Y\right) \tag{6.9}$$

або

$$A\in\left(X\to Y\right), \tag{6.10}$$

є лінійним нормованим простором $\left(X\to Y\right)$, в якому

$$\left\| A\cdot x_n - A\cdot x_0\right\| = \left\| A\cdot\left(x_n - x_0\right)\right\| \le K\cdot\left\| x_n - x_0\right\| = 0, \tag{6.11}$$

оскільки з того, що $x_n \to x_0$, випливає, що

$$\left\| x_n - x_0\right\| = 0. \tag{6.12}$$

Оператор $A^{-1}\in\left(Y\to X\right)$**,** який задовольняє рівняння

$$\begin{cases} A^{-1}\cdot\left(A\cdot x\right) = x, \ \forall\, x\in X, \\ A\cdot\left(A^{-1}\cdot y\right) = y, \ \forall\, y\in Y. \end{cases} \tag{6.13}$$

що матиме місце лише за умови

$$A^{-1}\cdot A = A\cdot A^{-1} = I \tag{6.14}$$

називають ***оберненим до оператора A***.

Зауважимо, що під добутком будь-яких операторів ми розуміємо їх послідовне застосування до елемента, що стоїть з правого боку від них.

Після введення оберненого оператора A^{-1} зрозуміло, що розв'язок рівняння

$$A\cdot x = y, \tag{6.15}$$

де $x\in X$, $y\in Y$, $A\in\left(X\to Y\right)$, матиме вигляд

$$x = A^{-1}y, \tag{6.16}$$

а розв'язок рівняння

$$A\cdot x - \lambda\cdot x = y, \tag{6.17}$$

де λ – скаляр, матиме вигляд

$$x = \left(A - \lambda\cdot I\right)^{-1}\cdot y. \tag{6.18}$$

Оператор R_λ, обернений до $\left(A-\lambda\cdot I\right)$, тобто,

$$R_\lambda = \left(A-\lambda\cdot I\right)^{-1} = \frac{I}{A-\lambda I} = -\left(\frac{1}{\lambda}\,\frac{I}{I-\frac{1}{\lambda}A}\right) = -\frac{1}{\lambda}\left(I+\frac{1}{\lambda}A+\frac{1}{\lambda^2}A^2+\frac{1}{\lambda^3}A^3+\cdots\right) \tag{6.19}$$

називають ***резольвентою оператора A або розв'язувальним оператором для рівняння*** (6.17).

Якщо ж рівняння (6.17) матиме вигляд

$$A\,x - \lambda\,x = 0, \tag{6.20}$$

то розв'язок рівняння (6.20) буде мати вигляд

$$x = \left(A - \lambda \cdot I\right)^{-1} \cdot 0 = 0 \cdot \tag{6.21}$$

Ті значення параметра λ, які дозволяють мати резольвенту R_λ, називають регулярними значеннями оператора A. Всі інші значення параметра λ, які не є регулярними, складають спектр оператора A, а значення λ, за яких однорідне рівняння* (6.20) *має розв'язок, відмінний від нульового, тобто для яких не виконується* (6.21), *називають характеристичними числами або власними значеннями λ_i, $i = 1,2,,,,n$ оператора A з порядком n, які теж є точками спектра цього оператора.

Нагадаємо, що множина ***власних значень λ_i, $i = 1,2,,,,n$ оператора*** A, яка знаходиться з рівняння

$$|A - \lambda\, I| = 0, \tag{6.22}$$

породжує множину власних векторів w_i, $i = 1,2,\dots,n$ цього оператора, яка знаходиться з системи рівнянь

$$(A - \lambda_i\, I)w_i = 0, \;\; i = 1,2,\dots,n\,, \tag{6.23}$$

доповненої рівнянням нормування

$$\sum_{i=1}^{n} w_i = 1 \tag{6.24}$$

Але потрібно пам'ятати, що до складу спектра оператора A можуть входити значення λ, які не є його характеристичними числами, тобто потужність спектра оператора є більшою потужності множини його характеристичних чисел.

Потрібно також пам'ятати, що, якщо оператор A є обмеженим, то весь його спектр лежить у крузі

$$\| \lambda \| \le \| A \|, \tag{6.25}$$

а за межами цього круга, тобто при

$$\| \lambda \| > \| A \|, \tag{6.26}$$

рівняння* (6.17) *має однозначний розв'язок, якщо відповідне йому однорідне рівняння

$$A \cdot x = 0 \tag{6.27}$$

має лише нульовий розв'язок, і має коректний розв'язок, якщо для $\forall\, x \in X$ ***виконується співвідношення***

$$\| x \| \le K \cdot \| A \cdot x \|, \;\; A \in \left(X \to X\right). \tag{6.28}$$

Під групою $u(t)$ операторів A розуміють таку їх сукупність, якій притаманні властивості:

1) $$u(0) = I\,; \tag{6.29}$$

2) $$u(t+\tau) = u(t) \cdot u(\tau), \quad -\infty < t, \tau < \infty; \tag{6.30}$$

3) $$\frac{du}{dt} = A \cdot u\,. \tag{6.31}$$

Зрозуміло, що ці три властивості мають лише ті оператори A, які є обмеженими і задовольняють співвідношення

$$u(t) = e^{t \cdot A} \cdot \tag{6.32}$$

З виразу (6.32) випливає, що, по-перше,

$$A = \frac{\ln u(t)}{t} \cdot \tag{6.33}$$

а по друге, оператор A можна визначити як похідну від групи $u(t)$ при $t = 0$.

Тому ***оператор A називають породним оператором для групи u(t) або, що одне і те саме, інфінітезимальним.***

Розкладаючи експоненту у виразі (6.32) у степеневий ряд, отримаємо

$$u(t) = 1 + t \cdot A + \frac{t^2}{2!} \cdot A^2 + \frac{t^3}{3!} \cdot A^3 + \cdots, \tag{6.34}$$

тобто, група $u(t)$ може бути заданою не лише через експоненту (6.32), а й у вигляді степеневого ряду (6.34).

А завершуючи цей матеріал, ми ще раз звертаємо увагу на те, що за класичним означенням оператор – це закон, за яким одній множині функцій ставиться у відповідність інша множина функцій, про що не потрібно забувати, обчислюючи за виразом (6.6) норму оператора. Тобто, не потрібно забувати, що вираз $||x||$, який стоїть у знаменнику, задає норму функції, наприклад, $x(\theta),\ \theta \in [a, b]$, яка може бути заданою або на множині функцій $C[a, b]$ з нормою у вигляді

$$\big||x|\big| = \max_{\theta \in [a,b]} |x(\theta)|, \tag{6.35}$$

або на множині функцій $L_2[a, b]$ з нормою у вигляді

$$\big||x|\big| = \sqrt{\int_a^b x^2(\theta) d\theta}. \tag{6.36}$$

6.2 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з обчисленням характеристик операторів

Аналізуючи характер формул, викладених у попередньому підрозділі 6.1, бачимо, що для їх реалізації в програмах мовою Python нам будуть потрібні деякі нові знання функцій і методів цієї мови програмування, додаткові до тих, про які уже йшла мова в підрозділах, присвячених викладенню додаткових відомостей з технології програмування мовою Python у першій частині цього навчального посібника, які ми дублювати не будемо.

Отже, для програмної реалізації мовою Python виразів (6.6),(6.14),(6.19), (6.22), (6.23), (6.24),(6.34) до тих знань, що ми уже маємо з цієї мови, у першу чергу потрібно додати інформацію про те, як реалізувати операції лінійної алгебри, які зосереджені в ППП numpy в модулі **numpy.linalg**, в модулі **numpy.matlib** та модулі **numpy.matrix** (numpy.mat), а також в ППП sciру в модулі **scipy.linalg** та в ППП sympy у вигляді функції **Matrix**, яка містить і атрибути та методи, потрібні для реалізації символьного варіанта матричної алгебри. З прикладів застосування функції **Matrix**, використовуючи інформацію, наведену в [5], ми і продовжимо. Отже,

Пояснювальний приклад № 1

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import*                            # Виклик усіх функцій ППП sympy
In [3]: x,y,z,w = symbols('x y z w')                  # Оголошення x,y,z,w
                                                      ….символьними
In [4]: M = Matrix([[x,y],[z,w]]); M                  # Створення матриці (2x2)
Out[4]:                                               ….у символьному вигляді
Matrix([
[x, y],
[z, w]])
In [5]: M1 = Matrix([[1,2],[3,4]]); M1                # Створення матриці (2x2)
Out[5]:                                               ….у числовому вигляді
Matrix([
[1, 2],
[3, 4]])
In [6]: M2 = Matrix([5,6,7]);M2                       # Створення матриці-стовпця
Out[6]:
Matrix([
[5],
[6],
[7]])
In [7]: M.det()                                       # Обчислення визначника
                                                      ….матриці в символьному
Out[7]:                                               …. вигляді
w*x - y*z
In [8]: M1.det()                                      # Обчислення визначника
Out[8]:                                               …. матриці в числовому
-2                                                    …. вигляді
In [9]: M.T                                           # Транспонування матриці
Out[9]:                                               ….в символьному вигляді
Matrix([
[x, z],
[y, w]])
In [10]: M1.T                                         # Транспонування матриці
Out[10]:                                              ….в числовому вигляді
Matrix([
[1, 3],
[2, 4]])
In [11]: M2.T                                         # Транспонування матриці-
                                                      ….стовпця у матрицю-рядок
Out[11]:
Matrix([[5, 6, 7]])
In [12]: M3 = simplify(M1.inv());M3                   # Обчислення оберненої матриці
Out[12]:
Matrix([
[ -2,    1],
[3/2, -1/2]])
```

```
In [13]: M4 = M1**(-1);M4                    # Інший варіант обчислення
Out[13]:                                     ….оберненої матриці
Matrix([
[ -2,    1],
[3/2, -1/2]])
In [14]: M1*M3                               # Результат перемноження
                                             ….матриць
Out[14]:
Matrix([
[1, 0],
[0, 1]])
In [15]: M1+M3                               # Результат додавання матриць
Out[15]:
Matrix([
[ -1,   3],
[9/2, 7/2]])
In [16]: M1**2                               # Результат піднесення
Out[16]:                                     …. матриці до квадрату
Matrix([
[ 7, 10],
[15, 22]])
In [17]: M1.eigenvals()                      # Обчислення власних
Out[17]: {5/2 - sqrt(33)/2: 1,               …. чисел матриці
5/2 + sqrt(33)/2: 1}
In [18]: M1.eigenvects()                     # Обчислення власних
Out[18]:                                     …. векторів матриці
[(5/2 - sqrt(33)/2,1,
  [Matrix([
   [-2/(-3/2 + sqrt(33)/2)],[1]])]),
 (5/2 + sqrt(33)/2,1,
  [Matrix([
   [-2/(-sqrt(33)/2 - 3/2)],[1]])])]
In [19]: M5 = Matrix([3,2,1])                # Створення матриці
In [20]: M2.dot(M5)                          # Обчислення скалярного
                                             ….добутку двох матриць
Out[20]:
34
In [21]: M2.cross(M5)                        # Обчислення векторного
                                             ….добутку двох матриць
Out[21]:
Matrix([
[-8],
[16],
[-8]])
In [22]: a = symbols('a:2:2');a              # Символьне внесення
                                             ….послідовності з подвійним
Out[22]: (a00, a01, a10, a11)                ….індексуванням
```

```
In [23]: b = symbols('b:2:2');b                    # Символьне внесення
                                                   ….послідовності з подвійним
Out[23]: (b00, b01, b10, b11)                      …. індексуванням
In [24]: A = Matrix([[a[0],a[1]],[a[2],a[3]]]);A   # Індексна трансформація
                                                   ….послідовності в матрицю
Out[24]:
Matrix([
[a00, a01],
[a10, a11]])
In [25]: B = Matrix([[b[0],b[1]],[b[2],b[3]]]);B   # Індексна трансформація
                                                   ….послідовності в матрицю
Out[25]:
Matrix([
[b00, b01],
[b10, b11]])
In [26]: A*B                                       # Результат перемноження
                                                   ….індексно трансформованих
Out[26]:                                           …. матриць
Matrix([
[a00*b00 + a01*b10, a00*b01 + a01*b11],
[a10*b00 + a11*b10, a10*b01 + a11*b11]])
```

Кінець пояснювального прикладу № 1

А далі звертаємо увагу на те, що модуль **scipy.linalg** надає такі ж можливості у виконанні операцій лінійної алгебри, як і модуль **numpy.linalg**, але містить в собі і низку таких операцій, яких немає в модулі **numpy.linalg**, а тому використовується частіше. Однак, оскільки вхідні дані в програмах мовою Python задаються у вигляді числових масивів, які є атрибутами ППП numpy, то навіть при використанні для реалізації операцій лінійної алгебри модуля **scipy.linalg**, що входить до ППП scipy, потрібно перед цим внести в програму команду, якою викликати ППП numpy. Заслуговує бути відзначеним також те, що в рамках ППП numpy операції лінійної алгебри здійснюються з прямокутними числовими масивами за правилами здійснення матричних операцій, залишаючись однак в результатах теж числовими масивами. Якщо ж в рамках ППП numpy є необхідність від масивів перейти до матриць, потрібно трансформувати масиви певної розмірності в матриці тієї ж розмірності, використавши клас функцій **matrix**, до символу якого можна застосовувати скорочення **mat**. Отже,

Пояснювальний приклад № 2

```
In [1]: import numpy as np                         # Виклик ППП numpy np
In [2]: import scipy.linalg as la                  # Виклик модуля scipy.linalg як la
In [3]: x = np.array([[1,3],[4,6]])                # Внесення масиву x
In [4]: la.det(x)                                  # Обчислення визначника
Out[4]: -6.0                                       ….двовимірного масиву x
In [5]: la.norm(x)                                 # Обчислення норми масиву x
Out[5]: 7.874007874011811                          ….у двовимірному просторі
In [6]: np.round_(la.norm(x),2)                    # Обмеження числа знаків у
Out[6]: 7.87                                       ….нормі масиву x
In [7]: x1=la.inv(x);x1                            # Обчислення масиву,
Out[7]:                                            ….оберненого до масиву x
```

```
array([[-1.       ,  0.5      ],
[ 0.66666667, -0.16666667]])
In [8]: np.dot(x,x1)                                  # Перемноження масиву x
Out[8]:                                               ….на обернений масив x1
array([[1., 0.],
 [0., 1.]])
In [9]: w,v=la.eig(x)                                 # Обчислення власних чисел w
                                                        ….та власних векторів v масиву x
In [10]: w                                            # Виклик на екран власних
Out[10]: array([-0.77200187+0.j,                      ….чисел w масиву x
 7.77200187+0.j])
In [11]: v                                            # Виклик на екран власних
Out[11]:                                              ….векторів v масиву x
array([[-0.8610177 , -0.40503571],
 [ 0.50857499, -0.91430087]])
In [12]: w=w.real                                     # Залишення у власних числах
                                                      …. w лише дійсних частин та
In [13]: w                                            ….виклик їх на екран
Out[13]: array([-0.77200187,  7.77200187])
In [14]: np.round_(w,2)                               # Обмеження числа знаків у
Out[14]: array([-0.77,  7.77])                        ….виразах для власних чисел w
In [15]: np.round_(v,2)                               # Обмеження числа знаків у
Out[15]:                                              ….виразах для власних векторів v
array([[-0.86, -0.41],
 [ 0.51, -0.91]])
In [16]: v[:,0]                                       # Виклик на екран власного
Out[16]: array([-0.8610177 ,  0.50857499])            ….вектора, сформованого
                                                      ….першим власним числом
In [17]: v[:,1]                                       # Виклик на екран власного
Out[17]: array([-0.40503571, -0.91430087])            ….вектора, сформованого
                                                      ….другим власним числом
In [18]: X=np.matrix('1 3;4 6'); X                    # Створення матриці X(2x2) з
                                                      ….елементів двовимірної
Out[18]:                                              …. стрічки та її виклик
matrix([[1, 3],
 [4, 6]])
In [19]: print(X)                                     # Інший варіант виклику
[[1 3]                                                ….на екран матриці X(2x2)
 [4 6]]
In [20]: Y=np.matrix('4 2;1 5');Y                     # Створення матриці Y(2x2) з
                                                      ….елементів двовимірної
Out[20]:                                              …. стрічки та її виклик
matrix([[4, 2],
 [1, 5]])
In [21]: print(Y)                                     # Інший варіант виклику
[[4 2]                                                ….на екран матриці Y(2x2)
 [1 5]]
```

```
In [22]: X+Y                                  # Сума матриць X та Y
Out[22]:
matrix([[ 5,  5],
 [ 5, 11]])
In [23]: X*Y                                  # Перемноження матриці X
Out[23]:                                      ….на матрицю Y
matrix([[ 7, 17],
 [22, 38]])
In [24]: Y*X                                  # Перемноження матриці Y
Out[24]:                                      ….на матрицю X
matrix([[12, 24],
 [21, 33]])
In [25]: X**2                                 # Піднесення матриці X
Out[25]:                                      ….до другого степеня
matrix([[13, 21],
 [28, 48]])
In [26]: Y.T                                  # Транспонування матриці Y
Out[26]:
matrix([[4, 1],
 [2, 5]])
In [27]: Y.I                                  # Обчислення матриці,
Out[27]:                                      ….оберненої до матриці Y
matrix([[ 0.27777778, -0.11111111],
 [-0.05555556,  0.22222222]])
In [28]: np.round_(Y.I,2)                     # Обмеження числа знаків в
Out[281]:                                     ….оберненій матриці
array([[ 0.28, -0.11],
 [-0.06,  0.22]])
In [29]: Z=np.mat([[7,2],[-4,1]]);Z           # Створення матриці Z(2x2) з
                                              ….елементів двовимірного
Out[29]:                                      ….списку та її виклик
matrix([[ 7,  2],
 [-4,  1]])
In [30]: print(Z)                             # Інший варіант виклику
[[ 7  2]                                      ….на екран матриці Z(2x2)
 [-4  1]]
In [31]: N=np.arange(6)                       # Внесення послідовності
                                              ….чисел {0, 1, …, 5}
In [32]: N.shape=(2,3);N                      # Трансформація послідовності
Out[32]:                                      ….внесених чисел у масив N(2x3)
array([[0, 1, 2],                             ….з двома рядками і трьома
[3, 4, 5]])                                   ….стовпцями
In [33]: P=np.mat(N);P                        # Трансформація масиву N(2x3)
Out[33]:                                      ….в матрицю P(2x3) з тими ж
matrix([[0, 1, 2],                            …. рядками та тими ж стовпцями
        [3, 4, 5]])
```

Кінець пояснювального прикладу № 2.

Ну і на завершення викладення цієї додаткової інформації з мови Python, необхідної для реалізації операцій лінійної алгебри, нагадаємо, як розв'язувати алгебраїчні рівняння та системи таких рівнянь.

Функції і методи для розв'язання алгебраїчних рівнянь і їх систем є і в ППП sympy і в ППП scipy. І, звичайно ж, їх **розв'язки** несуть більше інформації, якщо вони отримані **в ППП sympy**, оскільки **мають вигляд формул**, що зв'язують результати розв'язання, тобто корені рівнянь, з коефіцієнтами цих рівнянь, у той час, як **розв'язки** цих же рівнянь **в ППП sciру є кортежами чисел**. Тому спочатку варто спробувати розв'язувати алгебраїчні рівняння в ППП sympy і, лише переконавшись, що в цьому пакеті розв'язки не можуть бути отримані, реалізовувати алгоритми розв'язання цих рівнянь в ППП sciру, не забувши перед цим викликати ППП numpy, адже при розв'язанні доведеться використовувати числові масиви, які є атрибутами саме ППП numpy.

В пакеті **sympy** при розв'язанні алгебраїчного рівняння ключову роль відіграє **функція Eq(,) у формі Eq(expr1,expr2),** де символом expr1 позначається ліва частина цього рівняння, складена відносно незалежної змінної, наприклад x, а символом expr2 позначається його права частина, яка, як правило, є якоюсь числовою константою. Якщо ж у правій частині рівняння стоїть нуль, то допустимою є і така форма запису **Eq(expr,0).** А безпосередньо **розв'язує** алгебраїчне рівняння, складене відносно незалежної змінної x, функція **solveset(Eq(,),x)**, яка формує розв'язок у вигляді числового масиву коренів. **Якщо** алгебраїчне **рівняння розв'язків не має,** то ця функція повідомляє про це символом **EmptySet ()**, а якщо рівняння виявилось занадто складним, тобто, **якщо** вона це рівняння **розв'язати не може,** то повідомляє про це символом **ConditionSet ().**

Отже,

Пояснювальний приклад № 3

```
In [1]: import sympy                          # Виклик ППП sympy
In [2]: from sympy import *                   # Доступ до усіх функцій sympy
In [3]: x = symbols ('x')                     # Оголошення символьною x
In [4]: expr1=x**2+5*x                        # Формування лівої частини
In [5]: expr2=-6                              # Формування правої частини
In [6]: solveset(Eq(expr1,expr2),x)           # Розв'язання рівняння з
                                              ….використанням
Out[6]:                                       ….функції Eq(expr1,expr2)
FiniteSet(-3, -2)
In [7]: expr=x**2+5*x+6                       # Формування рівняння без нуля
                                              ....справа
In [8]: solveset(Eq(expr,0),x)                # Розв'язання рівняння з
                                              ….використанням
Out[8]:                                       ….функції Eq(expr,0)
FiniteSet(-3, -2)
In [9]: solveset(Eq(expr1-expr2,0),x)         # Розв'язання рівняння з
                                              ….використанням
Out[9]:                                       ….функції Eq(expr1-expr2,0)
FiniteSet(-3, -2)
```

Кінець пояснювального прикладу № 3.

Якщо ж ми хочемо отримати розв'язок алгебраїчного рівняння у вигляді словника, в якому вказуються корені і їх кратність, то для цього **в ППП sympy** є функція **roots( )**, в аргументних дужках якої вказується спочатку ліва частина рівняння, приведеного до форми з нульовою правою частиною, а далі вказується символ змінної, наприклад x, відносно якої складене це рівняння. А у тих випадках, коли нас цікавлять лише **дійсні корені** рівняння в рамках ППП sympy, то потрібно використати функцію **real_roots( ),** в аргументних дужках якої вказується лише ліва частина рівняння, приведеного до форми з нульовою правою частиною. А для відтворення списку коренів рівняння **з заданою кількістю знаків** в рамках ППП sympy можна скористатись цією ж функцією у формі **nroots(,),** в аргументних дужках якої вказується спочатку ліва частина рівняння, приведеного до форми з нульовою правою частиною, а далі вказується кількість знаків, які ми хочемо відтворити в числових значеннях коренів.

Отже,

Пояснювальний приклад № 4

```
In [1]: import sympy                          # Виклик ППП sympy
In [2]: from sympy import *                   # Доступ до усіх функцій sympy
In [3]: x = symbols('x')                      # Оголошення символьною x
In [4]: f1 = x**5+7*x**4+19*x**3\             # Формування рівняння без нуля
          +25*x**2+16*x+4
In [5]: roots(f1,x)                           # Розв'язання рівняння з
                                              ….використанням функції roots()
Out[5]:                                       ….та формуванням коренів
{-2: 2, -1: 3}                                ….у вигляді словника
In [6]: f2=x**5+6*x**4+14*x**3\               # Формування рівняння без нуля
          +17*x**2+12*x+4
In [7]: roots(f2,x)                           # Розв'язання рівняння з
                                              ….використанням функції roots( )
Out[7]:                                       ….та формуванням коренів
{-1: 1, -2: 2, \                              ….у вигляді словника
-1/2 - sqrt(3)*I/2: 1, -1/2 + sqrt(3)*I/2: 1}
In [8]: real_roots(f2,x)                      # Розв'язання рівняння з
                                              ….використанням real_roots( )
Out[8]:                                       ….та формуванням лише
[-2, -2, -1]                                  ….дійсних коренів списком
In [9]: nroots(f2,3)                          # Розв'язання рівняння з
                                              ….використанням nroots( , )
Out[9]:                                       ….та формуванням
 [-2.00, -2.00, -1.00,\                       ….списку коренів з заданою
-0.5 - 0.866*I, -0.5 + 0.866*I]               ….кількістю знаків
```

Кінець пояснювального прикладу № 4.

Якщо нам потрібно за допомогою ППП sympy **розв'язати систему лінійних рівнянь**, складених, наприклад, з використанням змінних x,y,z, з лівими частинами у вигляді a1*x+b1*y+c1*z+d1, a2*x+b2*y+c2*z+d2, a3*x+b3*y+c3*z+d3, в яких коефіцієнти дійсні числа, та з нульовими правими частинами, **то** можемо скористатись функцією **linsolve(**[a1*x+b1*y+c1*z+d1, a2*x+b2*y+c2*z+d2, a3*x+b3*y+c3*z+d3],x,y,z) **або** функцією **linsolve(Matrix**(([a1,b1,c1,d1], [a2,b2,c2,d2],[a3,b3,c3,d3])),(x,y,z)). **А якщо** нам потрібно в

рамках ППП sympy розв'язати систему алгебраїчних рівнянь, наприклад, ar1=1, ar2=0, ar3=2, хоча б одне з яких має **нелінійний характер**, **то** доцільно використовувати функцію **solve([Eq(ar1,1),Eq(ar2,0),Eq(ar3,2)],x,y,z) або** функцію **solve([ar1-1, ar2, ar3-2],x,y,z).** Покажемо дію усіх цих функцій на прикладах.

Отже,

Пояснювальний приклад № 5

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import *                           # Доступ до усіх функцій sympy
In [3]: x,y,z = symbols('x y z')                      # Символізація змінних x,y,z
In [4]: linsolve([2*x+3*y+z-9, x-2*y\                 # Розв'язання системи трьох
+z,5*x-y+3*z-6], x, y, z)                             ….лінійних алгебраїчних
                                                      ….рівнянь з використанням
Out[4]:                                               …. функції linsolve([ , , ],x,y,z)
FiniteSet((-3, 12/5, 39/5))
In [5]: linsolve(Matrix(([2,3,1,-9],[1,-2,1,0],\      # Розв'язання системи трьох
                                                      ….лінійних алгебраїчних
[5,-1,3,-6])),(x,y,z))                                ….рівнянь з використанням
Out[5]:                                               ….функції linsolve(Matrix(([ ],[ ],
FiniteSet((3, -12/5, -39/5))                          ….[ ])),(x,y,z))
In [6]: solve([x**2-y, x-y-1], x, y)                  # Розв'язання системи двох
                                                      ….нелінійних алгебраїчних
Out[6]:                                               ….рівнянь з використанням
[(1/2 - sqrt(3)*I/2, -1/2 - sqrt(3)*I/2),             ….функції solve([ , ], x, y)
 (1/2 + sqrt(3)*I/2, -1/2 + sqrt(3)*I/2)]
In [7]: ]: solve([Eq(x**2-y,0), Eq(x-y,1)], x, y)     # Розв'язання системи двох
                                                      ….нелінійних алгебраїчних
Out[7]:                                               ….рівнянь з використанням
[(1/2 - sqrt(3)*I/2, -1/2 - sqrt(3)*I/2),             ….функції solve([Eq( , ),Eq( , )],
(1/2 + sqrt(3)*I/2, -1/2 + sqrt(3)*I/2)]              ….x, y)
```

Кінець пояснювального прикладу № 5.

6.3 Задачі на обчислення характеристик операторів в програмах мовою Python

Програма мовою Python для обчислення норми оператора диференціювання функції в метричному функціональному просторі C[a,b] для випадку, коли a = 1, b = 3

(Програма 20)

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import *                           # Доступ до усіх функцій sympy
In [3]: import numpy as np                            # Виклик ППП numpy як np
In [4]: t = symbols('t')                              # Символізація змінної t
In [5]: x = Function('x')(t)                          # Символізація функції x(t)
In [6]: y = Function('y')(t)                          # Символізація функції y(t)
In [7]: z=Function('z')(t)                            # Символізація функції z(t)
In [8]: x=2*t-0.5*t**2+t**0.5                         # Формування функції x(t)
```

```
In [9]: y=x.diff();y
Out[9]:
0.5*t**(-0.5) - 1.0*t + 2
In [10]: z=y.diff();z
Out[10]:
-0.25*t**(-1.5) - 1.0
In [11]: t=np.linspace(1,3,21)
In [12]: g1=lambda t: 2*t-0.5*t**2+t**0.5

In [13]: g1vec=np.vectorize(g1)
In [14]: g11=g1vec(t)
In [15]: g111=np.piecewise(g11,[g11<0,g11>=0],\
             [lambda g11:-g11,lambda g11:g11])
In [16]: ng1=g111.max();ng1
Out[16]: 3.4715750888103103
In [17]: ng1=np.round_(ng1,3)
In [18]: ng1
Out[18]: 3.472
In [19]: g2=lambda t: 2-1.0*t+0.5*t**(-0.5)

In [20]: g2vec=np.vectorize(g2)
In [21]: g22=g2vec(t)
In [22]: g222=np.piecewise(g22,[g22<0,g22>=0],\
             [lambda g22:-g22,lambda g22:g22])
In [23]: ng2=g222.max()
In [24]: ng2
Out[24]: 1.5
In [25]: nd_dt=ng2/ng1;nd_dt
Out[25]: 0.43202764976958524
In [26]: np.round_(nd_dt,3)
Out[26]: 0.432
In [27]: g3= lambda t: -0.25*t**(-1.5) - 1.0

In [28]: g3vec=np.vectorize(g3)
In [29]: g33=g3vec(t)
In [30]: g333=np.piecewise(g33,[g33<0,g33>=0],\
             [lambda g33:-g33,lambda g33:g33])
In [31]: ng3=g333.max()
In [32]: ng3
Out[32]: 1.25
In [33]: ndd_dtdt=ng3/ng1;ndd_dtdt

Out[33]: 0.36002304147465436
In [34]: np.round_(ndd_dtdt,3)
Out[34]: 0.36
```

Кінець програми 20

```
# Формування першої
….похідної від функції x(t)

# Формування другої
….похідної від функції x(t)

# Внесення масиву t∈ [1,3]
# Подання x(t) лямбда-функцією
….g1
# Векторизація функції g1 та її
….визначення на відрізку [1,3]
# Абсолютизація функції g11

# Визначення норми ng1 функції
….x(t) в просторі C[1,3]
# Обмеження кількості знаків
….в нормі та її виклик

# Подання y(t)
….лямбда-функцією g2
# Векторизація функції g2 та
….формування її значень g22
# Абсолютизація функції g22

# Визначення норми ng2 функції
….y(t), яка є результатом дії
….оператора диференціювання
# Обчислення норми nd_dt
….оператора диференціювання
# Обмеження кількості знаків в
….нормі nd_dt
# Подання z(t)
….лямбда-функцією g3
# Векторизація функції g3 та
….формування її значень g33
# Абсолютизація функції g33

# Визначення норми ng3 функції
….z(t), яка реалізує подвійне
….диференціювання функції x(t)
# Обчислення норми ndd_dtdt
….оператора подвійного
….диференціювання
# Обмеження кількості
….знаків в нормі ndd_dtdt
```

Програма мовою Python для обчислення норми оператора диференціювання функції в метричному функціональному просторі L_2[a,b] для випадку, коли a=0, b=2

(Програма 21)

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import *                           # Доступ до усіх функцій sympy
In [3]: t = symbols('t')                              # Символізація змінної t
In [4]: x = Function('x')(t)                          # Символізація функції x(t)
In [5]: y = Function('y')(t)                          # Символізація функції y(t)
In [6]: z=Function('z')(t)                            # Символізація функції z(t)
In [7]: x=1+2*t-0.5*t**2-0.25*t**3                    # Формування функції x(t)
In [8]: y=x.diff();y                                  # Формування функції y(t) як
Out[8]:                                               ….першої похідної від x(t)
-0.75*t**2 - 1.0*t + 2
In [9]: z=y.diff();z                                  # Формування функції z(t) як
Out[9]:                                               ….другої похідної від x(t)
-1.5*t - 1.0
In [10]: f1=x*x;f1                                    # Формування функції f1 як
Out[10]:                                              ….квадрата функції x(t)
4*(-0.125*t**3 - 0.25*t**2 + t + 1/2)**2
In [11]: f11=expand(f1);f11                           # Формування функції f11
Out[11]:                                              ….розкриттям дужок у f1
0.0625*t**6 + 0.25*t**5 - 0.75*t**4 - \
          2.5*t**3 + 3.0*t**2 + 4*t + 1
In [12]: b=integrate(f11,(t,0,2));b                   # Обчислення інтеграла від
Out[12]:                                              ….функції f11
7.00952380952381
In [13]: nx=b**0.5;nx                                 # Обчислення норми функції x(t)
Out[13]:
2.64755053011719
In [14]: f2=y*y;f2                                    # Формування функції f2 як
Out[14]:                                              ….квадрата функції y(t)
4*(-0.375*t**2 - 0.5*t + 1)**2
In [15]: f22=expand(f2);f22                           # Формування функції f22
Out[15]:                                              ….розкриттям дужок у f2
0.5625*t**4 + 1.5*t**3 - 2.0*t**2 - 4.0*t + 4
In [16]: b1=integrate(f22,(t,0,2));b1                 # Обчислення інтеграла від
Out[16]:                                              ….функції f22
4.26666666666667
In [17]: ny=b1**0.5;ny                                # Обчислення норми функції y(t)
Out[17]:
2.06559111797729
In [18]: f3=z*z;f3                                    # Формування функції f3 як
Out[18]:                                              ….квадрата функції z(t)
2.25*(-t - 0.666666666666667)**2
In [19}: f33=expand(f3);f33                           # Формування функції f33
Out[19]:                                              ….розкриттям дужок у f3
2.25*t**2 + 3.0*t + 1.0
```

```
In [20]: b2=integrate(f33,(t,0,2));b2          # Обчислення інтеграла від
Out[20]:                                      ….функції f33
14.0000000000000
In [21]: nz=b2**0.5;nz                        # Обчислення норми функції z(t)
Out[21]:
3.74165738677394
In [22]: nd_dt=ny/nx;nd_dt                    # Обчислення норми nd_dt
Out[22]:                                      ….оператора диференціювання
0.780189497605494                             ….d/dt
In [23]: ndd_dtdt=nz/nx;ndd_dtdt              # Обчислення норми ndd_dtdt
Out[23]:                                      ….оператора подвійного
1.41325249290268                              ….диференціювання d^2/dt^2
```

Кінець програми 21

Програма мовою Python для обчислення власних чисел і власних векторів матричного оператора, його резольвенти та групи операторів, ним породженої (Програма 22)

```
In [1]: import numpy as np                    # Виклик ППП numpy як np
In [2]: import scipy.linalg as la             # Виклик модуля scipy.linalg як la
In [3]: x = np.array([[1,3,5],[4,6,1],[-2,1,2]])   # Внесення масиву x
In [4]: X=np.mat(x);X                         # Створення матриці X(3x3) з
                                              ….елементів двовимірного
Out[4]:                                       …. списку та її виклик
matrix([[ 1,  3,  5],
        [ 4,  6,  1],
        [-2,  1,  2]])
In [5]: w,v=la.eig(X)                         # Обчислення власних чисел w
                                              ….та власних векторів v матриці X
                                              …. матриці X
In [6]: w                                     # Виклик власних чисел w
Out[6]:
array([7.82367917+0.j, 0.58816041\
           +2.72963561j,
0.58816041-2.72963561j])
In [7]: v                                     # Виклик власних векторів v
Out[7]:
array([[-0.41166616+0.j, 0.66977668+0.j,
 0.66977668-0.j],
 [-0.91120979+0.j,-0.37993767\
-0.29151858j, -0.37993767\
+0.29151858j],[-0.01508968+0.j ,\
 0.17279449+0.5405604j ,
 0.17279449-0.5405604j ]])
In [8]: v[:,0]                                # Виклик на екран власного
Out[8]: array([-0.41166616+0.j,               ….вектора, сформованого -
0.91120979+0.j,                               ….першим власним числом
-0.01508968+0.j])
```

```
In [9]: v[:,1]                                  # Виклик на екран власного
Out[9]:                                         …. вектора, сформованого
                                                …. другим власним числом
array([ 0.66977668+0.j, -0.37993767\
-0.29151858j, 0.17279449+0.5405604j ])
In [10]: v[:,2]                                 # Виклик на екран власного
                                                ….вектора, сформованого
Out[10]:                                        …. третім власним числом
array([ 0.66977668-0.j, -0.37993767\
+0.29151858j, 0.17279449-0.5405604j ])
In [11]: detX=la.det(X);detX                    # Обчислення визначника
                                                ….матриці X та його виклик
Out[11]: 60.99999999999999                      ….на екран
In [12]: nX=la.norm(X);nX                       # Обчислення норми матриці X
Out[12]: 9.848857801796104                      ….та її виклик на екран
In [13]: ]: omX=la.inv(X);omX                   # Обчислення матриці omX,
Out[13]:                                        ….оберненої до матриці X
array([[ 0.18032787, -0.01639344, -0.44262295], ….та її виклик на екран
 [-0.16393443,  0.19672131,  0.31147541],
 [ 0.26229508, -0.1147541 , -0.09836066]])
In [14]: import sympy                           # Виклик ППП sympy
In [15]: from sympy import *                    # Виклик усіх функцій sympy
In [16]: I=Matrix(eye(3));I                     # Створення одиничної матриці
Out[16]:                                        ….в ППП sympy
Matrix([
[1, 0, 0],
[0, 1, 0],
[0, 0, 1]])
In [17]: X1=Matrix(X);X1                        # Трансформація матриці X
Out[17]:                                        ….в ППП sympy у матрицю X1
Matrix([
[ 1, 3, 5],
[ 4, 6, 1],
[-2, 1, 2]])
In [18]: X2=X*X                                 # Піднесення до квадрата
In [19]: X22=Matrix(X2);X22                     ….матриці X і трансформація
                                                ….квадрата матриці X
Out[19]:                                        ….в ППП sympy у матрицю X22
Matrix([                                        ….та її виклик на екран
[ 3, 26, 18],
[26, 49, 28],
[-2,  2, -5]])
In [20]: X3=X2*X                                # Піднесення до кубу
In [21]: X33=Matrix(X3);X33                     ….матриці X і трансформація
Out[21]:                                        ….кубу матриці X
Matrix([                                        …. в ППП sympy у матрицю X33
[ 71, 183,  77],                                ….та її виклик на екран
```

```
[166, 400, 235],
[ 16,   1, -18]])
In [22]: q,t =symbols('q t')                    # Оголошення символьними q,t
In [23]: R=I+X1*q+X22*q**2+X33*q**3;R           # Апроксимація резольвенти R
Out[23]:                                        ….матричного оператора X
Matrix([                                        ….чотирма першими членами її
[71*q**3 + 3*q**2 + q + 1, 183*q**3 \           …. розкладу у степеневий ряд
+ 26*q**2 + 3*q, 77*q**3 + 18*q**2 + 5*q],
[166*q**3 + 26*q**2 + 4*q, 400*q**3 \
+ 49*q**2 + 6*q + 1, 235*q**3 + 28*q**2 + q],
[  16*q**3 - 2*q**2 - 2*q,  q**3 + 2*q**2 \
+ q, -18*q**3 - 5*q**2 + 2*q + 1]])
In [24]: u=I+X1*t+X22*t**2/2+X33*t**3/6         # Апроксимація групи
                                                ….операторів u, породженої
Out[24]:                                        ….інфінітезимальним
Matrix([                                        ….оператором X, чотирма
[71*t**3/6 + 3*t**2/2 + t + 1, 61*t**3/2 \      ….першими членами її
                                                ….розкладу у степеневий ряд
+ 13*t**2 + 3*t,     77*t**3/6 + 9*t**2 + 5*t],
[  83*t**3/3 + 13*t**2 + 4*t, 200*t**3/3 +
 49*t**2/2 + 6*t + 1,
 235*t**3/6 + 14*t**2 + t],
[ 8*t**3/3 - t**2 - 2*t, t**3/6 + t**2\
 + t, -3*t**3 - 5*t**2/2 + 2*t + 1]])
```

Кінець програми 22

Розділ 7 ПРИКЛАДНІ АСПЕКТИ ТЕОРІЇ ОПЕРАТОРІВ (В ПРИКЛАДАХ І ПРОГРАМАХ)

7.1 Метод стиснених відображень та алгоритми його реалізації

У кінці однойменного підрозділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. В чому суть ідеї методу стиснених відображень оператора?
2. Застосовуючи метод стиснення відображень, доведіть, що алгебраїчне рівняння має єдину нерухому точку на заданому відрізку числової осі.
3. Застосовуючи метод стиснення відображень, доведіть існування єдиної нерухомої точки диференціального оператора.
4. Застосовуючи метод стиснення відображень, доведіть існування єдиної нерухомої точки оператора Фредгольма.
5. Що собою являє резольвента оператора Фредгольма?
6. За яким критерієм визначається досягнення потрібної точності наближеного розв'язку неоднорідного інтегрального рівняння Фредгольма 2-го роду?

Тож із відповідей на ці питання ми і розпочнемо викладення змісту сьомого розділу другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих запитань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2]. Отже, почнемо викладення матеріалу цього розділу з розлогої, але з пропусками і деякими перефразуваннями та зміною нумерації формул відфільтрованої цитати, взятої з нашого базового навчального посібника [2].

Цей метод є одним із ключових для багатьох застосунків функціонального аналізу в прикладних задачах ІТ-сфери, але, перш ніж викласти його суть, ***наведемо теорему про єдину спільну точку*** послідовності вкладених одна в одну замкнутих куль, яка базується на матеріалі, викладеному в підрозділі 1.2 з використанням виразів (1.41) та (1.42) і формулюється так: ***послідовність вкладених одна в одну замкнених куль*** $K_1 \supset K_2 \supset K_3 \supset ...,$ ***які є підмножинами повного метричного простору R, таких що***

$$K_n = \{x_n, \rho(x, x_n) \le r_n\}, \quad n = 1,2,3,..., \tag{7.1}$$

і радіуси r_n ***яких наближаються до нуля при*** $n \to \infty$***, мають єдину спільну точку*** x_0**.**

Доведення цієї теореми базується на тому, що для послідовності (7.1), яким би ми малим не вибрали число $\varepsilon > 0$, завжди можна побудувати замкнуту кулю з радіусом $r_k < \varepsilon$, для якої справедливим буде вираз

$$\rho(x, x_k) \le r_k, \tag{7.2}$$

з якого, в силу повноти простору *R*, випливає, що які б дві точки цього простору ми не взяли, завжди можна буде побудувати кулю з радіусом, меншим відстані між цими точками, а тому послідовність $\{x_n\}$ є фундаментальною, для якої справедливо

$$\lim_{n \to \infty} x_n = x_0. \tag{7.3}$$

А з виразів (7.1), (7.2), (7.3) і випливає, що з вкладенням кулі в кулю, радіус кожної з наступних із яких є усе меншим, ми прийдемо до єдиної спільної для усіх цих куль точки x_0 простору R, в якому ця послідовність куль формується.

А тепер уявімо, що послідовність таких «куль» $[a_i, b_i]$, $i = 0,1,2,...,n$ ми формуємо на відрізку $[a,b]$ числової осі x за умови, що

$$\begin{aligned} &b = b_0 > b_1 > b_2 > ... > b_n > ..., \\ &a = a_0 < a_1 < a_2 < ... < a_n < ... \end{aligned} \tag{7.4}$$

Згідно зі сформульованою вище та доведеною теоремою ця послідовність, стискуючись, прямуватиме до єдиної спільної точки на числовій осі X. Тож, якщо ми визначимо закон $f(x)$, за яким при стисненні відрізка $[a,b]$ числової осі x кожній точці x_i, $i = 0,1,2,...,n$ попереднього відрізка $[a_i, b_i]$, $i = 0,1,2,...,n$ відповідатиме точка x_i^*, $i = 1,2,...,n$ наступного відрізка, вкладеного в попередній, то знайти цю єдину спільну точку можна, розв'язавши рівняння

$$x = f(x). \tag{7.5}$$

А далі випишемо означення: ***відображення, задане в метричному просторі R оператором A, називається стисненим, якщо для*** $x_1 \in R$ ***і*** $x_2 \in R$ ***виконується нерівність***

$$\rho(Ax_1, Ax_2) \le \theta\rho(x_1, x_2), \tag{7.6}$$

де

$$0 < \theta < 1 \tag{7.7}$$

Далі доцільною для розгляду є ***теорема Банаха***, яка, фактично, є розширенням на оператори доведеної вище теореми про єдину спільну точку послідовності вкладених одна в одну куль, заданих у повному метричному просторі, і яка формулюється так: ***стиснене відображення оператором A повного метричного простору R в себе має одну і тільки одну нерухому точку.***

Для доведення цієї теореми припустимо, що оператор ***A*** діє на точку $x_i \in R$, $i = 0,1,2,...,n$ так, що

$$x_1 = Ax_0, \quad x_2 = Ax_1, ..., x_n = Ax_{n-1}, ... \tag{7.8}$$

Покладемо також, що

$$\begin{aligned} &\rho(x_1, x_2) = \theta\rho(x_0, x_1), \\ &\rho(x_2, x_3) = \theta\rho(x_1, x_2) = \theta^2\rho(x_0, x_1), \\ &..... \\ &\rho(x_{n-1}, x_n) = \theta^{n-1}\rho(x_0, x_1), \end{aligned} \tag{7.9}$$

де θ задається виразом (7.7).

Задамо $q > n$. Тоді, згідно з виразами (7.6) і (7.9), матимемо

$$\begin{aligned} &\rho(x_n, x_q) = \rho(Ax_{n-1}, Ax_{q-1}) \le \theta\rho(x_{n-1}, x_{q-1}) = \theta\rho(Ax_{n-2}, Ax_{q-2}) \le \theta^2\rho(x_{n-2}, x_{q-2}) = \\ &= \theta^2\rho(Ax_{n-3}, Ax_{q-3}) \le \theta^3\rho(x_{n-3}, x_{q-3}) = ... = \theta^{n-1}\rho(Ax_0, Ax_{q-n}) \le \theta^n\rho(x_0, x_{q-n}) \end{aligned} \tag{7.10}$$

На підставі виразу (1.35) для метрики $\rho(x_0, x_{q-n})$ можна записати, що

$$\rho(x_0, x_{q-n}) = \rho(x_0, x_1) + \rho(x_1, x_2) + ... + \rho(x_{q-(n-1)}, x_{q-n}) \tag{7.11}$$

Залишимо в нерівності (7.10) лише перший і останній члени, що лише підсилить її. В результаті такого кроку отримаємо, що

$$\rho(x_n, x_q) \le \theta^n \rho(x_0, x_{q-n}). \tag{7.12}$$

Нехай

$$q, n \to \infty. \tag{7.13}$$

Підставивши вирази (7.9) у (7.11), а результат цієї підстановки у вираз (7.12), з урахуванням умови (7.7), отримаємо

$$\rho(x_n, x_q) \le \theta^n \rho(x_0, x_1)\left(1 + \theta + \theta^2 + \ldots + \theta^{q-n-1} + \ldots\right) = \frac{\theta^n}{1-\theta}\rho(x_0, x_1) \tag{7.14}$$

Зауважимо, що при отриманні виразу (7.14) ми використали формулу для суми членів нескінченно спадної геометричної прогресії зі знаменником θ.

Легко бачити, що зі зростанням n множник θ^n у правій частині нерівності (7.14), внаслідок виконання умови (7.7), буде наближатись до нуля, що, в свою чергу, свідчить про те, що зі зростанням n, q і ліва частина цієї нерівності наближатиметься до нуля, тобто, що послідовність $\{x_n\}$ є фундаментальною, а тому для загального члена

$$x_n = Ax_{n-1} \tag{7.15}$$

послідовності (7.8) при $n \to \infty$ справедливою є рівність

$$x = Ax, \tag{7.16}$$

яка, згідно з виразом (7.5) теореми про існування нерухомої точки, підтверджує ту частину теореми Банаха, в якій стверджується, що стиснене відображення, яке здійснюється оператором $\boldsymbol{A}$ в повному метричному просторі $\boldsymbol{R}$, має нерухому точку x. І цілком очевидно, що ця нерухома точка є розв'язком операторного рівняння (7.16).

А далі, слідуючи викладеному у нашій роботі [2], дамо послідовно та взаємопов'язано відповіді на інші поставлені у цьому підрозділі питання. І почнемо з того, що розглянемо алгебраїчне рівняння

$$F(x) = 0, \tag{7.17}$$

задане на відрізку $[a,b]$, наприклад , рівняння

$$x^2 - 5x + 4 = 0 \tag{7.18}$$

задане на відрізку [0,2].

Приведемо рівняння (7.17) до вигляду (7.5), тобто, до вигляду

$$x = f(x), \tag{7.19}$$

яким для виразу (7.18) буде рівняння

$$x = \frac{1}{5}x^2 + \frac{4}{5}. \tag{7.20}$$

Накладемо на функцію $f(x)$ умови, щоб вона на відрізку $[a,b]$ була неперервною, диференційованою та щоб її значеннями були числа з цього ж відрізка, тобто, щоб

$$\forall x \in [a,b] \Rightarrow f(x) \in [a,b]. \tag{7.21}$$

Для прикладу (7.20) умова (7.21) набуває вигляду

$$\forall x \in [0,2] \Rightarrow \left(\frac{1}{5}x^2 + \frac{4}{5}\right) \in [0,2]. \tag{7.22}$$

Підстановкою значень аргументу x із відрізка [0,2] у вираз (7.22) легко переконатись у тому, що умова (7.21) для функції (7.20) нашого прикладу виконується.

Накладемо на функцію $f(x)$ ще одну умову, щоб її похідна $f'(x)$ в усіх точках відрізка $[a,b]$ за модулем була меншою одиниці, тобто, щоб

$$\forall x \in [a,b] \Rightarrow |f'(x)| \le k < 1, \tag{7.23}$$

де

$$k = \frac{\max}{x \in [a,b]} |f'(x)|. \tag{7.24}$$

Для прикладу (7.20) матимемо

$$f'(x) = \frac{d}{dx}\left(\frac{1}{5}x^2 + \frac{4}{5}\right) = \frac{2}{5}x, \tag{7.25}$$

$$k = \frac{\max}{x \in [0,2]} \left|\frac{2}{5}x\right| = \frac{4}{5} < 1. \tag{7.26}$$

Як бачимо з виразів (7.25), (7.26), умова (7.7) для нашого прикладу (7.20) виконується.

Надамо відрізку $[a,b]$ статус метричного простору з метрикою

$$\rho(x_1, x_2) = |x_1 - x_2| \tag{7.27}$$

і припустимо, що оператор $\boldsymbol{A}$ здійснює відображення цього простору самого в себе, тобто, що вираз (7.19) набуває вигляду

$$Ax = f(x). \tag{7.28}$$

Доведемо, що відображення (7.28) є стисненим, а тому має єдину нерухому точку, яка і є коренем алгебраїчного рівняння (7.17).

Підставляючи вираз (7.28) у вираз для метрики (7.27), отримаємо

$$\rho(Ax_1, Ax_2) = |f(x_1) - f(x_2)|. \tag{7.29}$$

Якщо ми помножимо і розділимо праву частину виразу (7.29) на метрику $|x_1 - x_2|$, то отримаємо

$$\rho(Ax_1, Ax_2) = \frac{|f(x_1) - f(x_2)|}{|x_1 - x_2|} |x_1 - x_2| = |f'(\xi)||x_1 - x_2|, \tag{7.30}$$

де

$$\xi \in [x_1, x_2] \subset [a,b]. \tag{7.31}$$

Для нашого прикладу (7.20) вираз (7.31) матиме вигляд

$$\xi \in [x_1, x_2] \subset [0,2]. \tag{7.32}$$

З урахуванням виразів (7.23), (7.24) та (7.31) вираз (7.30) набуває вигляду

$$\rho(Ax_1, Ax_2) = |f'(\xi)||x_1 - x_2| \le k|x_1 - x_2| = k\rho(x_1, x_2), . \tag{7.33}$$

Оскільки, як показано вище, $k<1$ (для нашого прикладу $k=\frac{4}{5}=0{,}8$, що випливає з виразу (7.26), то вираз (7.33) задає стиснене відображення, яке, згідно з теоремою Банаха, має єдину нерухому точку x^*.

Використовуючи вирази (7.3), (7.15) та (7.16), можна записати, що

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty} Ax_{n-1} = Ax^* = x^*. \tag{7.34}$$

Фактично виразом (7.34) ми задаємо алгоритм руху шляхом послідовних наближень від довільної точки x_0 відрізка $[a,b]$ до нерухомої точки x^*.

Для нашого прикладу (7.20) цей алгоритм, виходячи з виразів (7.19), (7.20) та (7.28), матиме вигляд

$$\lim_{n\to\infty} x_n = \lim_{n\to\infty}\left(\frac{1}{5}x_{n-1}^2+\frac{4}{5}\right). \tag{7.35}$$

Точність наближень на кожному кроці алгоритму (7.34) ми можемо оцінити за виразом (7.14), який для нашого випадку набуває вигляду

$$\left|x_n - x^*\right| \le k^n \frac{\left|x_0 - x_1\right|}{1-k}. \tag{7.36}$$

А далі розглянемо простір ***C[a,b]*** неперервних функцій $y=y(x)$, заданих на відрізку $x\in[a,b]$ числової осі, зі значеннями, заданими на відрізку $y\in[M,N]$ числової осі, та метрикою

$$\rho(y,y_1)=\sup_{x\in[a,b]}\left|y-y_1\right|, \tag{7.37}$$

де y, y_1 – точки цього простору.

Нехай у просторі ***C[a,b]*** задане диференціальне рівняння

$$y' = f(x,y) \tag{7.38}$$

з початковою умовою

$$y(x_0)=y_0. \tag{7.39}$$

Накладемо на функцію $f(x,y)$ умову, щоб вона була такою, що задовольняє умови Ліпшиця, тобто, щоб

$$\left|f(x,y_0)-f(x,y_1)\right| \le L\left|y_0-y_1\right|, \tag{7.40}$$

де $(x,y_0),(x,y_1)$ – точки прямокутної області $\boldsymbol{G}$, обмеженої відрізками $x\in[a,b]$, $y\in[M,N]$ на площині (x,y), L – константа, числове значення якої визначимо дещо пізніше.

Проінтегруємо диференціальне рівняння (7.38) в межах від x_0 до x

$$\int_{x_0}^{x}\frac{dy}{dx}dx = \int_{x_0}^{x} f(x,y)dx. \tag{7.41}$$

Матимемо

$$y(x)-y(x_0)=\int_{x_0}^{x} f(x,y)dx, \tag{7.42}$$

або

$$y = y_0 + \int_{x_0}^{x} f(x,y)dx. \tag{7.43}$$

Очевидно, що інтегральне рівняння (7.43) в операторному вигляді можна переписати і так

$$y = Ay. \tag{7.44}$$

Якщо ми доведемо, що оператор ***A*** здійснює стиснене відображення простору функцій ***C[a,b]*** в себе, то тим самим ми доведемо, що це відображення задає єдину нерухому точку в цьому просторі, яка у функціональній інтерпретації є розв'язком інтегрального рівняння (7.43), яке, в свою чергу, є іншою формою запису диференціального рівняння (7.38), а тому це доведення буде одночасно і доведенням того, що у заданому просторі як інтегральне рівняння (7.43), так і диференціальне рівняння (7.38) мають розв'язок, який до того ж є єдиним.

Застосовуючи метод стиснених відображень, маємо право записати, що

$$\begin{aligned}\rho(y_1,y_2) = \rho(Ay_0,Ay_1) = \sup_{x\in[a,b]}|Ay_0 - Ay_1| \le \sup_{x\in[a,b]}\int|f(x,y_0) - f(x,y_1)||dx| \le \\ \le \sup_{x\in[a,b]}\int_{x_0}^{x} L|y_0 - y_1||dx| = L|x - x_0|\sup_{x\in[a,b]}|y_0 - y_1| = \theta\rho(y_0,y_1),\end{aligned} \tag{7.45}$$

де

$$\theta = L|x - x_0|. \tag{7.46}$$

Згідно з ідеологією методу стиснених відображень для того, щоб вираз (7.45) реалізовував цю ідеологію, потрібно, щоб виконувалась нерівність

$$\theta < 1. \tag{7.47}$$

Порівнюючи вирази (7.46) та (7.47), бачимо, що вираз (7.45) реалізовуватиме ідеологію методу стиснених відображень і спрямовуватиме послідовність

$$y_1 = Ay_0,\ y_2 = Ay_1,\ y_3 = Ay_2,...,y_n = Ay_{n-1},... \tag{7.48}$$

до єдиної нерухомої точки y області ***G***, яка буде проєкцією функції $y = y(x)$ на метричний функціональний простір ***C[a,b]*** та розв'язком інтегрального рівняння (7.43) у тому випадку, коли константа ***L*** буде задовольняти нерівність

$$|x - x_0| < \frac{1}{L}. \tag{7.49}$$

Ну і нарешті, використовуючи метод стиснених відображень, доведемо існування розв'язку інтегрального рівняння Фредгольма 2-го роду

$$\varphi(x) = f(x) + \lambda\int_{a}^{b} K(x,y)\varphi(y)dy, \tag{7.50}$$

яке пов'язує між собою в просторі неперервних функцій ***C[a,b]*** неперервні функції $\varphi(x)$ з нормою

$$\|\varphi(x)\| = \sup_{x\in[a,b]}|\varphi(x)| \tag{7.51}$$

з неперервними в точках площини (x,y), обмеженої межами прямокутника $[a \le x, y \le b]$, функціями $K(x,y)$ з нормою

$$\|K(x,y)\| = \sup_{[a \le x, y \le b]} |K(x,y)|. \tag{7.52}$$

Якщо в рівнянні (7.50) припустити, що

$$f(x) = 0, \tag{7.53}$$

то отримаємо інтегральне рівняння Фредгольма 1-го роду або, що одне і те ж, однорідне рівняння Фредгольма.

Оператор A який в просторі C перетворює клас функцій φ в самих себе згідно з виразом

$$A\varphi = \lambda \int_a^b K(x,y)\varphi(y)dy, \tag{7.54}$$

називається оператором Фредгольма з ядром $K(x,y)$.

З урахуванням виразу (7.54) рівняння Фредгольма (7.50) можна переписати в операторному вигляді як

$$\varphi = f + A\varphi, \tag{7.55}$$

а його n-та ітерація при послідовних наближеннях до розв'язку матиме вигляд

$$\varphi_n = f + A\varphi_{n-1}. \tag{7.56}$$

Оператор Фредгольма має три основних властивості: по-перше, він лінійний, по-друге, він неперервний, по-третє, він обмежений, що достатньо і для доведення, що відображення ним у метричному просторі має єдину нерухому точку, яка є розв'язком рівняння Фредгольма, і для побудови резольвентного алгоритму отримання цього розв'язку. Розкриємо ці властивості:

властивість перша - оператор Фредгольма лінійний, тому що

$$\begin{aligned} A(\varphi_1 + \varphi_2) &= \lambda \int_a^b K(x,y)(\varphi_1(y) + \varphi_2(y))dy = \lambda \int_a^b K(x,y)\varphi_1(y)dy + \\ &+ \lambda \int_a^b K(x,y)\varphi_2(y)dy = A\varphi_1 + A\varphi_2 \end{aligned}; \tag{7.57}$$

властивість друга – оператор Фредгольма неперервний, тому що, якщо існує границя φ послідовності $\{\varphi_n\}$, тобто,

$$\lim_{n\to\infty} \varphi_n = \varphi, \tag{7.58}$$

де

$$\varphi_n(x) = f(x) + \lambda \int_a^b K(x,y)\varphi_{n-1}(y)dy = f + A\varphi_{n-1}, \tag{7.59}$$

то

$$\begin{aligned} \lim_{n\to\infty} \varphi_n &= \lim_{n\to\infty} \left(f(x) + \lambda \int_a^b K(x,y)\varphi_{n-1}(y)dy \right) = f(x) + \lambda \int_a^b K(x,y) \lim_{n\to\infty} \varphi_{n-1}(y)dy = \\ &= f(x) + \lambda \int_a^b K(x,y)\varphi(y)dy = f + A\varphi = \varphi \end{aligned} \tag{7.60}$$

властивість третя – оператор Фредгольма обмежений, тому що

$$\begin{aligned}
&\|A\varphi\| = \sup_{[a\le x,y\le b]}|A\varphi| = \sup_{[a\le x,y\le b]}\left|\lambda\int_a^b K(x,y)\varphi(y)dy\right| \le \\
&\le \sup_{[a\le x,y\le b]}\lambda\int_a^b |K(x,y)|\,|\varphi(y)|\,dy = \\
&= \lambda\int_a^b \sup_{[a\le x,y\le b]}|K(x,y)| \sup_{[a\le y\le b]}|\varphi(y)|\,dy = \lambda\|K(x,y)\|\,\|\varphi(y)\|\int_a^b dy = \\
&= \lambda\,\|K(x,y)\|\,\|\varphi(y)\|\,(b-a)
\end{aligned} \tag{7.61}$$

Покажемо, що оператор Фредгольма здійснює стиснене відображення простору функцій φ самого в себе. Для цього застосуємо стандартний алгоритм методу стиснених відображень, але не до метрик, а до норм, згідно з яким матимемо

$$\begin{aligned}
&\|\varphi_1,\varphi_2\| = \sup_{[a\le x\le b]}|\varphi_1(x)-\varphi_2(x)| = \sup_{[a\le x\le b]}|(f+A\varphi_0)-(f+A\varphi_1)| = \\
&= \sup_{[a\le x\le b]}|A(\varphi_0-\varphi_1)| = \sup_{[a\le x\le b]}\left|\lambda\int_a^b K(x,y)(\varphi_0(y)-\varphi_1(y))dy\right| \le \\
&\le \lambda\int_a^b \sup_{[a\le x,y\le b]}|K(x,y)| \sup_{[a\le y\le b]}|\varphi_0(y)-\varphi_1(y)|dy = \\
&= \lambda\|K(x,y)\|\,\|\varphi_0,\varphi_1\|\int_a^b dy = \lambda\|K(x,y)\|(b-a)\|\varphi_0,\varphi_1\| = \theta\|\varphi_0,\varphi_1\|,
\end{aligned} \tag{7.62}$$

$$\begin{aligned}
&\|\varphi_2,\varphi_3\| = \sup_{[a\le x\le b]}|\varphi_2(x)-\varphi_3(x)| = \sup_{[a\le x\le b]}|(f+A(A\varphi_0))-(f+A(A\varphi_1))| = \\
&= \sup_{[a\le x\le b]}|A^2(\varphi_0-\varphi_1)| = \sup_{[a\le x\le b]}\left|\lambda\int_a^b K(x,y)[\lambda\int_a^b K(y,\omega)(\varphi_0(\omega)-\varphi_1(\omega))d\omega]dy\right| \le \\
&\le \lambda^2\int_a^b\{[\sup_{[a\le x,y\le b]}|K(x,y)|]^2 \sup_{[a\le\omega\le b]}|\varphi_0(\omega)-\varphi_1(\omega)|\int_a^b d\omega\}dy = \\
&= \lambda^2\|K(x,y)\|^2\,\|\varphi_0,\varphi_1\|(b-a)\int_a^b dy = \lambda^2\|K(x,y)\|^2(b-a)^2\|\varphi_0,\varphi_1\| = \theta^2\|\varphi_0,\varphi_1\|,
\end{aligned} \tag{7.63}$$

Продовжуючи, за аналогією, для послідовності (7.56) отримаємо

$$\begin{aligned}
&\|\varphi_n,\varphi_{n+1}\| = \sup_{[a\le x\le b]}|\varphi_n(x)-\varphi_{n+1}(x)| = \sup_{[a\le x\le b]}|(f+A(A^{n-1}\varphi_0))-(f+A(A^{n-1}\varphi_1))| = \\
&= \sup_{[a\le x\le b]}|A^n(\varphi_0-\varphi_1)| \le \lambda^n\|K(x,y)\|^n(b-a)^n\|\varphi_0,\varphi_1\| = \theta^n\|\varphi_0,\varphi_1\|
\end{aligned} \tag{7.64}$$

У виразах (7.62), (7.63), (7.64)

$$\theta = \lambda\|K(x,y)\|(b-a). \tag{7.65}$$

Як ми пам'ятаємо, для того, щоб вирази (7.62), (7.63), (7.64) здійснювали стиснення, що веде до нерухомої точки, яка є розв'язком інтегрального рівняння (7.50), необхідно, щоб виконувалась нерівність (7.47).

А оскільки норма $\|K(x,y)\|$ ядра $K(x,y)$ оператора Фредгольма і відрізок $[a,b]$ числової осі, на якому здійснюється інтегрування в інтегральному рівнянні (7.50), визначаються

умовами задачі, для якої синтезоване це рівняння, то з виразів (7.51), (7.52) витікає, що для того, щоб неоднорідне інтегральне рівняння Фредгольма (7.50) мало єдиний розв'язок $\varphi(x)$, потрібно вибирати параметр λ, виходячи зі співвідношення

$$\lambda < \frac{1}{\|K(x,y)\|\,(b-a)}. \tag{7.66}$$

Оскільки норма $\|\varphi_n, \varphi_{n+1}\|$ задає відстань між двома точками φ_n і φ_{n+1} метричного простору ***C[a,b]*** неперервних функцій з метрикою (7.37), а тому

$$\rho(\varphi_n, \varphi_{n+1}) = \frac{\sup}{[a \le x \le b]} |\varphi_n(x) - \varphi_{n+1}(x)| = \|\varphi_n, \varphi_{n+1}\|, \tag{7.67}$$

то вираз (7.64), який ми використали для того, щоб довести, що оператор Фредгольма здійснює стиснення відображень у заданому функціональному метричному просторі, можна використовувати також і для прийняття рішення про те, на якому етапі можна зупиняти ітераційний процес, який має вигляд

$$\begin{aligned}
&\varphi_0(x) = f(x) = A^0 f,\\
&\varphi_1(x) = f(x) + \lambda \int_a^b K(x,y)\varphi_0(y)dy = f(x) + \lambda \int_a^b K(x,y) f(y)dy = A^0 f + Af,\\
&\varphi_2(x) = f(x) + \lambda \int_a^b K(x,y)\varphi_1(y)dy = A^0 f + Af + A^2 f,\\
&\text{.....},\\
&\varphi_n(x) = f(x) + \lambda \int_a^b K(x,y)\varphi_{n-1}(y)dy = \sum_{i=0}^{n} A^i f(x),
\end{aligned} \tag{7.68}$$

Для цього потрібно задатись допустимим значенням ε похибки наближення і зупинити обчислення за виразами (7.68) на тому значенні n, для якого виконуватиметься нерівність

$$\|\varphi_n, \varphi_{n+1}\| < \varepsilon. \tag{7. 69}$$

Ну і завершимо ми теоретичні посилання цього підрозділу, які дають відповіді на питання, сформульовані на його початку, побудовою резольвенти оператора Фредгольма. Для цього перепишемо операторне рівняння (7.55) у вигляді

$$\varphi - A\varphi = f. \tag{7.70}$$

Операторне рівняння (7.70) можна подати і так:

$$(I - A)\varphi = f. \tag{7.71}$$

Визначимо обернений оператор $(I - A)^{-1}$ і помножимо на нього зліва операторне рівняння (7.71). Отримаємо

$$(I - A)^{-1}(I - A)\varphi = (I - A)^{-1} f, \tag{7.72}$$

або

$$\varphi = (I - A)^{-1} f = R_{\Phi} f. \tag{7.73}$$

Згідно з означенням, наведеним у підрозділі 6.1, оператор

$$R_{\Phi} = (I - A)^{-1} = \frac{I}{I - A} = I + A + A^2 + A^3 + \ldots = \sum_{i=0}^{\infty} A^i \tag{7.74}$$

і є резольвентою операторного рівняння Фредгольма. А оскільки, визначаючи її, ми використали формулу для суми членів нескінченно спадної геометричної прогресії зі знаменником A, то існуватиме резольвента R_{Φ} лише в тому випадку, коли

$$\|A\| < 1, \tag{7.75}$$

тобто, коли

$$\lambda < \frac{1}{\|K(x,y)\|\,(b-a)\|\varphi\|}, \tag{7.76}$$

що цілком узгоджується з виразом (7.66).

З урахуванням виразу (7.74) розв'язок (7.73) операторного рівняння (7.55) можна переписати і так

$$\varphi(x) = \sum_{i=0}^{\infty} A^i f(x) \tag{7.77}$$

А застосовуючи до останнього рівняння в системі наближень (7.68) граничний перехід, матимемо

$$\lim_{n\to\infty} \varphi_n(x) = \lim_{n\to\infty} \sum_{i=0}^{n} A^i f(x) = \sum_{i=0}^{\infty} A^i f(x). \tag{7.78}$$

Із виразів (7.77) та (7.78) випливає, що

$$\lim_{n\to\infty} \varphi_n(x) = \varphi(x). \tag{7. 79}$$

Цим ми підтвердили, що в процесі послідовних наближень (7.68) ми обов'язково прийдемо до наближеного розв'язку рівняння Фредгольма 2-го роду, похибка якого не перевищуватиме задану нами величину ε.

7.2 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з реалізацією методу стиснених відображень

Аналізуючи характер формул, викладених у попередньому підрозділі 7.1, бачимо, що для їх реалізації в програмах мовою Python нам будуть потрібні деякі нові знання функцій і методів цієї мови програмування, додаткові до тих, про які уже йшла мова в підрозділах, присвячених викладенню додаткових відомостей з технології програмування мовою Python у першій частині цього навчального посібника та у підрозділі 6.2 другої його частини, які ми дублювати не будемо.

Легко бачити, для програмної реалізації мовою Python виразів (7.5), (7.15), (7.16), (7.19), (7.20), (7.21), (7.34), (7.35), (7.55), (7.58), (7.68), (7.77), (7.78), (7. 79) до тих знань, що ми уже маємо з цієї мови, потрібно додати інформацію про те, як реалізовувати циклічні алгоритми обчислень, що крок за кроком наближають нас до отримання розв'язку операторного рівняння з заданою точністю. Одразу ж відзначимо, що і у цьому підрозділі ми будемо спиратись на матеріали, викладені в роботах [4], [5], звідки і почерпнемо інформацію про оператори управління обчислювальним процесом.

І почнемо з нагадування про те, що всі команди програми виконуються послідовно в порядку їх запису, але цей порядок може бути примусово змінений за допомогою внесення в програму операторів циклу та умовних інструкцій.

Тож продовжимо нарощування цього матеріалу з викладення суті операторів циклу, яких в мові Python є лише два – оператор ***for,*** що в перекладі означає ***для,*** та оператор ***while,*** що в перекладі означає ***доки***.

For-цикл використовують, коли відомою є його довжина, а ***while-цикл*** використовують, коли невідомо, якої довжини він має бути.

Характерною особливістю оператора циклу ***for*** є те, що він працює або з функцією ***range(k),*** що в перекладі означає ***діапазон цілих чисел від 0 до k-1,*** або з функцією ***list( ),*** що в ***перекладі означає список атрибутів, поміщених в аргументні дужки, при цьому функція range(k)*** розміщується в одному рядку з командою, що містить в собі оператор ***for*** , але після нього, а функцією ***list( ) список*** атрибутів задається в рядку, що передує рядку з командою, яка містить в собі оператор ***for.*** І цикл повторюватиметься стільки разів, скільки чисел визначено в діапазоні чи скільки атрибутів списку поміщено в аргументні дужки. Варіанти застосування оператора циклу ***for*** наведені в пояснювальному прикладі № 6.

Характерною особливістю оператора циклу ***while*** є те, що в рядку з командою, що містить в собі оператор циклу ***while,*** обов'язково поміщується умова, виконання якої дозволяє продовжувати бути в циклі змінній, початкове значення якої задається в рядку, що передує рядку з командою, яка містить в собі оператор циклу ***while.*** Варіанти застосування оператора циклу ***while*** наведені в пояснювальному прикладі № 7.

А тепер розкриємо суть «умовних інструкцій»: 1) інструкція ***if*** , яка містить в собі ще й приховане ***then***, тобто фактично є інструкцією ***if-then***, що в перекладі означає ***якщо _, то _***, а за змістом вимагає виконання команди, яка йде другою, якщо умова, що задана першою командою, виконується; 2) інструкція ***if-else***, яка містить в собі ще й приховане ***then***, тобто фактично є інструкцією ***if-then-else***, що в перекладі означає ***якщо _, то _, інакше _***, а за змістом вимагає виконання команди, яка йде другою, якщо умова, що задана першою командою, виконується, та вимагає виконання команди, яка йде третьою, якщо умова задана першою командою, не виконується; 3) інструкція ***if-elif-else***, яка містить в собі ще й двічі приховане ***then***, тобто фактично є інструкцією ***if-then-elif-then-else***, що в перекладі означає ***якщо _, то _, або якщо _, то _, інакше _***, а за змістом вимагає виконання команди, яка йде другою, якщо умова, що задана першою командою, виконується, вимагає виконання команди, яка йде четвертою, якщо умова, що задана першою командою, не виконується, але виконується умова, що задана третьою командою, та вимагає виконання команди, яка йде п'ятою, якщо не виконуються умови, що задані першою та третьою командами. Як працюють ці умовні інструкції продемонстровано в пояснювальному прикладі № 8.

А на завершення матеріалу, присвяченого циклам та умовним інструкціям, звертаємо увагу на те, що в тілі кожного з циклів (як з оператором ***for,*** так і з оператором ***while***) можуть міститись як умовні інструкції, так і оператори ***break*** та ***continue,*** перший з яких завершує цикл достроково і передає право виконуватись команді, яка розміщена одразу після завершальної команди циклу, а другий дозволяє виконувати наступну ітерацію циклу до завершення попередньої, що продемонстровано в пояснювальному прикладі № 9.

Пояснювальний приклад № 6

```
In [1]: import numpy                    # Виклик ППП numpy
In [2]: from numpy import*              # Доступ до усіх функцій numpy
In [3]: for i in range(3):              # Для i з діапазону від 0 до 2
            print(i+i**2)               ….друкувати i+i^2 в стовпчик
```

```
0
2
6
In [4]: for i in range(3):                  # Для i з діапазону від 0 до 2
            print(i+i**2,end=" ")           ….друкувати i+i^2 в рядок

0 2 6
In [5]: for x in range(3):                  # Для x з діапазону від 0 до 2
            print('Ура!')                   ….друкувати Ура! в стовпчик

Ура!
Ура!
Ура!
In [6]: for x in range(3):                  # Для x з діапазону від 0 до 2
            print('Ура!',end=" ")           ….друкувати Ура! в рядок

Ура!Ура!Ура!
In [7]: L=[2,3,4,5]                         # Внесення списку L
In [8]: for x in L:                         # Для x зі списку L
            print(x**3)                     ….друкувати x^3 в стовпчик

8
27
64
125
In [9]: for x in L:                         # Для x зі списку L
            print(x**3,end=" ")             ….друкувати x^3 в рядок

82764125
```

Кінець пояснювального прикладу № 6

Пояснювальний приклад № 7

```
In [1]: import numpy                        # Виклик ППП numpy
In [2]: for numpy import*                   # Доступ до усіх функцій numpy
In [3]: i=0                                 # Внесення початкових даних
In [4]: while i<5:                          # Формування циклу «доки i<5
            i=i+1                           ….до значення i додавати 1
            print(i**2)                     ….і друкувати значення i^2 в
                                            ….стовпчик»

1
4
9
16
25
```

```
In [6]: while i<5:
            i=i+1
            print(i**3,end=" ")
```

Формування циклу «доки i<5
….до значення *i* додавати 1
….і друкувати значення i^3 в
….рядок»

```
1491625
In [7]: i=3
In [8]: q=5
In [9]: while i<6 and q<8:
            i=i+1
            q=q+1
            print(i,q)
```

Внесення початкових даних
….у вигляді параметрів ***I*** та ***q***
Формування циклу «доки **i<6**
….і ***q<8*** до значень ***I,q*** додавати
….1 і друкувати значення
….пар ***(I,q)*** в стовпчик»

```
4 6
5 7
6 8
In [10]: i=4;q=7
In [11]: while i<6 or q<8:
             i=i+1

             q=q+1
             print(i,q)
```

Внесення початкових пар ***I,q***
Формування циклу «доки **i<6**
….або ***q<8*** до значень ***I,q***
….додавати одиницю
….і друкувати значення
….пар ***(I,q)*** в стовпчик»

```
5 8
6 9
In [12]: i=5;q=10
In [13]: while i<7 or q<15:

             i=i+1
             q=q+1
             print((i,q),end=" ")
```

Внесення початкових пар ***I,q***
Формування циклу «доки **i<7**
….або ***q<15*** до значень ***I,q***
….додавати одиницю
….і друкувати значення
…. пар ***(I,q)*** в рядок»

```
(6, 11) (7, 12) (8, 13) (9, 14) (10, 15)
In [14]: while i<7 and q<15:

             i=i+1
             q=q+1
             print((i,q),end=" ")
```

Демонстрація на прикладі
….того, що без внесення
…. початкових даних,
….реалізація while-циклу
…. не відбудеться

```
In [15}: i=5;q=10
In [16]: while i<7 and q<15:

             i=i+1
             q=q+1
             print((i,q),end=" ")
```

Внесення початкових пар ***I,q***
Формування циклу «доки **i<7**
….і ***q<15*** до значень ***I,q***
…. додавати одиницю
….і друкувати значення
…. пар ***(I,q)*** в рядок»

```
(6, 11) (7, 12)
In [17]: L=[5,4,3,-1,3,3,3]                    # Внесення списку
In [18]: i=6                                   # Внесення початкового індексу
In [18]: while i:                              # Формування циклу «доки i>0
             print(L[i],end=" ")               ….друкувати список в
                                               ….зворотному порядку»,
             i-=1                              ….на що вказує умова i-=1»,
                                               ….але при цьому втрачається
                                               ….перший член списку,
3 3 3 -1 3 4                                   ….внесеного на початку
```

Кінець пояснювального прикладу № 7

Пояснювальний приклад № 8

```
In [1]: import numpy                           # Виклик ППП numpy
In [2]: i=0                                    # Внесення початкового індексу
In [3]: s1='ліфт працює'                       # Внесення першої стрічки умов
In [4]: s2='ліфт не працює'                    # Внесення другої стрічки умов
In [5]: s3='йду пішки'                         # Внесення умови на виході
In [6]: for поверх in range(9):                # Формування циклу «для
                                               ….поверху із 9,
             if поверх <5 and s1:              ….якщо поверх <5 і ліфт
                поверх=поверх+1                ….працює,оглядаючи поверхи,
                print('поверх %s=%s' %(поверх, s3))  ….приймаю рішення, що
                                               ……..
поверх 1=йду пішки                             ….і на 1-ий поверх йду пішки,
поверх 2=йду пішки                             ….і на 2-ий поверх йду пішки,
поверх 3=йду пішки                             ….і на 3-ій поверх йду пішки,
поверх 4=йду пішки                             ….і на 4-ий поверх йду пішки,
поверх 5=йду пішки                             ….і на 5-ий поверх йду пішки
In [7]: for поверх in range(9):                # Формування циклу «для
                                               ….поверху із 9,
             if поверх <5 and s1:              ….якщо поверх <5 і ліфт
                поверх=поверх+1                …. працює,оглядаючи поверхи,
                print('поверх %s=%s' %(поверх, s3))  ….приймаю попереднє рішення,
             elif поверх >5 and s2:            ….або, якщо поверх >5 і ліфт
                                               ….не працює,
                поверх=поверх+1                ….оглядаючи поверхи,
                print('поверх %s=%s' %(поверх, s3))  ….приймаю рішення, що

поверх 1=йду пішки                             ….і на 1-ий поверх йду пішки,
поверх 2=йду пішки                             ….і на 2-ий поверх йду пішки,
поверх 3=йду пішки                             ….і на 3-ій поверх йду пішки,
поверх 4=йду пішки                             ….і на 4-ий поверх йду пішки,
поверх 5=йду пішки                             ….і на 5-ий поверх йду пішки,
поверх 6=йду пішки                             ….і на 6-ий поверх йду пішки,
поверх 7=йду пішки                             ….і на 7-ий поверх йду пішки,
```

```
поверх 8=йду пішки                                  ….і на 8-ий поверх йду пішки,
поверх 9 =йду пішки                                 ….і на 9-ий поверх йду пішки
```

Кінець пояснювального прикладу № 8

Пояснювальний приклад № 9

```
In [1]: import numpy                                # Виклик ППП numpy
In [2]: s1='втомився'                               # Внесення першої стрічки умов
In [3]: s2='пішов дощ'                              # Внесення другої стрічки умов
In [4]: відстань=0                                  # Внесення початку відліку
In [5]: while відстань < 15:                        # Формування циклу «доки
                                                    ….відстань <15
            print(відстань, end=" ")                ….друкувати її значення в рядок,
            if відстань==5 and s1:                  ….а, якщо відстань дорівнює 5 і
                                                    ….«втомився», то завершити
               break                                …. цикл достроково,
            elif відстань==10 and s2:               ….або, якщо відстань дорівнює
                                                    ….10 і «пішов дощ»,
               break                                ….то теж завершити цикл,
            else:                                   ….інакше
               відстань=відстань+1                  ….до змінної «відстань»
                                                    ….додавати одиницю»

0 1 2 3 4 5                                         # Результат руху за таких умов
In [6]: while відстань < 15:                        # Формування циклу «доки
                                                    ….відстань <15 друкувати її
            print(відстань, end=" ")                ….значення в рядок,
            if відстань==10 and s1:                 ….а, якщо відстань дорівнює 10 і
                                                    ….«втомився», то завершити
               break                                ……..цикл достроково,
            elif відстань==5 and s2:                ….або, якщо відстань дорівнює 5
                                                    ….і «пішов дощ»,
               break                                ….то теж завершити цикл,
            else:                                   ….інакше
               відстань=відстань+1                  ….до змінної «відстань»
                                                    ….додавати одиницю»

5                                                   # Результат руху за таких умов
In [7]: while відстань < 15:                        # Формування циклу «доки
                                                    ….відстань <15 друкувати її
            print(відстань, end=" ")                …. значення в рядок,
            if відстань==10 and s1:                 ….а, якщо відстань дорівнює 10 і
                                                    ….«втомився», то завершити
               break                                …. цикл достроково,
            elif відстань==5 and s2:                ….або, якщо відстань дорівнює 5
                                                    ….і «пішов дощ»,
               break                                ….то теж завершити цикл,
            else:                                   ….інакше
               відстань=відстань+1                  ….до змінної «відстань»
```

```
                                                    ….додавати одиницю»
5 6 7 8 9 10                                        # Результат руху за таких умов
In [8]: for відстань in range(15):                  # Формування циклу «для
                                                    ….відстані із [0,14],
        if 5 < відстань < 10:                       ….якщо відстань більша 5
                                                    ….або менша 10,
          continue                                  ….перейти до наступної операції
                                                    ….в циклі,
        print(відстань,end=" ")                     ….пропустивши проміжні»

0 1 2 3 4 5 10 11 12 13 14                          # Результат руху за таких умов
```

Кінець пояснювального прикладу № 9

7.3 Задачі з розв'язання операторних рівнянь методом стиснених відображень в програмах мовою Python

Програма мовою Python для розв'язання алгебраїчного рівняння $F(x) = 0$ у варіанті $ax^2 + bx + c = 0$ **, заданого на відрізку значень $x \in [e, h]$ незалежної змінної методом стиснених відображень**

(Програма 23 для розв'язання наведеного вище алгебраїчного рівняння після подання його у вигляді $x = f(x)$, тобто у варіанті $x = -\frac{ax^2}{b} - c/b$**)**

```
In [1]: import numpy as np                          # Виклик ППП numpy як np
In [2]: import sympy as sm                          # Виклик ППП sympy як sm
In [3]: a = 1.0                                     # Внесення значень
In [4]: b = -6.0                                    ….коефіцієнтів
In [5]: c = 8.75                                    ….рівняння
In [6]: po = 0.05                                   # Внесення значення похибки
In [7]: e = 0                                       # Внесення границь
In [8]: h = 2                                       ….незалежної змінної
In [9]: N = 21                                      # Внесення кількості точок
In [10]: x, y = sm.symbols('x y')                   # Оголошення символьними x, y
In [11]: fx = -a*x**2/b-c/b                         # Подання рівняння як x = f(x)
In [12]: fy = -a*y**2/b-c/b                         # Подання рівняння як y = f(y)

In [13]: expr1 = -a*x**2/b-c/b                      # Подання функцій f(x),
In [14]: expr2 = -a*y**2/b-c/b                      ….f(y) у вигляді expr1, expr2
In [15]: f0 = sm.lambdify(x, expr1, "numpy")        # Конвертація функцій f(x), f(y)
In [16]: f00 = sm.lambdify(y, expr2, "numpy")       ….у numpy
In [17]: x = np.linspace(e, h, N)                   # Формування масиву x
In [18]: print(f"x: {x}")                           # Друкування масиву значень x
x: [0. 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.
 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2. ]
In [19]: f1 = f0(x)                                 # Формування і друкування
```

```
In [20]: print(f"f1: {f1}")
 f1: [1.45833333 1.46       1.465
1.47333333 1.485      1.5
 1.51833333 1.54       1.565
1.59333333 1.625      1.66
 1.69833333 1.74       1.785
1.83333333 1.885      1.94
 1.99833333 2.06       2.125     ]
In [21]: f11 = f1.min()                                  # Визначення мінімального та
In [22]: f12 = f1.max()                                  ….максимального значень
                                                         ….функції f(x)
In [23]: print(f"f11, f12: {f11, f12}")                  ….та їх друкування
f11, f12: (1.45833333333333,
 2.124999999999998)
In [24]: e1 = float(e); h1 = float(h)                    # Трансформація чисел
In [25]: if e1 < f11 and f12 < h1:                       # Формування 1-ї умови
                                                         ….«якщо _,то _»
             print("1st if")                             # Друкування 1-ї умови
             x[0] = 1.0                                  # Внесення початкового
                                                         ….значення кореня
             for i in range(N):                          # Формування 1-го циклу
                                                         ….«для i з (N)»
                print(f"i: {i}")                        # Друкування « i з (N)»
                x[i+1] = -a*(x[i])**2/b-c/b              # Формування рівняння
                                                         ….наближень
                xx = x[i+1]-x[i]                         # Формування різниці
                                                         ….сусідніх значень
                axx = abs(xx)                            # Формування модуля різниці
                if axx < po:                             # Формування 2-ї умови
                                                         ….«якщо _, то _»
                   print(x[i+1])                         # Друкування кореня при
                                                         ….виконанні 2-ї умови
                   break                                 # Вихід із 1-го циклу
                else:                                    # Формування 2-ї частини
                   i += 1                                ….2 умови «якщо _, то _, інакше»
                   print('1st else')                     ….та друкування її результату
         else:                                           # Формування 2-ї частини 1-ї
                                                         ….умови
             for j in range(1,4):                        # Формування 2-го циклу
                                                         ….«для j з [1,4)»
                print(f"j: {j}")                        # Друкування « j з [1,4)»
                e2 = e-j                                 # Визначення нових меж
                h2 = h+j                                 ….відрізка значень змінної x
                N1=(h1-e1)/0.1+1                         # Формування нового масиву
                N2=int(N1)                               ….значень змінної x та її
                y = np.linspace(e2, h2, N2)              ….форматування під новим
```

```
                f2 = f00(y)                                  ….символом y в діапазоні (N2)
                f21 = f2.min( )                              # Формування масиву f(y)
                f22 = f2.max( )                              # Визначення мінімуму та
                e3 = float(e2); h3 = float(h2)               ….максимуму функції f(y)
                                                             # Перетворення цілих чисел в
                if e3 < f21 and f22 < h3:                    ….дійсні
                                                             # Формування 3-ї умови
                   print("2nd if")                           ….«якщо _, то _»
                                                             # Друкування 3-ї умови
                   y[0] = 1.0                                ….«якщо _, то _»
                                                             # Внесення початкового
                   for k in range(N2):                       ….значення кореня
                                                             # Формування 3-го циклу
                      print(f"k: {k}")                       ….«для k із (N2)»
                      y[k+1] = -a*(y[k])**2/b-c/b            # Друкування «k з (N2)»
                                                             # Формування рівняння
                      yy = y[k+1]-y[k]                       ….наближень
                                                             # Формування різниці
                      ayy = abs(yy)                          ….сусідніх значень
                      print(f"ayy: {ayy}")                   # Формування модуля різниці
                      if ayy < po:                           # Друкування модуля різниці
                                                             # Формування 3-ї умови
                         print(f"y[k+1]: {y[k+1]}")          ….«якщо _, то _»
                                                             # Друкування наближеного
                         break                               ….значення кореня
                      else:                                  # Вихід із 2-го циклу
                         k += 1                              # Формування 2-ї частини 3-ї
                         print('2nd else')                   ….умови «якщо _, то _, інакше _»
                                                             ….та друкування її результату
j: 1                                                         # Масив видрукованого
2nd if
k: 0
ayy: 0.625
2nd else
k: 1
ayy: 0.2734375
2nd else
k: 2
ayy: 0.16057332356770804
2nd else
k: 3
ayy: 0.1059101050271205
2nd else
k: 4
ayy: 0.07455950924982035
```

```
2nd else
k: 5
ayy: 0.054731667403494555
2nd else
k: 6
ayy: 0.041356092063042915
y[k+1]: 2.3355681973111864
j: 2
j: 3
```

Кінець програми 23

Програма мовою Python для розв'язання інтегрального рівняння Фредгольма 2-го роду

$$\boldsymbol{\varphi(x) = f(x) + \lambda \int_a^b K(x,y)\varphi(y)dy},$$

в якому: $a = 0,\ b = \pi,\ f(x) = \sin(x),\ K(x,y) = xe^{-y},\quad \varepsilon = 0.01$, **методом стиснених відображень**

(Програма 24)

```
In [1]: import numpy as np                          # Виклик ППП numpy як np
In [2]: import sympy as smp                         # Виклик ППП sympy як smp
In [3]: a = 0                                       # Внесення значень
In [4]: b = 3                                       меж інтегрування
In [5]: po = 0.05                                   # Внесення значення похибки
                                                    ….результату
In [6]: N = 31                                      # Внесення кількості точок на
                                                    ….відрізку [a,b]
In [7]: x, y = smp.symbols('x y')                   # Оголошення символьними x,y
In [8]: fxy = x*smp.exp(-y)                         # Внесення ядра оператора
                                                    ….Фредгольма і підготовка його
In [9]: expr=fxy                                    ….до конвертації в numpy
In [10]: f0= smp.lambdify((x,y),expr,"numpy")       # Конвертація в numpy ядра
                                                    ….оператора
In [11]: x = np.linspace(a,b,N)                     # Формування масиву x
In [12]: y = np.linspace(a,b,N)                     # Формування масиву y
In [13]: f1 = f0(x,y)                               # Формування і друкування
In [14]: print(f"f1: {f1}")                         ….масиву значень ядра
f1: [0.         0.09048374 0.16374615 0.22224547
 0.26812802 0.30326533 0.32928698 0.34760971
0.35946317 0.36591269 0.36787944 0.36615819
 0.36143305 0.35429133 0.34523575 0.33469524
 0.32303443 0.31056199 0.297538   0.28418038
0.27067057 0.2571585  0.24376695 0.23059534
 0.21772309 0.2052125  0.1931113  0.18145488
 0.17026818 0.159567340.14936121]
```

```
In [15]: f11 = f1.max()
In [16]: print(f"f11: {f11}")
f11: 0.36787944117144233
In [17]: q = 1/(f11*(b-a))
!n [18]: q1 = q - 0.5

In [19]: q1=q1.round(1)
In [20]: print(f"q1: {q1}")
q1: 0.4
In [21]: x, y = smp.symbols('x y')

In [22]: g = smp.Function('g')(x)
In [23]:h = smp.Function('h')(x)
In [24]: f2 = x**2

In [25]: g = f2

In [26]: gg = []

In [27]: gg.append(g)

In [28]: for i in range(20):

             print(f"i: {i}")

             g = f2 + smp.integrate(q1*x*
           smp.exp(-y)*(gg[i]).subs(x,y),(y,0,3))

             gg.append(g)

             h = gg[i+1] - gg[i]

             mh = smp.integrate(h**2,(x,0,3)).n(3)
             print(f"mh: {mh}, {type(mh)}")
             nh = mh**0.5
             print(f"nh: {nh}, {type(nh)}")

             if nh > po:
                i += 1

             else:
                print(f"gg[i+1]: {gg[i+1].n(3)}")

                break
```

```
# Визначення норми ядра
….оператора Фредгольма
….та її друкування
# Визначення верхньої межі
….параметра рівняння
….Фредгольма, його
….приведення до умов
….методу стиснених
….відображень та друкування
# Повернення змінних до
….символьних
# Символізація функції g
# Символізація функції h
# Внесення функції
….зовнішнього впливу
….та її оголошення розв’язком
….0-го порядку
# Формування пустого списку
….розв’язків
# Внесення в список розв’язку
….0-го порядку
# Формування for-циклу
….наближень
# Друкування номера
….наближення
# Основна формула для
….обчислення і-го наближення
….розв’язку
# Внесення і-го наближення
….в список
# Формування різниці сусідніх
….наближень
# Інтегрування квадрата різниці
….наближень та його друкування
# Формування метрики
….між наближеннями та її
….друкування
# Перевірка умови з точності
….і формування наступного
….кроку в циклі
# Умова завершення циклу та
….друк кінцевого результату
….наближень
# Команда на завершення
….циклу і перехід до наступної
….команди програми
```

```
i: 0                                                    # Друк таблиці результатів,
mh: 1.92, <class 'sympy.core.numbers.Float'>            ….отриманих в циклі
nh: 1.38437852708354, <class
'sympy.core.numbers.Float'>
i: 1
mh: 0.197, <class 'sympy.core.numbers.Float'>
nh: 0.443458310822449, <class
'sympy.core.numbers.Float'>
i: 2
mh: 0.0202, <class 'sympy.core.numbers.Float'>
nh: 0.142055441765376, <class
'sympy.core.numbers.Float'>
i: 3
mh: 0.00207, <class 'sympy.core.numbers.Float'>
nh: 0.0455071838073526, <class
'sympy.core.numbers.Float'>
gg[i+1]: x**2 + 0.672*x                                 # Наближений розв’язок
                                                        ….інтегрального рівняння
                                                        ….Фредгольма з заданою
                                                        ….похибкою
```

Кінець програми 24.

Розділ 8 СПЕЦІАЛЬНІ ОПЕРАТОРИ З КЛАСУ НЕПЕРЕРВНИХ (В ПРИКЛАДАХ І ПРОГРАМАХ)

8.1 Прямий та обернений оператори Лапласа

У кінці восьмого підрозділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. Що собою являє оператор Лапласа?
2. Які ви знаєте властивості оператора Лапласа?
3. Які головні переваги аналізу в області зображень?
4. Що собою являють відображення за Лапласом похідної від неперервної функції та інтегралу від неї?
5. Як визначити передаточну функцію системи, якщо відоме диференціальне рівняння, котре описує процеси у цій системі?.
6. Як, маючи передаточну функцію лінійної динамічної системи, відтворити диференціальне рівняння, яке описує процеси у цій системі?
7. Що собою являє обернений оператор Лапласа? Які форми його реалізації ви знаєте?
8. Як визначити оригінал за відомим його зображенням?

Тож із відповідей на ці питання ми і розпочнемо викладення змісту першого підрозділу восьмого розділу другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих питань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2]. Отже, почнемо викладення матеріалу цього підрозділу з розлогої цитати, взятої з нашого базового навчального посібника [2], в якій змінена лише нумерація формул та пропущені деякі проміжні викладки, що робить цей матеріал не стільки цитатою, скільки її вільним переказом.

З загального курсу вищої математики відомо, що **за допомогою оператора Лапласа**

$$L\{f(t)\} = \int_0^\infty f(t)e^{-pt}dt \tag{8.1}$$

кожній неперервній функції *f* ***часу*** *t*, ***заданій на множині дійсних чисел, яка задовольняє умову*** $f(t) = 0$ ***при*** $t < 0$, ***умови Діріхле і називається оригіналом, можна поставити у відповідність функцію*** *F* ***комплексної змінної*** $p = \sigma + j\omega$, ***яка називається зображенням оригіналу на комплексній площині. Ця відповідність записується у такий спосіб:***

$$F(p) \Leftrightarrow f(t). \tag{8.2}$$

Нагадаємо, що оператор Лапласа належить до класу лінійних, а тому для нього є справедливими співвідношення

$$L\{f_1(t) + f_2(t)\} = L\{f_1(t)\} + L\{f_2(t)\}, \tag{8.3}$$

$$L\{a\, f(t)\} = a\, L\{f(t)\}, \tag{8.4}$$

де a – скаляр.

Головною перевагою аналізу в області зображень *F*(*p*), ***тобто на комплексній площині, порівняно з аналізом в області оригіналів*** *f*(*t*), ***тобто у часовому просторі, є те, що за нульових початкових умов операції диференціювання*** $\frac{d}{dt}$ ***оригіналу*** *f*(*t*) ***у часовому***

просторі відповідає операція перемноження на комплексну змінну p ***його зображення*** $F(p)$ ***на комплексній площині***, тобто

$$\frac{d}{dt}f(t) \Leftrightarrow pF(p), \tag{8.5}$$

$$\frac{d^2}{dt^2}f(t) \Leftrightarrow p^2F(p), \tag{8.6}$$

$$\frac{d^n}{dt^n}f(t) \Leftrightarrow p^nF(p), \tag{8.7}$$

а операції інтегрування оригіналу $f(t)$ ***у часовому просторі відповідає операція ділення на комплексну змінну*** p ***його зображення*** $F(p)$ ***на комплексній площині***, тобто

$$\int_0^t f(\tau)d\tau \Leftrightarrow \frac{F(p)}{p}, \tag{8.8}$$

$$\int_0^t \dots \int_0^t f(\tau_1, \tau_2, \dots, \tau_n)d\tau_1 d\tau_2 \dots d\tau_n \Leftrightarrow \frac{F(p)}{p^n} \tag{8.9}$$

Завдяки властивостям (8.3), (8.4), (8.5) та їх наслідкам (8.7), (8.9) диференціальним та інтегральним рівнянням, записаним у часовому просторі, відповідають алгебраїчні рівняння на комплексній площині, розв'язувати які набагато простіше, оскільки цьому вчать ще у школі.

Наприклад, диференціальному рівнянню в області оригіналів $x(t)$, $y(t)$

$$a_2\frac{d^2y}{dt^2} + a_1\frac{dy}{dt} + a_0y = b_1\frac{dx}{dt} + b_0x, \tag{8.10}$$

використовуючи оператор Лапласа (8.1) та умови (8.3)–(8.6), на комплексній площині можна поставити у відповідність алгебраїчне рівняння

$$a_2p^2Y(p) + a_1pY(p) + a_0Y(p) = b_1pX(p) + b_0X(p), \tag{8.11}$$

з якого випливає, що розв'язок $y(t)$ диференціального рівняння (8.10) на комплексній площині можна отримати у вигляді

$$Y(p) = \frac{b_1p+b_0}{a_2p^2+a_1p+a_0}X(p), \tag{8.12}$$

або у вигляді

$$Y(p) = W(p)X(p), \tag{8.13}$$

де

$$W(p) = \frac{b_1p+b_0}{a_2p^2+a_1p+a_0} \tag{8.14}$$

передаточна функція динамічної системи, процеси в якій описуються диференціальним рівнянням (8.10), яка несе в собі стільки ж інформації про динамічний об'єкт, скільки і це диференціальне рівняння, оскільки в її структуру входять усі його коефіцієнти.

З рівняння (8.13) легко бачити, що, якщо відома передаточна функція $W(p)$ динамічної системи і перетворений за допомогою оператора Лапласа її вхідний сигнал X(p), то, скориставшись цим рівнянням, можна знайти перетворену за Лапласом реакцію $Y(p)$ цієї системи на даний вхідний сигнал, а якщо відомими є перетворені за Лапласом її вхідний $X(p)$ та вихідний $Y(p)$ сигнали, що отримані, наприклад, експериментально та апроксимовані відповідними математичними виразами, то, скориставшись рівнянням (8.13) у вигляді

$$W(p) = \frac{Y(p)}{X(p)}, \tag{8.15}$$

можна знайти передаточну функцію $W(p)$ цієї динамічної системи, підставляючи яку у рівняння (8.13) та переносячи її знаменник в ліву частину цього рівняння, отримаємо аналог рівняння (8.11) на комплексній площині для цієї динамічної системи, від якого перехід до диференціального рівняння здійснюється формальною заміною добутків степенів k-го порядку $(k = 0,\ 1,\ 2, \dots, n)$ комплексної змінної p на зображення за Лапласом вихідного $Y(p)$ та вхідного X(p) сигналів, тобто формальною заміною $p^k Y(p),\ p^k X(p)$ на похідні того ж порядку, тобто на $\frac{d^k y}{dt^k},\ \frac{d^k x}{dt^k}$.

А щоб від знайденого за виразом (8.13) зображення $Y(p)$ вихідного сигналу на комплексній площині перейти до його аналогу $y(t)$ на дійсній осі часу потрібно скористатись оберненим оператором Лапласа, загальним виглядом якого є вираз

$$y(t) = L^{-1}\{Y(p)\} = \frac{1}{2\pi j}\int_{\sigma-j\infty}^{\sigma+j\infty} Y(p)e^{pt}dp, \tag{8.16}$$

згідно з яким за відомим зображенням *Y*(*p*) ***можна знайти оригінал*** *y*(*t*)**,** який, як правило, використовується лише для побудови таблиць відповідності між *y*(*t*) та *Y*(*p*), а в практиці аналізу частіше використовуються формули розкладання, отримані математиками шляхом застосування теореми лишків при інтегруванні у виразі (8.16), одна з яких – для некратних полюсів p_i зображення $Y(p)$ при поданні його у вигляді

$$Y(p) = \frac{C(p)}{D(p)}, \tag{8.17}$$

де *C(p), D(p)*– багаточлени за степенями *p* порядків *m* та *n*, відповідно, має вигляд

$$y(t) = \sum_{i=1}^{n} \frac{C(p_i)}{D'(p_i)}\ e^{p_i t}, \tag{8.18}$$

де p_i – це корені рівняння

$$D(p) = 0, \tag{8.19}$$

які називають полюсами виразу (8.17), а

$$D'(p_i) = \frac{dD}{dp}(p = p_i). \tag{8.20}$$

В разі ж, якщо зображення $Y(p)$ при поданні його у вигляді (8.17) має один полюс кратності k та *n-1* некратних полюсів, то формула розкладання для визначення $y(t)$ набуває вигляду

$$y(t) = \frac{1}{(k-1)!}\ \frac{d^{(k-1)}}{dp^{(k-1)}}\left(\frac{(p-p_1)^k C(p)}{D(p)}e^{pt}\right)_{p=p_1} + \sum_{i=2}^{n-1}\frac{C(p_i)}{D'(p_i)}\ e^{p_i t}. \tag{8.21}$$

Цілком очевидно, що в разі, якщо кратних полюсів буде декілька, то стільки ж у цій формулі буде складових, на кшталт виписаної у виразі (8.21) першою, а у другій складовій виразу (8.21) сума зменшиться на кількість цих кратних полюсів.

Для того, щоб до кінця зрозуміти як працюють формули розкладання (8.18), (8.21), розглянемо конкретний приклад.

Нехай передаточна функція динамічної системи має вигляд

$$W(p) = \frac{2p+1}{p^2+5p+6}, \tag{8.22}$$

і нехай на її вхід подається вхідний сигнал $x(t)$ у вигляді одиничної сходинки, тобто у вигляді

$$x(t) = 1(t), \tag{8.23}$$

де

$$1(t) = \begin{cases} 1 \text{ для } \forall t \in [0, \infty), \\ 0 \text{ для } \forall t < 0 \end{cases} \tag{8.24}$$

І треба віднайти реакцію $y(t)$ даної динімічної системи на цей вхідний сигнал.

Цілком очевидно, що для того, щоб знайти зображення $Y(p)$ цієї реакції на комплексній площині, потрібно у формулу (8.13) замість абстрактної передаточної функції $W(p)$ підставити її конкретне значення, взяте з виразу (8.22), а також підставити і зображення $X(p)$ за Лапласом вхідного сигналу $x(t)$, конкретизованого виразом (8.23). А тому спочатку, скориставшись формулою (8.1), віднайдемо $X(p)$.

Підставляючи вираз (8.23) в оператор Лапласа (8.1), матимемо

$$X(p) = L\{x(t)\} = \int_0^\infty 1(t)e^{-pt}dt = \int_0^\infty e^{-pt}dt = \frac{1}{(-p)}e^{-pt}\Big|_0^\infty = \frac{1}{p} \tag{8.25}$$

А підставляючи вирази (8.22) та (8.25) у вираз (8.13), для нашої задачі отримаємо

$$Y(p) = \frac{2p+1}{p^3+5p^2+6p} = \frac{C(p)}{D(p)}, \tag{8.26}$$

звідки

$$C(p) = 2p + 1 \tag{8.27}$$

$$D(p) = p^3 + 5p^2 + 6p \tag{8.28}$$

Полюси зображення (8.26) знайдемо з рівняння (8.19), яке для нашої задачі набуває вигляду

$$p^3 + 5p^2 + 6p = 0. \tag{8.29}$$

Легко бачити, що коренями цього рівняння, а відповідно і полюсами зображення (8.26), будуть числа:

$$p_1 = 0; \quad p_2 = -2; \quad p_3 = -3, \tag{8.30}$$

які є різними і серед яких немає кратних, а тому для визначення оригіналу $y(t)$ будемо використовувати формулу розкладання у вигляді (8.18) та $D'(p)$ на основі виразу (8.28) у вигляді

$$D'(p) = \frac{dD}{dp} = 3p^2 + 10p + 6. \tag{8.31}$$

Для умов (8.30) вираз (8.18) набуває вигляду

$$y(t) = \frac{C(p_1)}{D'(p_1)}e^{p_1 t} + \frac{C(p_2)}{D'(p_2)}e^{p_2 t} + \frac{C(p_3)}{D'(p_3)}e^{p_3 t}. \tag{8.32}$$

Підставляючи у вираз (8.32) вирази (8.27), (8.31) і (8.30) та виконуючи відповідні алгебраїчні обчислення, отримаємо

$$y(t) = \frac{1}{6} + \frac{3}{2}e^{-2t} - \frac{5}{3}e^{-3t}. \tag{8.33}$$

А тепер визначимо реакцію цієї ж динамічної системи з передаточною функцією (8.22) на лінійно-наростаючий вхідний сигнал $x(t)$ у вигляді

$$x(t) = t\,1(t), \tag{8.34}$$

в моделі (8.34) якого одинична сходинкова функція $1(t)$ використана для того, щоб обнулити цей сигнал при від'ємних значеннях аргументу.

Підставляючи вираз (8.34) у формулу (8.1) оператора Лапласа та застосовуючи спосіб інтегрування частинами, знайдемо, що у цьому випадку

$$X(p) = L\{x(t)\} = \int_0^\infty t\,e^{-pt}dt = \frac{1}{p^2}. \tag{8.35}$$

У свою чергу, підставляючи вирази (8.22) та (8.35) у вираз (8.13), для цього випадку матимемо

$$Y(p) = \frac{2p+1}{p^4+5p^3+6p^2} = \frac{C(p)}{D(p)}. \tag{8.36}$$

Аналізуючи вираз (8.36), бачимо, що вираз для $C(p)$ не змінився, тобто, він залишився у вигляді (8.27), а вираз для $D(p)$ став іншим і набув вигляду

$$D(p) = p^4 + 5p^3 + 6p^2, \tag{8.37}$$

а тому іншим став і вираз (8.31), який отримав у цьому випадку вигляд

$$D'(p) = \frac{dD}{dp} = 4p^3 + 15p^2 + 12p. \tag{8.38}$$

Іншим стало і рівняння (8.29), оскільки для цього випадку воно набуло вигляду

$$p^4 + 5p^3 + 6p^2 = 0. \tag{8.39}$$

І хоча коренів у цього рівняння теж залишилось три, але перший корінь, який дорівнює нулю, став кратним з кратністю $k = 2$, тобто, замість полюсів зображення (8.26) у вигляді (8.30) для зображення (8.36) полюси матимемо у вигляді

$$p_1 = \{0: k = 2\}\;\; p_2 = -2;\;\; p_3 = -3, \tag{8.40}$$

а тому для визначення оригіналу $y(t)$ у цьому випадку будемо використовувати формулу розкладання у вигляді (8.21), яка набуває вигляду

$$y(t) = \frac{d}{dp}\left(\frac{2p+1}{p^2+5p+6}e^{pt}\right)_{p=p_1} + \sum_{i=2}^{3}\frac{C(p_i)}{D'(p_i)}\,e^{p_i t}. \tag{8.41}$$

Диференціюючи вираз, що стоїть у дужках співвідношення (8.41) з урахуванням того, що у цьому виразі є і добуток функцій, і їх дріб, та, підставляючи в отриманий результат перший полюс із записаних у (8.40), а також підставляючи у другу структурну складову співвідношення (8.41) вираз (8.38) та другий і третій полюси з записаних у (8.40) і виконуючи відповідні алгебраїчні обчислення, отримаємо

$$y(t) = \frac{7}{36} - \frac{t}{6} - \frac{3}{4}e^{-2t} + \frac{5}{9}e^{-3t}. \tag{8.42}$$

8.2 Прямий та обернений оператори Фур'є

У кінці восьмого розділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. Що собою являють прямий і обернений оператори Фур'є?
2. Як пов'язані між собою прямі оператори Лапласа і Фур'є?
3. Яким чином, маючи передаточну функцію лінійної динамічної системи, отримати її частотні характеристики: дійсну, уявну, амплітудну, фазову, амплітудно-фазову?

Тож із відповідей на ці запитання ми і розпочнемо викладення змісту другого підрозділу восьмого розділу другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих питань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2]. Отже, почнемо викладення матеріалу цього підрозділу з розлогої цитати, взятої з нашого базового навчального посібника [2], в якій змінена лише нумерація формул та пропущені деякі проміжні викладки, що робить цей матеріал не стільки цитатою, скільки її вільним переказом.

Отже, як уже відзначено нами в нашому базовому навчальному посібнику [2], ***для аналізу частотного спектра сигналу, описаного функцією x(t), заданою у гільбертовому просторі, Фур'є запропонував використати оператор***

$$X(j\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt\,, \tag{8.43}$$

який множині функцій $\{x(t)\}$ дійсного аргументу t, для яких виконуються умови Діріхле та умова обмеженості енергії

$$\int_{-\infty}^{\infty} x^2(t)dt < \infty, \tag{8.44}$$

ставить у відповідність множину функцій $\{X(j\omega)\}$ уявного аргументу $j\omega$, в якому ω є круговою частотою, що пов'язана з основною частотою f періодичних коливань сигналу $x(t)$ співвідношенням

$$\omega = 2\pi f = \frac{2\pi}{T}, \tag{8.45}$$

де T – період коливання цього сигналу.

А для встановлення відповідності між множиною функцій {X(jω)}, отриманих за допомогою оператора* (8.43), *і множиною функцій {x(t)} Фур'є запропонував обернений оператор

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(j\omega)e^{j\omega t}d\omega, \tag{8.46}$$

який, незважаючи на його математичну красоту, при розв'язанні конкретних задач спектрального аналізу сигналів практичного застосування не знаходить, тому до нього в подальшому ми звертатись не будемо.

А далі звертаємо увагу на те, що, як показано у попередньому підрозділі, достатньо повну інформацію про динаміку лінійної системи несе її передаточна функція $W(p)$, яка є функцією комплексної змінної $p = \sigma + j\omega$.

Якщо накласти умову

$$\sigma = 0, \tag{8.47}$$

то передаточна функція $W(p)$ вироджується у функцію уявної частини $j\omega$ комплексної змінної p, тобто в $W(j\omega)$.

Для прикладу, нехай

$$W(p) = \frac{5}{2p+1}. \tag{8.48}$$

За виконання умови (8.47) із (8.48) маємо

$$W(j\omega) = \frac{5}{1+j2\omega} \tag{8.49}$$

Функцію $W(j\omega)$, визначену виразом (8.49), шляхом очевидних перетворень з використанням комплексного числа, спряженого з комплексним числом, що стоїть у її знаменнику, можна привести до такої форми

$$W(j\omega) = \frac{5}{1+j2\omega}\,\frac{1-j2\omega}{1-j2\omega} = \frac{5}{1+4\omega^2} + j\left(\frac{-10\omega}{1+4\omega^2}\right) = R(\omega) + jQ(\omega), \tag{8.50}$$

де

$$R(\omega) = \frac{5}{1+4\omega^2}, \tag{8.51}$$

$$Q(\omega) = -\frac{10\omega}{1+4\omega^2}. \tag{8.52}$$

Вирази (8.49)–(8.52) за своєю структурою справедливі для довільної функції $W(p)$. Оскільки як функція $W(p)$, так і її складові $R(\omega)$ та $Q(\omega)$ у своїй структурі містять всі коефіцієнти передаточної функції $W(p)$, то можна стверджувати, що кожна з них несе однакову кількість інформації про динаміку системи, яка характеризується цією передаточною функцією. У цьому легко переконатись на наведеному прикладі.

З теорії функцій комплексної змінної відомо, що такі функції можна подавати не лише в алгебраїчній формі, як показано у виразі (8.50), а й у показниковій, тобто у вигляді

$$W(j\omega) = A(\omega)e^{j\varphi(\omega)}, \tag{8.53}$$

де

$$A(\omega) = \sqrt{R^2(\omega) + Q^2(\omega)}, \tag{8.54}$$

$$\varphi(\omega) = arctg\,\frac{Q(\omega)}{R(\omega)} \tag{8.55}$$

$$R(\omega) = A(\omega)\cos\varphi(\omega), \tag{8.56}$$

$$Q(\omega) = A(\omega)\sin\varphi(\omega). \tag{8.57}$$

Оскільки параметр ω ***в усіх наведених вище виразах має чітко визначений фізичний сенс – це кругова частота, – то всі функції, що залежать від нього, називаються частотними характеристиками, з яких:***

$R(\omega)$ – ***дійсна частотна характеристика (ДЧХ),***

$Q(\omega)$ – ***уявна частотна характеристика (УЧХ),***

$\boldsymbol{A(\omega)}$ – ***амплітудна частотна характеристика (АЧХ),***

$\boldsymbol{\varphi(\omega)}$ – ***фазова частотна характеристика (ФЧХ),***

$W(j\omega)$ – ***амплітудно-фазова частотна характеристика (АФЧХ).***

А далі звертаємо увагу, по-перше, на те, що АЧХ характеризує ступінь згасання кожної частотної складової сигналу при його проходженні через динамічну систему з її входу на вихід, а ФЧХ характеризує зсув за фазою, який виникає при появі кожної частотної складової сигналу на виході динамічної системи відносно її появи на вході за рахунок часу її проходження через цю систему.

А, по-друге, звертаємо увагу на те що, ввівши умову (8.47), ми від перетворення за Лапласом, яке кожній неперервній функції, визначеній на часовій осі як оригінал, ставить у відповідність неперервну функцію на комплексній площині, фактично, переходимо до

перетворення Фур'є, яке відображає оригінали на модифіковану комплексну площину, оскільки реалізується не у вигляді (8.43), а (для АФЧХ) у вигляді

$$W(j\omega) = \int_0^{\infty} g(t)e^{-j\omega t}dt, \tag{8.58}$$

що цілком правомірно, оскільки імпульсна перехідна функція $g(t)$, як відомо, визначена лише для значень $t \in [0,\infty)$, а тому

$$\int_{-\infty}^{0} g(t)e^{-j\omega t}dt = 0. \tag{8.59}$$

А модифікація комплексної площини полягає у тому, що кожна її точка задається не просто парою дійсних чисел, як при відображенні за Лапласом, а парою дійсних чисел, кожне з яких є значенням функцій кругової частоти.

Підсумовуючи викладене вище, можемо констатувати, що прямий оператор Фур'є реалізовує закон відображення неперервних функцій дійсного аргументу, заданого на числовій осі, у множину неперервних функцій комплексного аргументу, заданого на комплексній площині, координати точок якої теж є дійсними числами, але, на відміну від тих чисел, які задаються прямим оператором Лапласа, ці дійсні числа конкретизуються значеннями неперервних функцій кругової частоти.

8.3 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з застосуванням операторів Лапласа

Аналізуючи характер формул, викладених у попередніх підрозділах 8.1, 8.2, бачимо, що для їх реалізації в програмах мовою Python нам будуть потрібні деякі нові знання функцій і методів цієї мови програмування, додаткові до тих, про які вже йшла мова в підрозділах, присвячених викладенню додаткових відомостей з технології програмування мовою Python у першій частині цього навчального посібника та у підрозділах 6.2, 7.2 другої його частини, які ми дублювати не будемо.

Продовжимо нарощування цього матеріалу з зауваження про те, що застосовувати оператор Лапласа для трансформації функції ***f*** дійсної змінної ***t***, тобто функції ***f = f(t),*** на комплексну площину у вигляді функції ***F*** комплексної змінної ***p,*** тобто функції ***F = F(p),*** потрібно з використанням символьного ППП ***sympy*** та операції інтегрування в ньому ***integrate*(*f*,(*t*,0,*oo*)**, в якій верхня нескінченна межа позначається подвійною латинською літерою ***o***, тобто записується у вигляді ***oo***. Якщо функція ***f*** виявиться занадто складною, ми можемо замість ***F*** після застосування функції ***integrate*(*f*,(*t*,0,*oo*)** отримати її необчислений еквівалент у вигляді функції ***Integral*( )**. У цьому випадку потрібно спочатку сформувати символьний вираз у вигляді ***expr = Integral*(*f, t*)**, до якого потім застосувати метод ***doit*( )** у вигляді ***expr.doit*( )**.

Як друге зауваження звертаємо увагу на те, що в разі, якщо потрібно перетворювати за Лапласом похідну $f_1 = f'(t)$ від складної функції ***f(t)***, то доцільно спочатку обчислити цю похідну, і може виявитись, що ні за допомогою функції ***diff*(*f, t*)**, ні за допомогою методу ***f.diff*(*t*)** цю похідну обчислити не вдається, а в результаті застосування методу диференціювання ми отримуємо лише вираз $f1 =$ ***Derivative*( )**. У цьому випадку для отримання прийнятного кінцевого результату ми теж маємо застосувати метод ***doit*( )** у вигляді $f1.doit()$.

Ну і третє зауваження: звертаємо увагу на те, що в разі, якщо після виконання в ППП ***sympy*** попередніх операцій або після застосування оберненого оператора Лапласа у вигляді теорем розкладання ми в результаті отримаємо вирази, що за коефіцієнти містять корені чи

правильні дроби або степені π чи **e,** то для їх подання у вигляді десяткових дробів з потрібною нам кількістю знаків ***n***, не обов'язково трансформувати кожний такий вираз в ППП ***numpy***, як це ми вже демонстрували раніше, а достатньо застосувати до нього метод ***evalf*(*n*)**, наприклад $\pi.evalf(5)$ або ***sqrt*(2).*evalf* (7)**.

А останнім нашим зауваженням буде констатація факту, що пояснювальних прикладів до вищевикладеного ми наводити не будемо, оскільки вони не нестимуть принципових відмінностей у використанні порівняно з тими, що нами уже вивчені в попередніх розділах і програмах, тож одразу ж перейдемо до створення програм реалізації прямого і оберненого операторів Лапласа.

8.4 Задачі щодо застосування операторів Лапласа в програмах мовою Python

Програма мовою Python для розв'язання задач, пов'язаних з використанням прямого оператора Лапласа для трансформації функцій *f(t)* дійсної змінної *t* на комплексну площину у вигляді функцій *F(p)* комплексної змінної *p*

(Програма 25)

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import *                           # Виклик усіх фукцій sympy
In [3]: t = symbols ('t')                             # Оголошення символьними
In [4]: p = symbols ('p')                             ….змінних t, p
In [5]: f = Function ('f')(t)                         # Оголошення символьними
In [6]: F = Function ('F')(p)                         ….функцій f(t), F(p)
In [7]: f = t                                         # Внесення значення функції f(t)
In [8]: f1 = f*exp(-p*t)                              # Формування ядра оператора
                                                      ….L{f(t)}
In [9]: F1 = integrate (f1,(t,0,oo))                  # Обчислення інтеграла Лапласа
In [10]: F1                                           # Виведення на екран значення
Out[10]:                                              ….інтеграла Лапласа у вигляді
Piecewise((p**(-2), Abs(arg(p)) < pi/2),              ….складеної функції з
 (Integral(t*exp(-p*t), (t, 0, oo)), True))           ….кортежним аргументом
In [11]: K = ((p**(-2), Abs(arg(p)) < pi/2),          # Виокремлення аргументу
(Integral(t*exp(-p*t), (t, 0, oo)), True))            ….функції Piecewise( )
In [12]: K[0]                                         # Виведення з кортежу K
Out [12]: (p**(-2), Abs(arg(p)) < pi/2)               ….першого елемента K[0]
In [13]: K[0][0]                                      # Виведення з кортежу K[0]
Out [13]: p**(-2)                                     ….першого елемента K[0][0]
In [14]: F = K[0][0]                                  # Обчислення оператора F
In [15]: print(F)                                     ….Лапласа та друк
Out[15]: p**(-2)                                      ….функції F(p)
In [16]: f2 = exp (- 2*t)                             # Внесення функції f2(t)
IIn [17]: f3 = f2*exp(-p*t)                           # Формування ядра інтеграла
                                                      ….L{f2(t)}
In [18]: F3 = integrate (f3,(t,0,oo))                 # Обчислення інтеграла Лапласа
In [19]: F3                                           # Виведення на екран значення
Out[19]:                                              ….інтеграла Лапласа у вигляді
Piecewise((1/(2*(p/2 + 1)), Abs(arg(p)) <= pi/2),     ….складеної функції з
```

```
(Integral(exp(-2*t)*exp(-p*t), (t, 0, oo)), True))
In [20]: K1 = ((1/(2*(p/2 + 1)), Abs(arg(p)) <= pi/2),
(Integral(exp(-2*t)*exp(-p*t), (t, 0, oo)), True))
In [21]: K1[0]
Out [21]: (1/(2*(p/2 + 1)), Abs(arg(p)) <= pi/2)
In [22]: K1[0][0]
Out [22]: 1/(p+2)
In [23]: F2 = K1[0][0]
In [24]: print(F2)
Out[24]: 1/(p+2)
In [25]: f4 = f*f2*exp(-p*t)

In [26]: F5 = integrate (f4,(t,0,oo))
In [27]: F5
Out [27]:
Piecewise((1/(4*(p/2 + 1)**2), Abs(arg(p)) <= pi/2),
(Integral(t*exp(-2*t)*exp(-p*t), (t, 0, oo)), True))
In [28]: K2 = ((1/(4*(p/2 + 1)**2), Abs(arg(p)) <= pi/2),
(Integral(t*exp(-2*t)*exp(-p*t), (t, 0, oo)), True))
In [29]: K2[0]
Out [29]: (1/(4*(p/2 + 1)**2), Abs(arg(p)) <= pi/2)
In [30]: K2[0][0]
Out [30]: 1/(p+2)**2
In [31]: F4 = K2[0][0]
In [32]: print(F4)
Out[32]: 1/(p+2)**2
In [33]: f5 = f*f2*sin (3*t)

In [34]: f6 = f5*exp(-p*t)
In [35]: F6 = integrate (f6,(t,0,oo))
In [36]: F6
Out[36]:
Piecewise((6/((1 + 9/(p + 2)**2)**2*(p + 2)**3),
2*Abs(arg(p + 2)) < pi), (Integral(t*exp(-2*t)*
exp(-p*t)*sin(3*t), (t, 0, oo)), True))
In [37]: K3 = ((6/((1 + 9/(p + 2)**2)**2*(p + 2)**3),
2*Abs(arg(p + 2)) < pi), (Integral(t*exp(-2*t)*
exp(-p*t)*sin(3*t), (t, 0, oo)), True))
In [38]: K3[0]
Out [38]: ((6/((1 + 9/(p + 2)**2)**2*(p + 2)**3),
2*Abs(arg(p + 2)) < pi))
In [39]: K3[0][0]
Out [39]:
6/((1 + 9/(p + 2)**2)**2*(p + 2)**3)
In [40]: F7 = K3[0][0]
In [41]: print(F7)
```

….кортежним аргументом
Виокремлення аргументу
….функції Piecewise( )
Виведення з кортежу ***K1***
….першого елемента ***K1[0]***
Виведення з кортежу ***K1[0]***
….першого елемента ***K1[0][0]***
Обчислення оператора ***F2***
….Лапласа та друк
….функції $F_2(p)$
Формування ядра інтеграла
….$L\{f(t)f_2(t)\}$
Обчислення інтеграла Лапласа
Виведення на екран значення
….інтеграла Лапласа у вигляді
….складеної функції з
….кортежним аргументом
Виокремлення аргументу
….функції Piecewise( )
Виведення з кортежу ***K2***
….першого елемента ***K2[0]***
Виведення з ***K2[0]***
….першого елемента
Обчислення оператора
….***F4*** Лапласа та друк
….функції $F_4(p)$
Формування добутку
….трьох функцій та ядра
….інтеграла Лапласа
Обчислення інтеграла
….Лапласа та на екран
….виведення
….у вигляді складеної
….функції з аргументом
….у вигляді кортежу ***K3***
Виокремлення аргументу
….Piecewise( ) у вигляді
….кортежу ***K3***
Виведення з кортежу ***K3***
….першого елемента ***K3[0]***

Виведення з ***K3[0]***
….першого елемента
….кортежу ***K3[0][0]***
Обчислення оператора
….$L\{f(t)f_2(t)\sin(3t)\}$

```
Out[41]:                                          ….та друкування його
6/((1 + 9/(p + 2)**2)**2*(p + 2)**3)              ….як функції F5(p)
```

Кінець програми 25

Програма мовою Python для розв'язання задач, пов'язаних з використанням оберненого оператора Лапласа у формі формул розкладання для трансформації функцій *F(p)* комплексної змінної *p на вісь* дійсної змінної *t* у вигляді функцій *f(t)*

(Програма 26)

```
In [1]: import sympy                              # Виклик ППП sympy
In [2]: from sympy import *                       # Виклик усіх фукцій sympy
In [3]: t = symbols ('t')                         # Символізація
In [4]: p = symbols ('p')                         ….змінних t, p
In [5]: x = Function ('x')(t)                     # Символізація
In [6]: y = Function ('y')(t)                     ….функцій x(t), y(t)
In [7]: C = Function ('C')(p)                     # Символізація
In [8]: D = Function ('D')(p)                     ….функцій C(p), D(p),
In [9]: W = Function ('W')(p)                     ….W(p)
In [10]: X = Function ('X')(p)                    # Символізація
In [11]: Y = Function ('Y')(p)                    ….функцій X(p), Y(p),
In [12]: D1 = Function ('D1')(p)                  ….D1(p)
In [13]: Y1 = Function ('Y1')(p,t)                # Символізація
In [14]: Y2 = Function ('Y2')(p,t)                ….функцій Y1(p,t), Y2(p,t)
In [15]: C = 2*p+4                                # Внесення чисельника і
In [16]: D = p**2+7*p+12                          ….знаменника для W(p)
In [17]: W = C/D                                  # Формування W(p)
In [18]: W                                        # Виклик на екран
Out[18]:                                          ….функції W(p)
(2*p + 4)/(p**2 + 7*p + 12)
In [19]: x = t                                    # Внесення функції x(t)
In [20]: x1 = x*exp(-p*t)                         ….та формування ядра
                                                  ….інтеграла Лапласа
In [21]: X1 = Function ('X1')(p)                  # Символізація функції
In [22]: X1 = integrate (x1,(t,0,oo));X1          ….X1(p) та її визначення
Out[22]:                                          ….інтегруванням функції x1
Piecewise((p**(-2), Abs(arg(p)) < pi/2),          ….у вигляді складеної
(Integral(t*exp(-p*t), (t, 0, oo)), True))        …. функції з аргументом K1
In [23]: K1 = ((p**(-2), Abs(arg(p)) < pi/2),     # Виокремлення аргументу
 (Integral(t*exp(-p*t), (t, 0, oo)), True))       ….K1 функції Piecewise( )
In [24]: K1[0]                                    # Виведення з кортежу K1
Out[24]: (p**(-2), Abs(arg(p)) < pi/2)            ….першого елемента K1[0]
In [25]: X = K1[0][0]                             # Обчислення оператора
In [26]: X                                        ….Лапласа L{x(t)} та
Out[26]:                                          ….виведення на друк у
p**(-2)                                           ….вигляді функції X(p)
In [27]: Y = W*X                                  # Визначення зображення
In [28]: Y                                        ….Y(p) за Лапласом
```

```
Out[28]:
(2*p + 4)/(p**2*(p**2 + 7*p + 12))
In [29]: C1 = Function (‘C1’)(p)
In [29]: expr = Y
In [30]: C1, D1 = fraction (expr)
In [31]: print(C1)
2*p + 4
In [32]: print(D1)
p**2*(p**2 + 7*p + 12)
In [33]: D2 = Function ('D2')(p)
In [34]: D2 = D1.diff(p)
In [35]: D2
Out[35]:
p**2*(2*p + 7) + 2*p*(p**2 + 7*p + 12)
In [36]: solveset (Eq(D1,0), p)
Out[36]:
FiniteSet(-4, -3, 0)
In [37]: roots (Eq(D1,0), p)
Out[37]: {-3: 1, -4: 1, 0: 2}
In [38]: d0 ={ }
In [39]: d0["a"]=-3
In [40]: d0["b"]=-4
In [41]: d0["c"]=0
In [42]: d0
Out[42]: {'a': -3, 'b': -4, 'c': 0}
In [43]: p1,p2,p3 = symbols('p1 p2 p3')
In [44]: p1 = d0['a']
In [45]: p1
Out[45]: -3
In [46]: p2 = d0['b']
In [47]: p2
Out[47]: -4
In [48]: p3 = d0['c']
In [49]: p3
Out[49]: 0
In [50]: Y1 = C1*exp(p*t)/D2
In [51]: Y1
Out[51]:
(2*p + 4)*exp(p*t)/(p**2*(2*p + 7) + 2*p*(p**2
+ 7*p + 12))
In [52]: Y2 = diff(C1*(p-p3)**2*exp(p*t)/D1,p)
In [53]: Y2
Out[53]:
t*(2*p + 4)*exp(p*t)/(p**2 + 7*p + 12) + (-2*p
- 7)*(2*p + 4)*exp(p*t)/(p**2 + 7*p + 12)**2
+ 2*exp(p*t)/(p**2 + 7*p + 12)
```

….функції ***y(t)*** та його
…. виведення на екран
Символізація функції ***C1()***
Створення виразу ***expr,***
….придатного для
….визначення чисельника
….та знаменника
….зображення ***Y(p),***
….їх виділення та друк
Символізація функції ***D2()***
Визначення похідної
….від знаменника ***D1(p)***
….зображення ***Y(p)*** та її
….виведення на екран
Визначення полюсів
….***Y(p)*** розв’язанням
….рівняння ***D1(p) = 0***
Інший спосіб визначення
….полюсів ***Y(p)***
Створення пустого
….словника ***d0*** і внесення
….в ***d0*** полюсів ***Y(p)***
….з ключами “a”, ”b”, ”c”
….та виведення на екран
….заповненого словника
Символізація ***p1, p2, p3***
Формування полюса ***p1***
….зображення ***Y(p)*** та його
….виведення на екран
Формування полюса ***p2***
….зображення ***Y(p)*** та його
….виведення на екран
Формування полюса ***p3***
….зображення ***Y(p)*** та його
….виведення на екран
Формування складової
….функції ***y(t)***,
….зумовленої простим
….полюсом ***Y(p)***, та її
….виведення на екран
Формування складової
….функції ***y(t)***,
….зумовленої кратним
….полюсом ***Y(p)***, та її
….виведення на екран

```
In [54]: y = Y1.subs(p,p1)+Y1.subs(p,p2)+Y2.subs(p,p3)   # Визначення y(t)
In [55]: y                                               ….застосуванням
Out[55]:                                                 ….для L^{-1}{Y(p)}
t/3 - 1/36 - 2*exp(-3*t)/9 + exp(-4*t)/4                 ….теореми розкладання
In [56]: p11 = plot(x,(t,0,2),show=False, line_color = 'c')   # Побудова кривої x(t) = t
In [57]: p22 = plot(y,(t,0,2),show=False, line_color = 'r')   # Побудова кривої y(t)
In [58]: p11.extend(p22)                                 # Поєднання на одному
In [59]: p11.show( )                                     ….рисунку обох кривих і
                                                         ….їх екранізація
```

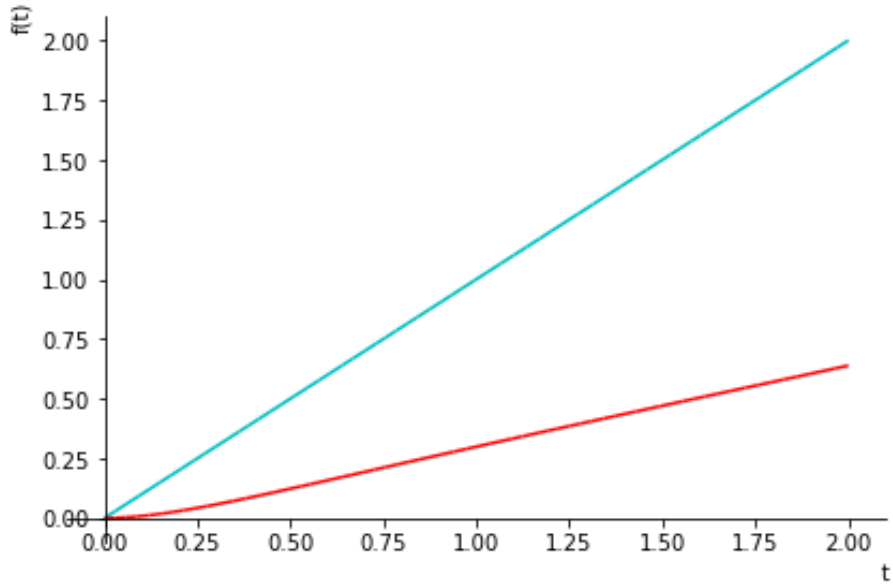


Рисунок 8.1 – Графіки вхідного сигналу ***x(t) = t***, що надходить в динамічну систему з заданою передаточною функцією ***W(p)***, та її вихідного сигналу ***y(t)*** на відрізку часу $t \in [0,2]$

Кінець програми 26

8.5 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з застосуванням операторів Фур'є

Аналізуючи вирази (8.43)–(8.58), якими визначаються різні варіанти і форми застосування прямого та оберненого операторів Фур'є, бачимо, що для їх програмної реалізації треба спочатку згадати і про те, про що ми уже розповіли в підрозділі 2.2 частини 1 цього нашого навчального посібника [1], супроводивши цю розповідь пояснювальним прикладом № 18, присвяченим операціям з комплексними числами в ППП sympy; і про те, що ми використали в підрозділі 2.3 при створенні програми 8, присвяченої визначенню норми, метрики та скалярного добутку в гільбертових просторах, елементами яких є функції комплексної змінної, а потім ще й додати ті особливості оперування з комплексними числами і функціями комплексної змінної в програмах мовою Python, про які мова раніше ще не йшла.

Отже, спочатку нагадаємо про те, що ***Python*** може працювати з комплексними числами ***u, v*** одразу ж після завантаження будь-якої його консолі, наприклад ***Spyder***, підтримуючи операції їх додавання **(*u*+*v*)**, множення **(*u**v*)**, ділення **(*u*/*v*)** та піднесення до степеня ***n*(*u***n*)**, а також визначення дійсної **(*u.real*)** та уявної **(*u.imag*)** частин комплексного числа**.** Але потрібно пам'ятати, що у цьому випадку уявна одиниця записується символом ***j***, який записується в уявній частині комплексного числа біля числа, яким вона визначається чисельно, справа без знака множення між ними, наприклад ***5j*** або ***0.25j***. Але, якщо ми побажаємо визначити число спряжене **(*u.conjugate*)** *з* заданим комплексним числом ***u***, то нам цього без виклику якогось пакета прикладних програм, сумісного з ***Python***, який здійснює операції з комплексними числами, зробити не вдасться. Приклади викладеного вище наведені в пояснювальному прикладі № 10.

Пояснювальний приклад № 10

```
In [1]: u=5+2j                          # Задаємо комплексне число u
In [2]: v=3-5j                          # Задаємо комплексне число v
In [3]: u*v                             # Перемножаємо комплексні
Out[3]: (25-19j)                        ….числа u та v
In [4]: u/v                             # Ділимо комплексне число u
Out[4]: (0.14705882352941177+           ….на комплексне число v
0.911764705882353j)
In [5]: z = 2*u+4*v                     # Визначаємо комплексне
In [6]: z                               ….число z як зважену суму
Out[6]: (22-16j)                        ….комплексних чисел u та v
In [7]: u+v                             # Визначаємо суму
Out[7]: (8-3j)                          ….комплексних чисел u та v
In [8]: u-v                             # Визначаємо різницю
Out[8]: (2+7j)                          ….комплексних чисел u та v
In [9]: (u+v)*(u-v)                     # Перемножаємо суму
Out[9]: (37+50j)                        ….чисел u та v на їх різницю
In [10]: (u+v)/(u-v)                    # Ділимо суму
Out[10]: (-0.09433962264150945-         ….чисел u та v на їх різницю
1.1698113207547172j)
In [11]: u**2                           # Підносимо комплексне
Out[11]: (21+20j)                       ….число u до квадрата
In [12]: v**3                           # Підносимо комплексне
Out[12]: (-198-10j)                     ….число v до куба
In [13]: u.real                         # Визначаємо дійсну частину
Out[13]: 5.0                            ….комплексного числа u
In [14}:u.imag                          # Визначаємо уявну частину
Out[14]: 2.0                            ….комплексного числа u
In [15]: v.real                         # Визначаємо дійсну частину
Out[15]: 3.0                            ….комплексного числа v
In [16]: v.imag                         # Визначаємо уявну частину
Out[16]: -5.0                           ….комплексного числа v
In [17]: u.conjugate                    # Переконуємось, що число
Out[17]: <function complex.conjugate>   ….спряжене з числом u
                                        ….без застосування якогось
```

….пакета програм, в якому є
….функції для операцій з
….комплексними числами,
….визначене бути не може

Кінець пояснювального прикладу № 10.

А тому більше можливостей для здійснення операцій з комплексними числами та функціями комплексної змінної надає нам ППП ***sympy***, який для можливості використання потрібно попередньо викликати, як це продемонстровано в пояснювальному прикладі № 11, в якому продемонстровано як працювати з комплексними числами. Але потрібно пам'ятати, що при здійсненні операцій з комплексними числами у ППП ***sympy*** уявна одиниця записується уже не символом ***j***, а символом ***I*** (великою латинською літерою), *який* записується в уявній частині комплексного числа біля числа, яким вона визначається чисельно, справа з використанням тепер уже символа множення між ними, наприклад ***5*I*** або ***0.25*I***. Ну і комплексні змінні, щоб з ними працювати в ППП ***sympy***, потрібно оголошувати символьними з використанням тих же функцій, які ми використовували при оголошенні символьними дійсних змінних, після чого і на комплексні змінні розповсюджуватиметься дія усіх тих функцій, які ми вже раніше використовували, працюючи в ППП ***sympy*** з символьними змінними дійсного характеру.

Пояснювальний приклад № 11

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import*                            # Виклик із sympy усіх функцій
In [3]: a,b,c,d = symbols ('a b c d')                 # Символізація a, b, c, d
In [4]: var('u,v')                                    # Оголошення символьними
Out[4]: (u, v)                                        ….змінних u, v
In [5]: z = symbols ('z:10');z                        # Оголошення символьною
Out[5]: (z0, z1, z2, z3, z4, z5, z6, z7, z8, z9)      ….змінної z з індексами
In [6]: p = symbols ('p')                             # Символізація змінної p
In [7]: p = u + v*I                                   # Формування комплексу p
In [8]: z0 = a*p**2+b*p;z0                            # Формування комплексу z0
Out[8]:                                               ….як функції від p та розкриття
a*(u + I*v)**2 + b*(u + I*v)                          ….його через складові u, v
In [9]: z0 = expand (z0);z0                           # Розкриття дужок у виразі для
Out[9]:                                               ….z0 та його спрощення
a*u**2 + 2*I*a*u*v - a*v**2 + b*u + I*b*v
In [10]: z0 = collect (z0,I);z0                       # Формування у комплексі z0
Out[10]:                                              ….дійсної та уявної складових
a*u**2 - a*v**2 + b*u + I*(2*a*u*v + b*v)
In [11]: z1 = z0.subs([(a,2),(b,4)]);z1               # Трансформація комплексу z0
Out[11]:                                              ….у комплекс z1 підстановкою
2*u**2 + 4*u - 2*v**2 + I*(4*u*v + 4*v)               ….значень параметрів a,b
In [12]: z2 = z1.subs([(u,3),(v,5)]);z2               # Трансформація комплексу z1
Out[12]:                                              ….у комплекс z2 підстановкою
-20 + 80*I                                            ….значень змінних u, v
In [13]: z3 = conjugate (z2);z3                       # Визначення комплексного
Out[13]:                                              ….числа z3, спряженого з
-20 - 80*I                                            ….комплексним числом z2
```

```
In [14]: z4 = c*p**3+d*p;z4                    # Формування комплексу z4
Out[14]:                                       ….як функції від p та розкриття
c*(u + I*v)**3 + d*(u + I*v)                   ….його через складові u, v
In [15]: z4 = expand (z4);z4                   # Розкриття дужок у виразі для
Out[15]:                                       ….z4 та його спрощення
c*u**3 + 3*I*c*u**2*v - 3*c*u*v**2 - \
I*c*v**3 + d*u + I*d*v
In [16]: z4 = collect (z4,I);z4                # Формування у комплексі z4
Out[16]:                                       ….дійсної та уявної складових
c*u**3 - 3*c*u*v**2 + d*u +\
 I*(3*c*u**2*v - c*v**3 + d*v)
In [17]: z5 = z4.subs([(c,3),(d,2)]);z5        # Трансформація комплексу z4
Out[17]:                                       ….у комплекс z5 підстановкою
3*u**3 - 9*u*v**2 + 2*u +\                     ….значень параметрів c,d
 I*(9*u**2*v - 3*v**3 + 2*v)
In [18]: z6 = z5.subs([(u,3),(v,5)]);z6        # Трансформація комплексу z5
Out[18]:                                       ….у комплекс z6 підстановкою
-588 + 40*I                                    ….значень змінних u, v
In [19]: z7 = conjugate (z6);z7                # Визначення комплексного
Out[19]:                                       ….числа z7, спряженого з
-588 - 40*I                                    ….комплексним числом z6
In [20]: z2*z3                                 # Перемноження комплексно-
Out[20]:                                       ….спряжених чисел z2 та z3
(-20 - 80*I)*(-20 + 80*I)
In [21]: expand (z2*z3)                        # Розкриття дужок та
Out[21]:                                       ….спрощення у добутку
6800                                           ….спряжених чисел z2 та z3
In [22]: z6*z7                                 # Перемноження комплексно-
Out[22]:                                       ….спряжених чисел z6 та z7
(-588 - 40*I)*(-588 + 40*I)
In [23]: expand (z6*z7)                        # Розкриття дужок та
Out[23]:                                       ….спрощення у добутку
347344                                         ….спряжених чисел z6 та z7
In [24]: z8 = z2+z6;z8                         # Формування комплексу z8
Out[24]:                                       ….як суми комплексів z2 та z6
-608 + 120*I
In [25]: z9 = z2-z6;z9                         # Формування комплексу z9
Out[25]:                                       ….як різниці комплексів z2 та z6
568 + 40*I
In [26]: z = symbols ('z10:17');z              # Оголошення символьною
Out[26]:                                       ….змінної z з індексами,
(z10, z11, z12, z13, z14, z15, z16)            ….починаючи з z10
In [27]: mod = symbols ('mod:2');mod           # Оголошення символьною
Out[27]: (mod0, mod1)                          ….змінної mod з індексами
In [28]: faz = symbols ('faz:2');faz           # Оголошення символьною
Out[28]: (faz0, faz1)                          ….змінної faz з індексами
```

```
In [29]: z10 = z8*z9;z10                                  # Формування комплексу z10
Out[29]:                                                  ….множенням комплексу z8
(-608 + 120*I)*(568 + 40*I)                               ….на комплекс z9
In [30]: z10 = expand (z10);z10                           # Розкриття дужок у
Out[30]:                                                  ….комплексі z10 та його
-350144 + 43840*I                                         ….спрощення
In [31]: z11 = z8/z9;z11                                  # Формування комплексу z11
Out[31]:                                                  ….діленням комплексу z8
(-608 + 120*I)*(568 - 40*I)/324224                        ….на комплекс z9
In [32]: z11 = expand(z11);z11                            # Розкриття дужок у
Out[32]:                                                  ….комплексі z11 та його
-313/298 + 85*I/298                                       ….спрощення
In [33]: z12 = z11.n(3);z12                               # Формування комплексу z12
Out[33]:                                                  ….шляхом утримання в z11
-1.05 + 0.285*I                                           ….лише трьох цифр у кожній
                                                          ….зі складових
In [34]: mod0 = Abs(z12);mod0                             # Визначення модуля
Out[34]:                                                  ….комплексного числа z12
1.09
In [35]: faz0 =arg(z12);faz0                              # Визначення аргументу
Out[35]:                                                  ….комплексного числа z12
-0.265 + pi                                               ….та його наближене
In [36]: faz0 = faz0.n(3);faz0                            ….формування шляхом
Out[36]:                                                  ….утримання лише
2.88                                                      ….трьох цифр
In [37]: z13 = mod0*exp(faz0*I);z13                       # Запис комплексного числа
Out[37]:                                                  ….z12 у показниковій формі
1.09*exp(2.88*I)                                          ….з позначенням його як z13
In [38]: z13 = z13.expand(complex=True);z13               # Трансформація показникової
Out[38]:                                                  ….форми комплексного числа
-1.05 + 0.285*I                                           ….z13 в алгебраїчну
In [39]: mod1 = Abs (z9);mod1                             # Визначення модуля
Out[39]:                                                  ….комплексного числа z9
8*sqrt(5066)                                              ….та його наближене
In [40]: mod1 = mod1.n(5);mod1                            ….формування шляхом
Out[40]:                                                  ….утримання лише
569.41                                                    ….п’яти цифр
In [41]: faz1 = arg(z9);faz1                              # Визначення аргументу
Out[41]:                                                  ….комплексного числа z9
atan(5/71)                                                ….та його наближене
In [42]: faz1 = faz1.n(5);faz1                            ….формування шляхом
Out[42]:                                                  ….наближеного
0.070306                                                  ….обчислення atan(5/71)
In [43]: z14 = mod1*exp(faz1*I);z14                       # Запис комплексного числа
Out[43]:                                                  ….z9 у показниковій формі
569.41*exp(0.070306*I)                                    ….з позначенням його як z14
```

```
In [44]: z14 = z14.expand(complex =                # Трансформація показникової
 True);z14                                         ….форми комплексного числа
Out[44]:                                           ….z14 в алгебраїчну
568.0 + 40.0*I
In [45]: re(z14)                                   # Визначення дійсної частини
Out[45]:                                           ….комплексного числа z14
568.00
In [46]: im(z14)                                   # Визначення уявної частини
Out[46]:                                           ….комплексного числа z14
40.000
```

Кінець пояснювального прикладу № 11

Але варто згадати і про те, що в ***ППП numpy***, в якому реалізується зокрема обчислення скалярного добутку векторів функцією ***matmul( )***, ця функція може обчислювати скалярний добуток і векторів, проєкції яких задані комплексними числами – приклади цього наведені у пояснювальному прикладі № 12. Вважаємо за доцільне нагадати також, що скалярний добуток двох векторів дорівнює сумі добутків однойменних проєкцій цих векторів, та звертаємо увагу, по перше, на те, що при викликах масивів в ***ППП numpy*** проєкції кожного вектора записуються рядками цих масивів, а тому добутки однойменних проєкцій обчислюються по вертикалі масиву, а по-друге, звертаємо увагу ще й на те, що в разі, якщо уявна частина комплексного числа дорівнює одиниці, то цю одиницю треба обов'язково вписувати перед символом ***j***, інакше програма відмовиться її розпізнавати, як це показано в кінці пояснювального прикладу № 12.

Пояснювальний приклад № 12

```
In [1]: import numpy as np                         # Виклик ППП numpy як np
In [2]: A2 = np.array([[1,2],[3,4]]);A2            # Формування масиву
Out[2]:                                            ….проєкцій двох векторів
array([[1, 2],                                     ….по дві у кожного, яким
       [3, 4]])                                    ….відповідають рядки
In [3]: np.matmul([1,2],[3,4])                     # Обчислення скалярного
Out[3]:                                            ….добутку двох векторів
11
In [4]: A2c = np.array([[1+2j,2-3j],[3+4j,         # Формування масиву
4-5j]]);A2c                                        ….проєкцій двох векторів
Out[4]:                                            ….з комплексними числами
array([[1.+2.j, 2.-3.j],                           ….в рядках
       [3.+4.j, 4.-5.j]])
In [5]: np.matmul([1+2j,2-3j],[3+4j,4-5j])         # Обчислення скалярного
Out[5]:                                            ….добутку двох векторів
(-12-12j)                                          ….з комплексними проєкціями
In [6]: A3=np.array([[1,2,5],[3,4,6]]);A3          # Формування масиву
Out[6]:                                            ….проєкцій двох векторів
array([[1, 2, 5],                                  ….по три у кожного, яким
       [3, 4, 6]])                                 ….відповідають рядки
In [7]: np.matmul([1,2,5],[3,4,6])                 # Обчислення скалярного
Out[7]:                                            ….добутку двох векторів
41
```

```
In [8]: A3c=np.array([[1+2j,2+3j,5-4j],[3+4j,                # Формування масиву
4+5j,6-5j]]);A3c                                            ….проєкцій двох векторів
Out[8]:                                                     ….по три у кожного у вигляді
array([[1.+2.j, 2.+3.j, 5.-4.j],                            ….комплексних чисел, яким
       [3.+4.j, 4.+5.j, 6.-5.j]])                           ….відповідають рядки
In [9]: np.matmul([1+2j,2+3j,5-4j],[3+4j,                   # Обчислення скалярного
4+5j,6-5j])                                                 ….добутку двох векторів
Out[9]:
(-2-17j
In [10]: A1c = np.array([[1+j,2-3j],[3+4j,                  # Формування масиву
4-5j]]);A1c                                                 ….проєкцій двох векторів
Traceback (most recent call last):                          ….з комплексними числами,
                                                            ….одне з яких записане у
  File "<ipython-input-12-3073e3950d2d>",                   ….формі, що не
 line 1, in <module>                                        ….сприймається в ППП sympy
   A1c = np.array([[1+j,2-3j],[3+4j,4-5j]]);A1c

NameError: name 'j' is not defined

In [11]: A11c = np.array([[1+1j,2-3j],[3+4j,                # Формування масиву
4-5j]]);A11c                                                ….проєкцій двох векторів
Out[11]:                                                    ….з комплексними числами,
array([[1.+1.j, 2.-3.j],                                    ….записаними у формі, що
       [3.+4.j, 4.-5.j]])                                   …. сприймається в ППП sympy
In [12]: np.matmul([1+1j,2-3j],[3+4j,4-5j])                 # Обчислення скалярного
Out[12]:                                                    ….добутку двох векторів
(-8-15j)                                                    ….з комплексними проєкціями,
                                                            ….записаними правильно
```

Кінець пояснювального прикладу № 12

8.6 Задачі щодо застосування операторів Фур'є в програмах мовою Python

Програма мовою Python для розв'язання задач, пов'язаних з використанням оператора Фур'є для визначення частотних характеристик об'єкта дослідження з передаточною функцією $W(p)=\frac{4p^2+p+5}{p^3+2p^2+5p+1}$, в якій комплексна змінна p задається виразом $p=u+jv$, де $j=\sqrt{-1}$ – уявна одиниця

(Програма 27)

```
In [1]: import sympy                                        # Виклик ППП sympy
In [2]: from sympy import*                                  # Виклик із sympy усіх функцій
In [3]: p = symbols ('p')                                   # Оголошення символьною p
In [4]: a = symbols ('a:3');a                               # Оголошення символьним
Out[4]: (a0, a1, a2)                                        ….параметра a з індексами
In [5]: b = symbols ('b:4');b                               # Оголошення символьним
Out[5]: (b0, b1, b2, b3)                                    ….параметра b з індексами
In [6]: f1 = Function ('f1')(p)                             # Оголошення символьними
```

```
In [7]: f2 = Function ('f2')(p)                    ….функцій f1(p), f2(p) та
In [8]: W = Function ('W')(p)                      ….функції W(p)
In [9]: f1 = sum([a[i]*p**i for i in               # Визначення функції f1(p)
 range(3)]);f1
Out[9]:
a0 + a1*p + a2*p**2
In [10]: f2 = sum([b[i]*p**i for i in              # Визначення функції f2(p)
range(4)]);f2
Out[10]:
b0 + b1*p + b2*p**2 + b3*p**3
In [11]: W = f1/f2;W                               # Визначення функції W(p)
Out[11]:
(a0 + a1*p + a2*p**2)/(b0 + b1*p +
b2*p**2 + b3*p**3)
In [12]: f11 = Function ('f11')(p)                 # Оголошення символьними
In [13]: f22 = Function ('f22')(p)                 ….функцій f11(p), f22(p)
In [14]: f11 = f1.subs([(a[0],5),(a[1],1),         # Визначення функції f11(p)
(a[2],4)]);f11
Out[14]:
4*p**2 + p + 5
In [15]: f22 = f2.subs([(b[0],1),(b[1],5),(b[2],2),# Визначення функції f22(p)
(b[3],1)]);f22
Out[15]:
p**3 + 2*p**2 + 5*p + 1
In [16]: W = f11/f22;W                             # Визначення функції W(p)
Out[16]:                                           ….як передаточної функції
(4*p**2 + p + 5)/(p**3 + 2*p**2 + 5*p + 1)         ….об’єкта дослідження
In [17]: v = symbols (‘v’,real=True)               # Оголошення символьною v
                                                   ….як дійсної змінної
In [18]: f111 = Function ('f111')(v)               # Оголошення символьними
In [19]: f222 = Function ('f222')(v)               ….функцій f111(v), f222(v)
In [20]: expr1=f11                                 # Надання функції f11(p)
                                                   ….статусу функції expr
In [21]: f111 = expr1.subs (p,v*I);f111            # Визначення функції f111(v)
Out[21]:
-4*v**2 + I*v + 5
In [22]: expr2=f22                                 # Надання функції f22(p)
                                                   ….статусу функції expr
In [23]: f222 = expr2.subs(p,v*I);f222             # Визначення функції f222(v)
Out[23]:
-I*v**3 - 2*v**2 + 5*I*v + 1
In [24]: W1 = Function ('W1')(v)                   # Символізація функції W1(v)
In [25]: W1 =f111/f222;W1                          # Визначення функції W1(v)
Out[25]:                                           ….як амплітудно-фазової
(-4*v**2 + I*v + 5)/(-I*v**3 - 2*v**2 +            ….частотної характеристики
 5*I*v + 1)                                        ….об’єкта дослідження
```

```
In [26]: f1111 = Function ('f1111')(v)
In [27]: f1111 = collect(f111,I);f1111
Out[27]:
-4*v**2 + I*v + 5
In [28]: f2222 = Function ('f2222')(v)
In [29]: f2222 = collect(f222,I);f2222
Out[29]:
-2*v**2 + I*(-v**3 + 5*v) + 1
In [30]: h2222 = conjugate(f2222);h2222
Out[30]:
-2*v**2 - I*(-v**3 + 5*v) + 1
In [31]: fh = Function ('fh')(v)
In [32]: fh = f2222*h2222;fh
Out[32]:
(-2*v**2 - I*(-v**3 + 5*v) + 1)*(-2*v**2 +
 I*(-v**3 + 5*v) + 1)
In [33]: ffhh = Function ('ffhh')(v)
In [34]: ffhh = expand(fh);ffhh
Out[34]:
v**6 - 6*v**4 + 21*v**2 + 1

In [35]: fhf = Function ('fhf')(v)
In [36]: fhf = f1111*h2222;fhf
Out[36]:
(-4*v**2 + I*v + 5)*(-2*v**2 –
I*(-v**3 + 5*v) + 1)
In [37]: fhfh = Function ('fhfh')(v)
In [38]: fhfh = expand (fhf);fhfh
Out[38]:
-4*I*v**5 + 7*v**4 + 23*I*v**3 -\
 9*v**2 - 24*I*v + 5
In [39]: fhhf = Function ('fhhf')(v)
In [40]: fhhf = collect (fhfh,I);fhhf
Out[40]:
7*v**4 - 9*v**2 + I*(-4*v**5 +
23*v**3 - 24*v) + 5
In [41]: W1 = fhhf/ffhh;W1
Out[41]:
(7*v**4 - 9*v**2 + I*(-4*v**5 + 23*v**3 –
24*v) + 5)/(v**6 - 6*v**4 + 21*v**2 + 1)
In [42]: D = Function ('D')(v)
In [43]: D = re(W1);D
Out[43]:
(7*v**4 - 9*v**2 + 5)/(v**6 - 6*v**4 +
21*v**2 + 1)
In [44]: U = Function ('U')(v)
```

```
# Символізація функції f1111(v)
# Визначення функції f1111(v)
….як комплексної функції
….змінної v
# Символізація функції f2222(v)
# Визначення функції f2222(v)
….як комплексної функції
….змінної v
# Визначення функції h2222(v)
….як функції комплексно-
….спряженої з функцією f2222(v)
# Символізація функції fh(v)
# Визначення функції fh(v)
….як добутку комплексно-
….спряжених функцій
…. f2222(v) та h2222(v)
# Символізація функції ffhh(v)
# Спрощення функції fh(v)
….шляхом перемноження
….виразів у дужках та
….об’єднання подібних
# Символізація функції fhf(v)
# Перемноження чисельника
….f1111(v) АФЧХ W1(v) на
….комплексно спряжений
….її знаменник h2222(v)
# Символізація функції fhfh(v)
# Спрощення функції fhf(v)
….шляхом перемноження
….виразів у дужках та
….об’єднання подібних
# Символізація функції fhhf(v)
# Визначення функції fhhf(v)
….як комплексної функції
….змінної v

# Трансформація АФЧХ W1(v)
….у форму, що не містить
….уявної одиниці I
….в знаменнику
# Символізація функції D(v)
# Визначення функції D(v)
….як дійсної частотної
….характеристики (ДЧХ)
….об'єкта дослідження
# Символізація функції U(v)
```

```
In [45]: U = im(W1);U
Out[45]:
(-4*v**5 + 23*v**3 - 24*v)/(v**6 - 6*v**4 +
21*v**2 + 1)
In [46]: A = Function ('A')(v)
In [47]: A = Abs(W1);A
Out[47]:
sqrt(16*v**10 - 135*v**8 + 595*v**6 –
953*v**4 + 486*v**2 + 25)*Abs(1/(v**6 –
 6*v**4 + 21*v**2 + 1))
In [48]: Faz = Function ('Faz')(v)
In [49]: Faz = arg(W1);Faz
Out[49]:
arg((7*v**4 - 9*v**2 + I*v*(-4*v**4 +
23*v**2 - 24) + 5)/(v**6 - 6*v**4 +
21*v**2 + 1))
In [50]: Faz = atan(U/D);Faz
Out[50]:
atan((-4*v**5 + 23*v**3 - 24*v)/(7*v**4 –
 9*v**2 + 5))
In [51]: from sympy.plotting import plot

In [52]: plot(A,(v,0,10))
Out[52]: <sympy.plotting.plot.Plot at 0x
15bd2205310>
```

```
# Визначення функції U(v)
….як уявної частотної
….характеристики (УЧХ)
….об'єкта дослідження
# Символізація функції A(v)
# Визначення функції A(v)
….як амплітудної частотної
….характеристики (АЧХ)
….об'єкта дослідження

# Символізація функції Faz(v)
# Визначення функції Faz(v)
….як фазової частотної
….характеристики (ФЧХ)
….об'єкта дослідження

# Визначення функції Faz(v)
….іншим способом


# Виклик функції для
….побудови графіків
# Побудова графіка АЧХ
```

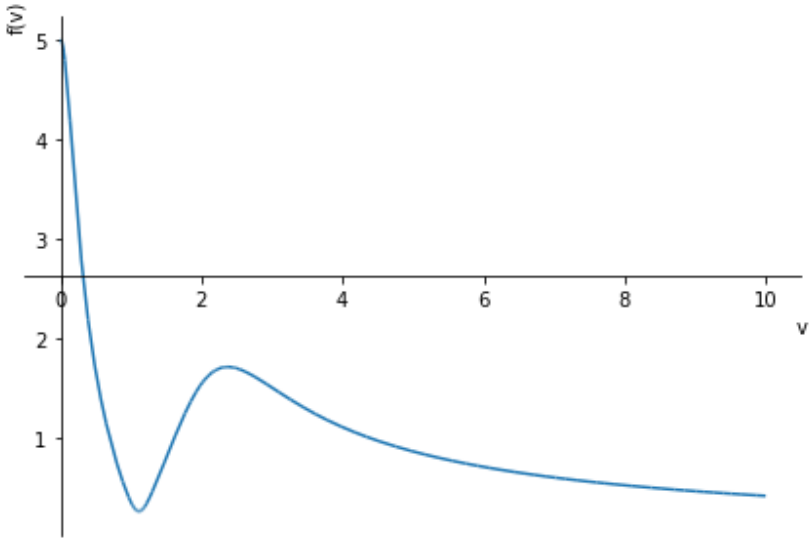


Рисунок 8.2 – Графік амплітудної частотної характеристики об’єкта з передаточною функцією, заданою в вихідних умовах програми

In [53]: plot(Faz,(v,0,10)) # Побудова графіка ФЧХ
Out[53]: <sympy.plotting.plot.Plot at 0x
15bd2205760>

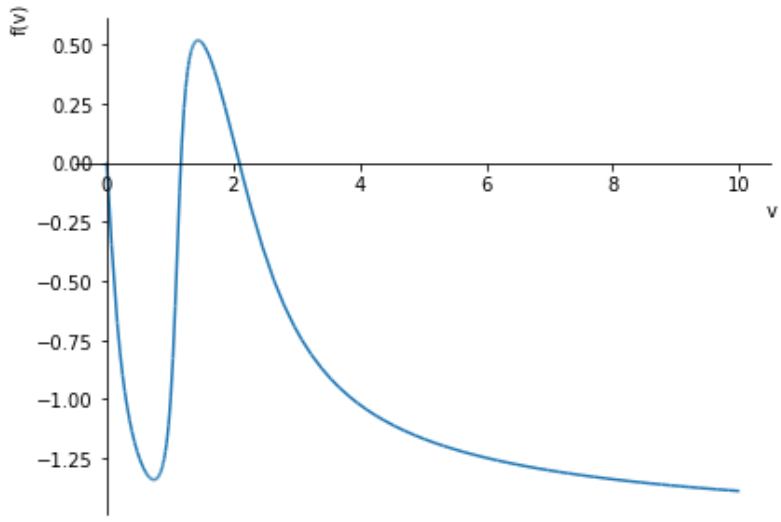


Рисунок 8.3 – Графік фазової частотної характеристики об’єкта
з передаточною функцією, заданою в вихідних умовах програми

Кінець програми 27

Розділ 9 СПЕЦІАЛЬНІ ОПЕРАТОРИ З КЛАСУ ДИСКРЕТНИХ (В ПРИКЛАДАХ І ПРОГРАМАХ)

9.1 Характеристики дискретних операторів, які перетворюють диференціальні рівняння в різницеві

У кінці восьмого підрозділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. Дайте означення решітчастої функції, наведіть приклад із побудовою графіка.
2. Які різницеві оператори є аналогом похідної для решітчастих функцій?
3. Чому недостатньо лише правого різницевого оператора першого порядку для характеристики швидкості зміни решітчастої функції? Який вихід?
4. Чим відрізняється рівняння в скінченних різницях від різницевого рівняння?
5. Як за диференціальним рівнянням побудувати різницеве? Наведіть приклад.
6. Що собою являє Z-оператор Деча для відображення решітчастих функцій і чому виникає в ньому потреба? Наведіть приклад Z-відображення решітчастої функції.
7. У чому полягає перевага використання лівих різницевих операторів під час застосування Z-оператора для аналізу дискретних систем?
8. Як отримати дискретну передаточну функцію системи, що в часовій області описується різницевими рівняннями?
9. Що собою являє обернений оператор Деча, і яку практичну реалізацію його ви знаєте?

Тож із відповідей на ці питання ми і розпочнемо викладення змісту першого підрозділу дев'ятого розділу другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих питань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2]. Отже, почнемо викладення матеріалу цього підрозділу з розлогої цитати, взятої з нашого базового навчального посібника [2], в якій змінена не лише нумерація формул, а й опущено багато проміжних викладок, що робить цей матеріал не стільки цитатою, скільки її вільним переказом, через що ми текст цієї цитати не беремо в лапки.

Прямий оператор Лапласа, який нами розглянуто у попередньому розділі, реалізовував закон відображення множини неперервних функцій дійсного аргументу, заданого на числовій осі, у множину неперервних функцій комплексного аргументу, заданого на комплексній площині, координати точок якої теж є дійсними числами. А обернений оператор Лапласа діяв навпаки, реалізуючи обернений процес перетворення вказаних множин одна в одну.

Прямий оператор Фур'є, який нами теж розглянуто у попередньому розділі, реалізовував закон відображення неперервних функцій дійсного аргументу, заданого на числовій осі, у множину неперервних функцій комплексного аргументу, заданого на комплексній площині, координати точок якої теж є дійсними числами, але, на відміну від тих чисел, які задаються прямим оператором Лапласа, ці дійсні числа конкретизуються значеннями неперервних функцій кругової частоти. А обернений оператор Фур'є, як і обернений оператор Лапласа, діяв навпаки, теж реалізуючи обернений процес перетворення вказаних множин одна в одну.

Але для аналізу процесів у дискретних системах потрібно використовувати апарат функцій дискретного аргументу, а тому математики задались метою сконструювати такий оператор, який би зміг здійснити відображення множини функцій дискретного аргументу в множину функцій неперервного аргументу, а також сконструювати обернений оператор для оберненого відображення цих множин. Цю задачу розв'язав математик Деч, тож

запропоновані ним прямий і обернений оператори заслуговують на те, щоб бути названими прямим і оберненим операторами Деча, незважаючи на те, що при створенні їх теорії він використав деякі уже відомі математичні конструкції, запропоновані іншими математиками.

Але, перш ніж викласти матеріал, присвячений перетворенням множин функцій прямим та оберненим операторами Деча, згадаємо оператори, за допомогою яких із неперервних функцій неперервного аргументу синтезують функції дискретного аргументу.

Тож нехай $f(t)$ є неперервною функцією неперервного аргументу. Виберемо інтервал дискретності T і визначимо значення функції $f(t)$ лише при значеннях аргументу t, кратних T. У результаті цих дій отримаємо $f[kT]$ - функцію дискретного аргументу kT ($k = 0, 1, 2, \ldots$), яку математики домовились називати решітчастою (або гратчастою). Записується оператор відображення неперервної функції $f(t)$ у решітчасту функцію $f[kT]$ так:

$$f[kT] = \int_0^\infty f(t)\delta(t - kT)dt, \quad \text{T} > 0, \text{ k} = 0,1,2,..., \tag{9.1}$$

де δ(t) − дельта-функція, властивості якої визначаються виразами:

$$\delta(t) = \begin{cases} \infty, & t = 0, \\ 0, & t \neq 0, \end{cases} \tag{9.2}$$

$$\int_{-\infty}^{\infty} \delta(t)dt = 1. \tag{9.2}$$

Але на практиці цей оператор реалізується у вигляді алгоритму, який виглядає так:

$$f[kT] = f(t)|_{t=kT}, \quad T > 0, \ \ k = 0,1,2, \ldots \tag{9.3}$$

Легко бачити, що після вибору числового значення інтервалу дискретності T решітчаста функція $f[kT]$ стає функцією лише аргументу k, а тому автори багатьох навчальних посібників і монографій записують її у вигляді не $f[kT]$, а $f[k]$. Роблячи так, потрібно пам'ятати, що, по-перше, коефіцієнти у функціях $f[k]$ і $f[kT]$ різняться, а по-друге, що для відтворення неперервної функції $f(t)$, яка є породною для решітчастої функції $f[k]$, потрібно порядковий номер k, який виступає як аргумент решітчастої функції $f[k]$, помножити на числове значення інтервалу дискретності T до виконання операції над ним. Для того, щоб не тримати цього у пам'яті, будемо зберігати період дискретності T в аргументі решітчастої функції $f[kT]$.

Як відомо з курсу математичного аналізу, швидкість зміни неперервної функції $f(t)$ у кожній точці графіка характеризується значенням її похідної $\frac{df}{dt}$ у цій точці, яка є неперервним оператором, який визначається співвідношенням

$$f'(t) = \frac{df}{dt} = \lim_{\Delta t \to 0} \frac{\Delta f}{\Delta t}, \tag{9.4}$$

яке не може мати місця для решітчастих функцій. Але оскільки сусідні значення аргументу kT решітчастої функції $f[kT]$ відрізняються на T, то, сформувавши різницю

$$\Delta f[kT] = f[(k + 1)T] - f[kT], \ \ k = 0,1,2, \ldots \tag{9.5}$$

ми можемо формально використати цю різницю Δf[kT] для характеристики швидкості зміни решітчастої функції $f[kT]$ у точці з аргументом kT. Різницю $\Delta f[kT]$ називають правою скінченною різницею першого порядку решітчастої функції $f[kT]$ або правим різницевим оператором першого порядку.

За аналогією з (9.5) можна визначити праву скінченну різницю другого і наступного порядків у вигляді

$$\Delta^n f[kT] = \Delta^{n-1} f[(k + 1)T] - \Delta^{n-1} f[kT], \tag{9.6}$$

$$n = 1,2, \ldots, \quad k = 0,1,2, \ldots,$$

пам'ятаючи при цьому, що

$$\Delta^0 f[kT] = f[kT] \tag{9.7}$$

Підставляючи у формулу (9.6) $n = 2$, легко переконатись у тому, що отримаємо

$$\Delta^2 f[kT] = f[(k+2)T] - 2f[(k+1)T] + f[kT], \tag{9.8}$$
$$k = 0, 1, 2, \ldots.$$

тобто, легко переконатись у тому, що правий різницевий оператор другого порядку може бути виражений через алгебраїчну суму трьох значень решітчастої функції, взятих з наростаючим аргументом. Узагальнюючи цей висновок, можна стверджувати, що і правий різницевий оператор *n-ого* порядку може бути виражений через алгебраїчну суму *(n+1)-го* значень решітчастої функції, взятих з наростаючим аргументом.

Відомо, що значення похідної $f'(t)$ від неперервної функції $f(t)$ у точці з аргументом t не залежить від того зліва чи справа наближається Δt до нуля (див. формулу (9.4)). Інша ситуація виникає при наближенні до точки з аргументом kT під час аналізу решітчастої функції $f[kT]$, а тому для характеристики швидкості її зміни крім правої скінченної різниці $\Delta f[kT]$ першого порядку доцільно ввести ще й ліву скінченну різницю першого порядку за виразом

$$\nabla f[kT] = f[kT] - f[(k-1)T], \quad k = 1,2, \ldots \tag{9.9}$$

Цю різницю називають лівим різницевим оператором першого порядку.

Нагадаємо, що символи «Δ» та «∇» є грецькими літерами, відповідно, «дельта» та «набла».

За аналогією з (9.6) визначається і ліва скінченна різниця порядку n

$$\nabla^n f[kT] = \nabla^{n-1} f[kT] - \nabla^{n-1} f[(k-1)T], \tag{9.10}$$
$$n = 1, 2, \ldots, \quad k = 0, 1, 2, \ldots,$$

із якої при n = 2 отримаємо

$$\nabla^2 f[kT] = f[kT] - 2f[(k-1)T] + f[(k-2)T], \tag{9.11}$$
$$k = 0, 1, 2, \ldots.$$

що дозволяє нам переконатись у тому, що і лівий різницевий оператор другого порядку може бути виражений через алгебраїчну суму трьох значень решітчастої функції, але взятих зі спадним аргументом. Узагальнюючи цей висновок, можна стверджувати також, що і лівий різницевий оператор *n-ого* порядку може бути виражений через алгебраїчну суму $(n+1)$-го значень решітчастої функції, взятих зі спадним аргументом.

Оскільки аналогом похідної *n-го* порядку, де $n = 1, 2, \ldots$, для решітчастої функції $f[kT]$ є права та ліва скінченні різниці того ж порядку, то для решітчастих функцій $x[kT]$ та $y[kT]$ можна сконструювати рівняння в скінченних різницях $\Delta^i y[kT]$ ($i = 0, 1, 2, \ldots, n$), $\Delta^q x[kT]$ ($q = 0, 1, 2, \ldots, m$) або $\nabla^i y[kT]$ ($i = 0, 1, 2, \ldots, n$), $\nabla^q x[kT]$ ($q = 0, 1, 2, \ldots, m$), яке буде відображенням диференціального рівняння, синтезованого відносно функцій y(t), x(t) та їх похідних того ж порядку.

Покажемо на прикладі як можна побудувати дискретне відображення диференціального рівняння. Нехай маємо диференціальне рівняння

$$\frac{d^2y}{dt^2} + 4\frac{dy}{dt} + 2y = 2x, \tag{9.12}$$

з початковими умовами

$$\begin{cases} y(0) = 0, \\ y'(0) = 1. \end{cases} \tag{9.13}$$

Нехай T – період дискретності функцій $x(t)$, $y(t)$ для породження решітчастих функцій $x[kT]$, $y[kT]$. Нагадаємо, що аргументи t і kT пов'язані між собою співвідношенням

$$t_k = kT, \quad k = 0,1,2,\ldots \tag{9.14}$$

Зрозуміло, що похідній, яку маємо у рівнянні (9.12), в її дискретному відображенні відповідатиме вираз

$$\frac{\Delta y(t_k)}{\Delta t_k} = \frac{\Delta y[kT]}{(k+1)T-kT} = \frac{\Delta y[kT]}{T}, \tag{9.15}$$

а другій похідній – вираз

$$\frac{\Delta^2 y(t_k)}{(\Delta t_k)^2} = \frac{\Delta^2 y[kT]}{T^2}. \tag{9.16}$$

Підставляючи (9.14)–(9.16) у рівняння (9.12), (9.13), отримаємо:

$$\frac{\Delta^2 y[kT]}{T^2} + 4\frac{\Delta y[kT]}{T} + 2y[kT] = 2x[kT], \quad k = 0,1,2,\ldots \tag{9.17}$$

$$\begin{cases} y[0] = 0, \\ \frac{\Delta y[0]}{T} = 1 \end{cases} \tag{9.18}$$

Рівняння (9.17) є рівнянням в скінченних різницях, породженим диференціальним рівнянням (9.12), а рівняння (9.18) – це початкові умови для рівняння в скінченних різницях, породжені початковими умовами (9.13), заданими для диференціального рівняння (9.12). Зрозуміло, що чим меншим буде значення періоду дискретності T, тим менше будуть відрізнятись один від одного значення t_k та t_{k+1} або t_{k-1} і, як наслідок, ближчими до розв'язку $y(t)$ диференціального рівняння (9.12) у точках tk будуть розв'язки $y[kT]$ рівняння в скінченних різницях (9.17), яке практично у такому вигляді ніхто не розв'язує, а розв'язують його після перетворення в різницеве рівняння, яке містить у собі не скінченні різниці, а значення решітчастих функцій, взятих при декількох значеннях аргументу. Для здійснення цього перетворення потрібно у рівняннях (9.17), (9.18) замість скінченних різниць $\Delta y[kT]$, $\Delta^2 y[\mathrm{kT}]$ підставити їхні значення з формул (9.5), (9.8) та домножити усі члени рівняння (9.17) на T^2, а друге рівняння у системі (9.18) домножити на T. В результаті цих дій, приведення подібних членів та перенесення усіх членів, окрім першого, в праву частину рівняння, матимемо:

$$\begin{gathered} y[(k+2)T] = 2T^2 x[kT] - (4T-2)y[(k+1)T] - \\ -(2T^2 - 4T + 1)y[kT], \quad k = 0,1,2,\ldots \end{gathered} \tag{9.19}$$

$$\begin{cases} y[0] = 0, \\ y[1 \cdot T] = T \end{cases} \tag{9.20}$$

Вибравши конкретне значення T та сформувавши з заданої функції $x(t)$ решітчасту $x[kT]$, шляхом підстановки по черзі $k = 0$, потім $k = 1$ і далі у рівняння (9.19) отримаємо стільки значень розв'язку $y[kT]$, скільки нам потрібно.

У попередньому розділі було показано, що лінійні диференціальні рівняння можна розв'язувати не лише безпосередньо, але й завдяки попередньому їх перетворенню в алгебраїчні рівняння комплексної змінної з застосуванням прямого та оберненого операторів Лапласа.

Покажемо, що аналогічну процедуру можна застосувати і до різницевих рівнянь.

Нехай $y[kT]$ ($k = 0, 1, 2, \ldots$) є решітчастою функцією з періодом дискретності T, породженою неперервною функцією-оригіналом $y(t)$, для якої, як нам уже відомо, справедливо

$$y(t) = \begin{cases} y(t), & \forall t \geq 0, \\ 0, & \forall t < 0, \end{cases} \tag{9.21}$$

$$Y(p) = \int_0^\infty y(t)e^{-pt}dt \tag{9.22}$$

Оскільки для решітчастої функції мінімальний приріст аргументу *dt* дорівнює періоду дискретності *T*, то, з урахуванням виразу (9.14) у наближеній формі з певною похибкою, оператор Лапласа (9.22) для решітчастої функції можна записати у вигляді

$$Y(p) \cong \sum_{k=0}^{\infty} y[kT]e^{-pkT}T \tag{9.23}$$

Введемо нову комплексну змінну q як

$$q = pT. \tag{9.24}$$

і перепишемо вираз (9.24) так

$$\frac{1}{T}Y\left(\frac{q}{T}\right) = \sum_{k=0}^{\infty} y[kT]e^{-qk}. \tag{9.25}$$

Формула (9.25) задає дискретний оператор, який уже не є оператором Лапласа, але який решітчасту функцію *y*[*kT*], тобто функцію, визначену на множині міри «нуль», відображає у функцію *Y*(*q*), нормалізовану періодом дискретності *T* з масштабованою цим же періодом дискретності комплексною змінною *q*.

Проаналізувавши праву частину виразу (9.25), Густав Деч прийшов до висновку, що введенням в її структуру заміни

$$e^q = z \tag{9.26}$$

можна сконструювати новий оператор

$$Z\{y[kT]\} = \sum_{k=0}^{\infty} y[kT]z^{-k} = Y(z), \tag{9.27}$$

який відображає решітчасті функції *y*[*kT*], визначені на нульвимірному просторі дискретних значень аргументу *kT*, у точки за межами круга одиничного радіуса на комплексній площині *z*, кожній з яких відповідає неперервна функція *Y*(*z*).

І оскільки саме Густав Деч запропонував здійснювати це перетворення з використанням оператора (9.27), ми вважаємо, що цей оператор можна назвати оператором Деча, хоча він сам, звичайно ж, цей оператор своїм іменем не називав, а називав *Z*-перетворенням, як не називали Лаплас і Фур'є самі своїми іменами добре відомі нині усім математикам та інженерам оператори Лапласа і Фур'є, що розглянуті нами у попередньому розділі. Варто зауважити, що дехто з математиків оператор (9.27) називає оператором Лорана, виходячи з того, що ряд, яким цей оператор відображається, є частиною ряду Лорана, але звертаємо увагу на те, що присвоєння цьому оператору імені Лорана є некоректним, оскільки ряд Лорана містить складові не лише з від'ємними степенями змінної ***z,*** але і з її додатними степенями також.

Як приклад знайдемо *Z*-перетворення одиничної решітчастої функції

$$1[\text{kT}] = \begin{cases} 1, & k = 0,1,2,\ldots \\ 0, & k < 0 \end{cases}, \tag{9.28}$$

Підставляючи у вираз (9.27) *y*[*kT*] = 1[*kT*] і знаходячи суму членів нескінченно спадної геометричної прогресії зі знаменником z^{-1}, матимемо

$$1(z) = Z\{1[kT]\} = \sum_{k=0}^{\infty} 1[k]z^{-k} = 1 + z^{-1} + z^{-2} + \cdots = \frac{1}{1-z^{-1}}. \tag{9.29}$$

Під час відображення різницевих рівнянь з використанням *Z*-оператора доводиться знаходити *Z*-зображення зсунутих вправо чи вліво значень решітчастих функцій, відносно

яких ці рівняння складені, а тому у нашому базовому навчальному посібнику [2] наведені алгоритми їх знаходження для обох випадків.

Але оскільки різницеві рівняння, які отримані з диференціальних рівнянь застосуванням правого різницевого оператора (9.6), приводяться до рівнянь у формі (9.9) і використовуються для прогнозування розвитку тренду процесу, що формується на виході інерційного динамічного об'єкта, який фільтрує завади випадкового характеру, то застосовувати *Z*-перетворення для їх розв'язання недоцільно. А тому є сенс розглядати *Z*-перетворення лише решітчастих функцій, зсунутих вліво. І тут ми наведемо лише кінцевий результат, відсилаючи тих, хто хоче переконатись, що це саме так, до нашого базового навального посібника [2].

Отже, при зсуві вліво решітчастої функції на n кроків (періодів дискретизації) її *Z*-зображення можна знайти з виразу

$$Z\{y[(k-n)T]\} = z^{-n}Y(z). \tag{9.30}$$

Розглянемо процедуру Z-перетворення різницевого рівняння на конкретному прикладі

Нехай різницеве рівняння має вигляд

$$y[kT] - 5y[(k-1)T] + 2y[(k-2)] = 4x[kT] - 3x[(k-1)T] \tag{9.31}$$

з нульовими значеннями функції *y[kT]* при від'ємних значеннях аргументу

Перетворюючи обидві частини різницевого рівняння (9.31) з використанням Z-оператора (9.27) та виразу (9.30), отримаємо

$$Y(z) - 5\,z^{-1}Y(z) + 2\,z^{-2}Y(z) = 4\,X(z) - 3\,z^{-1}X(z). \tag{9.32}$$

А виносячи за дужки в лівій частині *Y(z)*, а у правій частині *X(z)*, із рівняння (9.32) матимемо розв'язок різницевого рівняння (9.31) на комплексній площині у вигляді

$$Y(z) = \frac{4-3\,z^{-1}}{1-5\,z^{-1}+2\,z^{-2}}X(z), \tag{9.33}$$

або

$$Y(z) = W(z)\,X(z), \tag{9.34}$$

де

$$W(z) = \frac{4-3\,z^{-1}}{1-5\,z^{-1}+2\,z^{-2}} \tag{9.35}$$

– дискретна передаточна функція динамічної системи, процес в якій в дискретному часі описується різницевим рівнянням (9.31).

Нехай

$$x[kT] = 5\ 1[kT]. \tag{9.36}$$

Тоді, згідно з (9.29),

$$X(z) = \frac{5}{1-z^{-1}}. \tag{9.37}$$

Підставляючи вираз (9.37) у (9.33), отримаємо

$$Y(z) = \frac{20-15\,z^{-1}}{1-6\,z^{-1}+7\,z^{-2}-2z^{-3}}. \tag{9.38}$$

Вираз (9.38) являє собою математичну модель реакції динамічної системи (9.31) на вхідний сигнал (9.36), синтезовану на комплексній площині. Тобто, вона є *Z*-зображенням цієї реакції.

Формально, для знаходження оригіналу за відомим зображенням потрібно використовувати обернений оператор, у даному випадку

$$Z^{-1}\{Y(z)\}. \tag{9.39}$$

Для того, щоб усвідомити, якою є найпростіша форма оберненого оператора (9.39), запропонована Густавом Дечем, розпишемо спочатку суму у формулі Z-перетворення в (9.27) у вигляді

$$Y(z) = \sum_{k=0}^{\infty} y[kT]z^{-k} = y[0] + y[1 \cdot T]z^{-1} + y[2T]z^{-2} + \cdots \tag{9.40}$$

З виразу (9.40) бачимо, що коефіцієнти при степенях z^{-k}, (k = 0, 1, …) і є значеннями решітчастої функції y[kT], яка є оригіналом для Z-зображення Y(z). А до форми (9.40) дробово-раціональна функція, що стоїть у правій частині виразу (9.38), яка є відношенням двох багаточленів, приводиться діленням чисельника на знаменник і взяттям такої кількості членів ряду, яка необхідна за умовами розв'язання. (Приклад наведено у нашому базовому навчальному посібнику [2])

Оскільки процес ділення багаточлена на багаточлен легко алгоритмізується, то на комп'ютері дуже швидко можна отримати яку завгодно сукупність значень решітчастого оригіналу на тому відрізку дискретних значень аргументу, який нас цікавить.

Але, щоб отримати багаточлени, які потрібно ділити, тобто, щоб привести вираз (9.34) до вигляду (9.38), потрібно знати Z-зображення X(z) вхідних сигналів x[kT]. Ця задача легко розв'язується, якщо скористатись уже готовими виразами для вхідних сигналів, наведеними Густавом Дечем в таблиці 37.1 його роботи [7], найбільш вживану у практиці частину з яких ми наводимо нижче у нашій таблиці 9.1.

Таблиця 9.1

$x[k]$	$X(z)$
$1[k]$	$\frac{z}{z-1}$
k	$\frac{z}{(z-1)^2}$
k^2	$\frac{z(z+1)}{(z-1)^3}$
a^k	$\frac{z}{z-a}$
$k\,a^{k-1}$	$\frac{z}{(z-a)^2}$
$a^k sinkt$	$\frac{a\,z\,sin\,t}{z^2-2a\,z\,cos\,t+a^2}$
$a^k \cos kt$	$\frac{z(z-\mathrm{a}\cos t)}{z^2-2a\,z\,cos\,t+a^2}$
$a^k sh\,kt$	$\frac{az\,sh\,t}{z^2-2a\,z\,ch\,t+a^2}$
$a^k ch\,kt$	$\frac{z(z-a\,ch\,t)}{z^2-2a\,z\,ch\,t+a^2}$
$\frac{1}{k}$, k=1,2,…	$ln\frac{1}{z-1}$
$\frac{a^k}{k!}$	$e^{\frac{a}{z}}$

Отже ми, скориставшись ідеєю Густава Деча, показали, яким чином кожній неперервній функції $Y(z)$, образом якої є точка на комплексній площині, окрім розміщених в крузі одиничного радіуса, можна поставити у відповідність решітчасту функцію $y[kT]$, $k = 0, 1, 2, ...$ дискретного аргументу на осі дійсних чисел, тобто показали, як практично реалізується обернений оператор Деча (9.39).

9.2 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з застосуванням дискретних операторів класу різницевих

Для отримання ***Z***-зображення ***Y*(*z*)** решітчастої функції ***y*[*kT*], *T* > 0, *k* = 0, 1, …** з використанням оператора Деча (9.27) мовою ***Python*** інформації з цієї мови, викладеної у попередніх підрозділах частини 2 та у частині 1 нашого навчального посібника, цілком достатньо, тому ми її повторювати не будемо, а продемонструємо, як її застосовувати у наступному підрозділі у Програмі 28. А ось для отримання значень цієї решітчастої функції ***y*[*kT*], *T* > 0, *k* = 0, 1, …** за відомим її ***Z***-зображенням ***Y(z)*** з використанням оберненого оператора Деча $\boldsymbol{Z^{-1}\{Y(z)\}}$ у вигляді (9.39), в якому використовується операція ділення многочлена на многочлен, нам потрібна буде нова інформація, присвячена тому, як мовою ***Python*** здійснюється ділення многочлена на многочлен. Функція ділення многочлена ***f*(*z*) = *g*(*z*) * *q*(*z*) + *r*(*z*)** на многочлен ***g*(*z*)** входить до ППП ***sympy*** і має вигляд ***q,r = div(f, g, domain = QQ')***, в результаті виконання якої отримуємо функцію ***q*(*z*)** як частку від ділення та функцію ***r*(*z*),** як залишок, що залишився нерозділеним. Роздрукувати результати ділення многочлена на многочлен можна функціями ***print*(*q*), *print*(*r*).** Як працює функція ділення многочлена на многочлен продемонстровано у пояснювальному прикладі № 13.

Пояснювальний приклад № 13

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import*                            # Виклик із sympy усіх функцій
In [3]: z = symbols ('z')                             # Символізація змінної z
In [4]: f = Function ('f')(z)                         # Символізація многочлена f(z)
In [5]: g = Function ('g')(z)                         # Символізація многочлена g(z)
In [6]: q = Function ('q')(z)                         # Символізація частки q(z)
In [7]: r = Function ('r')(z)                         # Символізація залишку r(z)
In [8]: f = 1.25*z**3-z**2                            # Формування многочлена f(z)
In [9]: g = 6*z**3-12.5*z**2+8.5*z-2                  # Формування многочлена g(z)
In [10]: q,r = div(f,g,domain = 'QQ')                 # Ділення f(z) на g(z)
In [11]: print(q)                                     # Друкування частки q(z)
5/24
In [12]: print(r)                                     # Друкування залишку r(z)
77*z**2/48 - 85*z/48 + 5/12
In [13]: f1 = r                                       # Надання залишку r(z)
                                                      ….статусу нового діленого f1(z)
In [14]: q1,r1 = div(f1,g,domain = 'QQ')              # Ділення f1(z) на g(z)
In [15]: print(q1)                                    # Друкування частки q1(z)
0
In [16]: print(r1)                                    # Друкування залишку r1(z)
77*z**2/48 - 85*z/48 + 5/12
```

Кінець пояснювального прикладу № 13

Як бачимо з пояснювального прикладу № 13, ділити многочлен, який є діленим, на многочлен, який є дільником, можна лише доти, доки найвищий показник степеня незалежної змінної у діленому є не нижчим найвищого показника степеня у дільника, а як тільки найвищий показник степеня у дільника стає вищим найвищого показника степеня у діленого, операція ділення многочлена на многочлен функцією ***sympy.div( )*** припиняється. Отже, скористатись цією стандартною функцією з ППП ***sympy*** для отримання значень решітчастої функції ***y[kT]*** із її ***Z***-зображення ***Y(z)*** з використанням оберненого оператора Деча у формі ділення многочлена на многочлен не вдасться. Тому будемо створювати власну програму для реалізації цієї форми оберненого оператора Деча. Але для цього нам знадобиться нова інформація стосовно застосування мови ***Python,*** яка ще не подавалась як у частині першій нашого навчального посібника, так і у попередніх підрозділах частини другої. Нам знадобиться такий метод в ППП ***sympy***, як ***f.as_ordered_terms( )***, який символьний вираз ***f*** перетворює у список операндів ***L1 = [, , ,],*** розташованих в порядку їх розміщення у символьному виразі ***f***, а символьний вираз ***g*** цей метод ***g.as_ordered_terms( )*** перетворює у список операндів ***L2 = [, , ,],*** розташованих в порядку їх розміщення у символьному виразі ***g***. Перетворивши у списки многочлени, які потрібно ділити, та викликавши за індексами перші члени цих списків ***L1[0],L2[0],*** ми розділимо перший член діленого на перший член дільника і на цей результат ***L1[0]/L2[0],*** який є першим членом ***a*** частки від ділення, домножимо дільник, тобто, отримаємо вираз ***a*g***. Потім віднімемо цей добуток від символьного виразу ***f*** та створимо новий вираз для діленого у вигляді ***f1 = f – a * g***, яке матиме показник степеня при змінній ***z*** на одиницю менший, та яке, знову ж таки, методом ***f1.as_ordered_terms()*** перетворимо у список операндів ***L11***, із якого за індексом викличемо перший член ***L11[0],*** який розділимо знову на ***L2[0]***, отримавши у такий спосіб другий член ***b*** частки від ділення – і далі процедуру повторюватимемо стільки разів, скільки членів з числовими коефіцієнтами у частці від ділення ми побажаємо мати. Як працює викладений вище алгоритм ділення многочлена на многочлен продемонстровано у пояснювальному прикладі № 14.

Пояснювальний приклад № 14

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import*                            # Доступ до усіх команд sympy
In [3]: z = symbols ('z')                             # Символізація змінної z
In [4]: f = Function ('f:6')(z)                       # Оголошення символьним
                                                      ….многочлена f з індексами
In [5]: g = Function ('g')(z)                         # Символізація многочлена g
In [6]: h = Function ('h')(z)                         # Символізація многочлена h
In [7]: r = Function ('r')(z)                         # Оголошення символьним
                                                      ….залишку r від ділення
In [8]: a,b,c = symbols ('a b c')                     # Оголошення символьними
                                                      ….складових a,b,c частки
In [9]: f0 = 0.75-z**(-1)                             # Формування діленого f0
In [10]: g = 3-6*z**(-1)+4*z**(-2)-z**(-3)            # Формування дільника g
In [11]: h = a+b+c+r                                  # Формування структури частки
In [12]: L1 = f0.as_ordered_terms( );L1               # Формування списку L1 із
Out[12]: [0.750000000000000, -1/z]                    ….складових многочлена f0
In [13]: L2 = g.as_ordered_terms( );L2                # Формування списку L2 зі
Out[13]: [3, -6/z, 4/z**2, -1/z**3]                   ….складових многочлена g
In [14]: L1[0]                                        # Виклик першого члена
```

```
Out[14]:                                  ….списку L1
0.750000000000000
In [15]: L2[0]                            # Виклик першого члена
Out[15]:                                  ….списку L2
3
In [16]: a=L1[0]/L2[0];a                  # Визначення першого
Out[16]:                                  ….члена a частки від
0.250000000000000                         ….ділення
In [17]: a*g                              # Обчислення результату
Out[17]:                                  ….множення a*g першого
0.75 - 1.5/z + 1.0/z**2 - 0.25/z**3       ….члена частки на дільник
In [18]: f1 = f0-a*g;f1                   # Формування нового
Out[18]:                                  ….виразу f1 для діленого
0.5/z - 1.0/z**2 + 0.25/z**3
In [19]: L11 = f1.as_ordered_terms( );L11 # Формування списку L11 із
Out[19]:                                  ….складових многочлена f1
[0.5/z, -1.0/z**2, 0.25/z**3]
In [20]: L11[0]                           # Виклик першого члена
Out[20]:                                  ….списку L11
0.5/z
In [21]: b=L11[0]/L2[0];b                 # Визначення другого
Out[21]:                                  ….члена b частки від
0.166666666666667/z                       ….ділення
In [22]: b*g                              # Обчислення результату
Out[22]:                                  ….множення b*g другого
0.166666666666667*(3 - 6/z + 4/z**2 –     ….члена частки на дільник
1/z**3)/z
In [23]: f2=f1-b*g;f2                     # Формування нового
Out[23]:                                  ….виразу f2 для діленого
-0.166666666666667*(3 - 6/z + 4/z**2 –
 1/z**3)/z + 0.5/z - 1.0/z**2 + 0.25/z**3
In [24]: f3=expand(f2);f3                 # Розкриття дужок у виразі f2
Out[24]:                                  ….та його спрощення
-0.416666666666667/z**3 +                 ….до вигляду f3
 0.166666666666667/z**4
In [25]: L111 = f3.as_ordered_terms( );L111 # Формування списку L111 із
Out[25]: [-0.416666666666667/z**3,        ….складових многочлена f3
 0.166666666666667/z**4]
In [26]: L111[0]                          # Виклик першого члена
Out[26]:                                  ….списку L111
-0.416666666666667/z**3
In [27]: c=L111[0]/L2[0];c                # Визначення третього
Out[27]:                                  ….члена c частки від
-0.138888888888889/z**3                   ….ділення
In [28]: c*g                              # Обчислення результату
Out[28]:                                  ….множення c*g третього
```

```
-0.138888888888889*(3 - 6/z + 4/z**2 –                ….члена частки на дільник
1/z**3)/z**3
In [29]: f4=f3-c*g;f4                                 # Формування нового
Out[29]:                                              ….виразу f4 для діленого
0.138888888888889*(3 - 6/z + 4/z**2 –
1/z**3)/z**3 - 0.416666666666667/z**3 +\
 0.166666666666667/z**4
In [30]: f5=expand(f4);f5                             # Розкриття дужок у виразі f4
Out[30]:                                              ….та його спрощення
-0.666666666666667/z**4 +\                            ….до вигляду f5
 0.555555555555555/z**5 - \
0.138888888888889/z**6
In [31]: r=f5/g;r                                     # Визначення залишку r
Out[33]:                                              ….після трьох кроків ділення
(-0.666666666666667/z**4 +                            ….у вигляді відношення,
 0.555555555555555/z**5 –                             ….підготовленого до
0.138888888888889/z**6)/(3 - 6/z +                    ….наступного кроку діленого
4/z**2 - 1/z**3)                                      ….f5 до дільника g
In [34]: h=a+b+c+r;h                                  # Формування частки від ділення
Out[34]:                                              ….у вигляді суми результатів
(-0.666666666666667/z**4 +                            ….трьох кроків ділення та
 0.555555555555555/z**5 –                             ….залишку, який за потреби
 0.138888888888889/z**6)/(3 - 6/z + 4/z**2            ….можна продовжувати
-1/z**3) + 0.25 + 0.166666666666667/z -\              ….ділити за цим же алгоритмом
 0.138888888888889/z**3
```

Кінець пояснювального прикладу № 14

9.3 Задачі щодо застосування дискретних операторів класу різницевих в програмах мовою Python

Програма мовою Python для розв'язання задач, пов'язаних з перетворенням диференціального рівняння

$$a_3\frac{d^3y}{dt^3}+a_2\frac{d^2y}{dt^2}+a_1\frac{dy}{dt}+a_0y=b_2\frac{d^2x}{dt^2}+b_1\frac{dx}{dt}+b_0x$$

в різницеве рівняння

$$c_0y[k]+c_1y[k-1]+c_2y[k-2]+c_3y[k-3]=$$
$$=d_0x[k]+d_1x[k-1]+d_2x[k-2]$$

та Z-зображенням цього різницевого рівняння у вигляді

$$Y(z)=\frac{d_0+d_1z^{-1}+d_2z^{-2}}{c_0+c_1z^{-1}+c_2z^{-2}+c_3z^{-3}}X(z).$$

(Програма 28)

```
In [1]: import sympy                                  # Виклик ППП sympy
In [2]: from sympy import*                            # Виклик із sympy усіх функцій
In [3]: y = symbols('y:4')                            # Індексна символізація y
In [4]: L1 = list(y);L1                               # Створення списку значень
Out[4]: [y0, y1, y2, y3]                              ….змінної y та їх визначення
In [5]: y0=L1[0];y1=L1[1];y2=L1[2];y3=L1[3]           ….за місцем у списку
In [6]: T = symbols('T')                              # Оголошення символьним
                                                      ….періоду дискретизації T
In [7]: h = Function('h:6')(y)                        # Символізація функції h(y)
In [8]: h0 = y0-y1                                    # Визначення першої різниці
                                                      ….функції h(y)
In [9]: h1 = h0/T                                     # Визначення дискретного
                                                      ….аналогу першої похідної
In [10]: h2 = y0-2*y1+y2                              # Визначення другої різниці
                                                      ….функції h(y)
In [11]: h3 = h2/T**2                                 # Визначення дискретного
                                                      ….аналогу другої похідної
In [12]: h4 = y0-3*y1+3*y2-y3                         # Визначення третьої різниці
                                                      ….функції h(y)
In [13]: h5 = h4/T**3                                 # Визначення дискретного
                                                      ….аналогу третьої похідної
In [14]: a = symbols('a:4')                           # Індексна символізація a
In [15]: L2 = list(a);L2                              # Створення списку значень
Out[15]: [a0, a1, a2, a3]                             ….коефіцієнтів a та їх визначення
In [16]: a0 = L2[0]; a1 = L2[1]; a2 = L2[2]; \        ….за місцем у списку
a3 = L2[3]
In [17]: lsr = Function('lsr:5')(y)                   # Символізація виразу lsr(y)
In [18]: lsr0 = a3*h5+a2*h3+a1*h1+a0*y0; lsr0         # Трансформація лівої
Out[18]:                                              ….частини диференціального
a0*y0 + a1*(y0 - y1)/T + a2*(y0 - 2*y1 +              ….рівняння у вираз lsr0, що
 y2)/T**2 + a3*(y0 - 3*y1 + 3*y2 - y3)/T**3           ….містить ліві кінцеві різниці
In [19]: lsr1=lsr0.expand();lsr1                      # Розкриття дужок у виразі
Out[19]:                                              ….в кінцевих різницях lsr0
a0*y0 + a1*y0/T - a1*y1/T + a2*y0/T**2 -\             ….та його приведення у
 2*a2*y1/T**2 + a2*y2/T**2 + a3*y0/T**3 - \           ….форму неупорядкованого
3*a3*y1/T**3 + 3*a3*y2/T**3 - a3*y3/T**3              ….різницевого виразу lsr1
In [20]: lsr2=lsr1.collect(y0);lsr2                   # Перший крок до
Out[20]:                                              ….упорядкування виразу lsr1
y0*(a0 + a1/T + a2/T**2 + a3/T**3) - \                ….з переведенням його у
a1*y1/T - 2*a2*y1/T**2 + a2*y2/T**2 -\                ….форму lsr2 шляхом
 3*a3*y1/T**3 + 3*a3*y2/T**3 - a3*y3/T**3             ….збирання коефіцієнтів біля y0
In [21]: lsr3=lsr2.collect(y1);lsr3                   # Другий крок до
Out[21]:                                              ….упорядкування виразу lsr1
y0*(a0 + a1/T + a2/T**2 + a3/T**3) + \                ….з переведенням його у
y1*(-a1/T - 2*a2/T**2 - 3*a3/T**3) + \                ….форму lsr3 шляхом
```

```
a2*y2/T**2 + 3*a3*y2/T**3 - a3*y3/T**3                  ….збирання коефіцієнтів біля y1
In [22]: lsr4=lsr3.collect(y2);lsr4                     # Третій крок до
Out[22]:                                                ….упорядкування виразу lsr1
y0*(a0 + a1/T + a2/T**2 + a3/T**3) + \                  ….з переведенням його у
y1*(-a1/T - 2*a2/T**2 - 3*a3/T**3) + \                  ….форму lsr4 шляхом
y2*(a2/T**2 + 3*a3/T**3) - a3*y3/T**3                   ….збирання коефіцієнтів біля y2
In [23]: c = symbols('c:4')                             # Індексна символізація c
In [24]: L11 = lsr4.as_ordered_terms( );L11             # Трансформація символьного
Out[24]:                                                ….виразу lsr4 в список L11 його
[y0*(a0 + a1/T + a2/T**2 + a3/T**3),                    ….складових
 y1*(-a1/T - 2*a2/T**2 - 3*a3/T**3),
 y2*(a2/T**2 + 3*a3/T**3),
 -a3*y3/T**3]
In [25]: c0 = L11[0]/y0;c0                              # Визначення коефіцієнта c0
Out[25]:                                                ….шляхом ділення на y0
a0 + a1/T + a2/T**2 + a3/T**3                           ….першого члена списку L11
In [26]: c1 = L11[1]/y1;c1                              # Визначення коефіцієнта c1
Out[26]:                                                ….шляхом ділення на y1
-a1/T - 2*a2/T**2 - 3*a3/T**3                           ….другого члена списку L11
In [27]: c2 = L11[2]/y2;c2                              # Визначення коефіцієнта c2
Out[27]:                                                ….шляхом ділення на y2
a2/T**2 + 3*a3/T**3                                     ….третього члена списку L11
In [28]: c3 = L11[3]/y3;c3                              # Визначення коефіцієнта c3
Out[28]:                                                ….шляхом ділення на y3
-a3/T**3                                                ….четвертого члена списку L11
In [29]: Y = symbols ('Y:4');Y                          # Індексна символізація
Out[29]: (Y0, Y1, Y2, Y3)                               ….змінної Y
In [30]: L111=list(Y);L111                              # Створення списку значень
Out[30]: [Y0, Y1, Y2, Y3]                               ….змінної Y та їх визначення
In [31]: Y0=L111[0];Y1=L111[1];Y2=L111[2];              ….за місцем у списку
Y3=L111[3]
In [32]: z = symbols('z')                               # Оголошення символьною z
In [33]: Y1=z**(-1)*Y0                                  # Визначення функції Y1(z)
In [34]: Y2=z**(-2)*Y0                                  # Визначення функції Y2(z)
In [35}: Y3=z**(-3)*Y0                                  # Визначення функції Y3(z)
In [36]: Q = Function ('Q')(z)                          # Символізація виразу Q(z)
In [37]: Q =c0*Y0+c1*Y1+c2*Y2+c3*Y3;Q                   # Трансформація виразу
Out[37]:                                                ….різницевого lsr4 на площину
Y0*(a0 + a1/T + a2/T**2 + a3/T**3) + Y0*(-a1/T          ….комплексну у вигляді
 - 2*a2/T**2 - 3*a3/T**3)/z + Y0*(a2/T**2 +             ….функції комплексної
3*a3/T**3)/z**2 - Y0*a3/(T**3*z**3)                     ….змінної z
In [38]: F2=Function('F2')(z)                           # Символізація виразу F2(z)
In [39]: F2=Q.subs(Y0,1);F2                             # Визначення виразу F2(z)
Out[39]:                                                ….шляхом підстановки у
a0 + (-a1/T - 2*a2/T**2 - 3*a3/T**3)/z +                ….вираз Q(z) замість
 (a2/T**2 + 3*a3/T**3)/z**2 + a1/T +                    ….змінної Y0 її одиничного
```

```
 a2/T**2 + a3/T**3 - a3/(T**3*z**3)
In [40]: a0=4;a1=8;a2=4;a3=1
In [41]: T=0.5
In [42]: d11=dict({"T":0.5});d11
Out[42]: {'T': 0.5}
In [43]: F21=F2.subs([(L2[0],4),(L2[1],8),
(L2[2],4),(L2[3],1)]);F21
Out[43]:
4 + (-8/T - 8/T**2 - 3/T**3)/z + (4/T**2 +
3/T**3)/z**2 + 8/T + 4/T**2 + T**(-3) –
1/(T**3*z**3)
In [44]: F22=F21.subs({"T":0.5});F22
Out[44]:
44.0 - 72.0/z + 40.0/z**2 - 8.0/z**3



In [45]: x = symbols('x:3')
In [46]: L3 = list(x);L3
Out[46]: [x0, x1, x2]
In [47]: x0=L3[0];x1=L3[1];x2=L3[2]
In [48]: T1 = symbols('T1')

In [49]: m = Function('m:4')(x)
In [50]: m0 = x0-x1

In [51]: m1 = m0/T1

In [52]: m2 = x0-2*x1+x2

In [53]: m3 = m2/T1**2

In [54]: b = symbols('b:3')
In [55]: L4 = list(b);L4
Out[55]: [b0, b1, b2]
In [56]: b0 = L4[0]; b1 = L4[1]; b2 = L4[2]
In [57]: psr = Function('psr:4')(x)
In [58]: psr0 = b2*m3+b1*m1+b0*x0; psr0
Out[58]:
b0*x0 + b1*(x0 - x1)/T1 + b2*(x0 - 2*x1 +
x2)/T1**2
In [59]: psr1=psr0.expand();psr1
Out[59]:
b0*x0 + b1*x0/T1 - b1*x1/T1 + b2*x0/T1**2
- 2*b2*x1/T1**2 + b2*x2/T1**2

In [60]: psr2=psr1.collect(x0);psr2
```

….значення
Внесення значень ***a***
….та ***T***
Створення словника ***d11***
….для ***T***
Трансформація виразу
….***F2(z)*** у вираз ***F21(z)***
….шляхом підстановки
….числових значень
….коефіцієнтів ***a***

Трансформація виразу
….***F21(z)*** у вираз ***F22(z)***, який є
….знаменником дискретної
….передаточної функції, шляхом
….підстановки значення ***T***
Індексна символізація ***x***
Створення списку значень
….змінної ***x*** та їх визначення
….за місцем у списку
Оголошення символьним
….періоду дискретизації ***T1***
Символізація функції ***m(x)***
Визначення першої різниці
….функції ***m(x)***
Визначення дискретного
….аналогу першої похідної
Визначення другої різниці
….функції ***m(x)***
Визначення дискретного
….аналогу другої похідної
Індексна символізація ***b***
Створення списку значень
….коефіцієнтів ***b*** та їх визначення
….за місцем у списку
Символізація виразу ***psr(x)***
Трансформація правої
….частини диференціального
….рівняння у вираз ***psr0***, що
….містить ліві кінцеві різниці
Розкриття дужок у виразі
….в кінцевих різницях ***psr0***
….та його приведення у
….форму невпорядкованого
….різницевого виразу ***psr1***
Перший крок до

```
Out[60]:
x0*(b0 + b1/T1 + b2/T1**2) - b1*x1/T1 –
2*b2*x1/T1**2 + b2*x2/T1**2

In [61]: psr3=psr2.collect(x1);psr3
Out[61]:
x0*(b0 + b1/T1 + b2/T1**2) + x1*(-b1/T1 –
2*b2/T1**2) + b2*x2/T1**2

In [62]: d = symbols('d:3')
In [63]: L33 = psr3.as_ordered_terms( );L33
Out[63]: [x0*(b0 + b1/T1 + b2/T1**2),
 x1*(-b1/T1 - 2*b2/T1**2), b2*x2/T1**2]
In [64]: d0 = L33[0]/x0;d0
Out[64]:
b0 + b1/T1 + b2/T1**2
In [65]: d1 = L33[1]/x1;d1
Out[65]:
-b1/T1 - 2*b2/T1**2
In [66]: d2 = L33[2]/x2;d2
Out[66]:
b2/T1**2
In [67]: X = symbols ('X:3');X
Out[67]: (X0, X1, X2)
In [68]: L333=list(X);L333
Out[68]: [X0, X1, X2]
In [69]: X0=L333[0];X1=L333[1];X2=L333[2]
In [70]: X1=z**(-1)*X0
In [71]: X2=z**(-2)*X0
In [72]: Q1 = Function ('Q1')(z)
In [73]: Q1 =d0*X0+d1*X1+d2*X2;Q1
Out[73]:
X0*(b0 + b1/T1 + b2/T1**2) + X0*(-b1/T1 –
 2*b2/T1**2)/z + X0*b2/(T1**2*z**2)
In [74]: F222=Function('F222')(z)
In [75]: F222=Q1.subs(X0,1);F222
Out[75]:
b0 + (-b1/T1 - 2*b2/T1**2)/z + b1/T1 +
b2/T1**2 + b2/(T1**2*z**2)
In [76]: b0=4;b1=2;b2=1
In [77]: T1=0.5
In [78]: d22=dict({"T1":0.5});d22
Out[78]: {'T1': 0.5}
In [79]: F2222=F222.subs([(L4[0],4),
(L4[1],2),(L4[2],1)]);F2222
Out[79]:
```

….впорядкування виразу ***psr1***
….з переведенням його у
….форму ***psr2*** шляхом
….збирання коефіцієнтів біля ***x0***
Другий крок до
….впорядкування виразу ***psr1***
….з переведенням його у
….форму ***psr3*** шляхом
….збирання коефіцієнтів біля ***x1***
Індексна символізація ***d***
Трансформація символьного
….виразу ***psr3*** в список ***L33*** його
….складових
Визначення коефіцієнта ***d0***
….шляхом ділення на ***x0***
….першого члена списку ***L33***
Визначення коефіцієнта ***d1***
….шляхом ділення на ***x1***
….другого члена списку ***L33***
Визначення коефіцієнта ***d2***
….шляхом ділення на ***x2***
….третього члена списку ***L33***
Індексна символізація
….змінної ***X***
Створення списку значень
….змінної ***X*** та їх визначення
….за місцем у списку
Визначення функції ***X1(z)***
Визначення функції ***X2(z)***
Символізація виразу ***Q1(z)***
Трансформація виразу
….різницевого ***psr3*** на площину
….комплексну у вигляді
….змінної ***z***
Символізація виразу ***F222(z)***
Визначення виразу ***F222(z)***
….шляхом підстановки у
….вираз ***Q1(z)*** замість ***X0***
….одиничного значення
Внесення значень ***b***
….та ***T1*** рівного ***T***
Створення словника ***d22***
….для ***T1***
Трансформація виразу
….***F222(z)*** у вираз ***F2222(z)***
….шляхом підстановки

```
4 + (-2/T1 - 2/T1**2)/z + 2/T1 + T1**(-2) +
 1/(T1**2*z**2)
In [80]: F22222=F2222.subs({"T1":0.5});F22222
Out[80]:
12.0 - 12.0/z + 4.0/z**2


In [81]: W = Function ('W')(z)
In [82]: W = F22222/F1111;W
Out[82]:
(12.0 - 12.0/z + 4.0/z**2)/(44.0 - 72.0/z
 + 40.0/z**2 - 8.0/z**3)
```

….числових значень
….коефіцієнтів ***b***
Трансформація виразу
….***F2222(z)*** у вираз ***F22222(z)***,
….який є чисельником
….дискретної передаточної
….функції об’єкта дослідження
Символізація виразу ***W(z)***
Обчислення дискретної
….передаточної функції
….функції об’єкта дослідження
….з конкретизацією параметрів

Кінець програми 28

Постскриптум до програми 28 та вступ до програми 29

В програмі 28 ми продемонстрували, як базове диференціальне рівняння, яке є математичною моделлю об’єкта дослідження в неперервній часовій області, трансформувати в рівняння в кінцевих різницях, які є дискретними аналогами похідних в дискретному часі, а рівняння в кінцевих різницях трансформувати в різницеве рівняння, яке зв’язує між собою лише значення решітчастих функцій, визначених при сусідніх значеннях дискретного часу, та як із різницевого рівняння, у свою чергу, з використанням Z-перетворення синтезувати дискретну передаточну функцію цього об’єкта, визначену на комплексній площині в її точках за межами круга одиничного радіусу, яка несе на комплексній площині про об’єкт дослідження стільки ж інформації, скільки її містило в неперервній часовій області базове диференціальне рівняння цього об’єкта дослідження.

В наступній же програмі 29 ми продемонструємо зворотний процес, тобто, покажемо, як, скориставшись уже відомою нам дискретною передаточною функцією об’єкта дослідження, синтезованою у **програмі 28**, та Z-перетворенням сигналу, що надходить на вхід цього об’єкта дослідження, скориставшись запропонованою Густавом Дечем формою оберненого Z-перетворення, віднайти у дискретному часі послідовність значень реакції даного об'єкта дослідження на заданий дискретний вхідний сигнал, вибраний нами з таблиці 9.1

Програма мовою Python для розв’язання задач, пов’язаних з визначенням послідовності значень $\{y[0],\ y[T],\ y[2T],\ y[3T],\ y[4T], \dots\}$ реакції $y[kT]$, $k = 0, 1, 2,\dots$, $T > 0$ об’єкта дослідження з конкретизованою у програмі 26 дискретною передаточною функцією

$$W(z) = \frac{2z^{-1} + 4z^{-2}}{44 - 72z^{-1} + 40z^{-2} - 8z^{-3}}$$

на дискретний вхідний сигнал $X[kT]$, $k = 0, 1, 2,\dots$, $T > 0$, що моделюється решітчастою функцією заданої структури з конкретизацією у дискретному часі у вигляді $x[kT] = kT$, $k = 0, 1, 2, \dots$, $T = 0{,}5$; Z-зображення для якої, згідно з таблицею 9.1, матиме вигляд

$$X(z) = T\frac{z}{(z-1)^2}$$

з конкретизацією на комплексній площині у вигляді

$$X(z) = \frac{0.5\, z^{-1}}{1 - 2z^{-1} + z^{-2}}$$

та з використанням рівняння $\boldsymbol{Y(z) = W(z)\, X(z)}$, в якому $\boldsymbol{Y(z) = Z\{y[kT]\}}$ – Z-зображення реакції $\boldsymbol{y[kT]}$.

(Програма 29)

```
In [1]: import sympy                                  # Внесення ППП sympy
In [2]: from sympy import*                            # Виклик усіх функцій sympy
In [3]: z = symbols('z')                              # Символізація змінної z
In [4]: T = symbols ('T')                             # Символізація періоду T
In [5]: X = Function ('X')(z)                         # Символізація функції X(z)
In [6]: Y = Function ('Y')(z)                         # Символізація функції Y(z)
In [7]: W = Function ('W')(z)                         # Символізація функції W(z)
In [8]: C = Function ('C')(z)                         # Символізація виразу C(z)
In [9]:D = Function ('D')(z)                          # Символізація виразу D(z)
In [10]: A = Function ('A:5')(z)                      # Символізація функції A(z)
                                                      ….з індексами
In [11]: y = symbols ('y:5')                          # Символізація функції y(z)
                                                      ….з індексами
In [12]: R = Function ('R')(z)                        # Символізація частки R(z)
In [13]: Q = Function ('Q')(z)                        # Символізація функції Q(z)
In [14]: W = (12.0 - 12.0/z + 4.0/z**2)/(44.0 –       # Конкретизація функції W(z)
 72.0/z+ 40.0/z**2 - 8.0/z**3)
In [15]: X = T*z**(-1)/(1-2*z**(-1)+z**(-2))          # Конкретизація  функції X(z)
In [16]: Y = W*X; Y                                   # Визначення функції Y(z)
Out[16]:
T*(12.0 - 12.0/z + 4.0/z**2)/(z*(1 - 2/z +
 z**(-2))*(44.0 - 72.0/z + 40.0/z**2 –
8.0/z**3))
In [17]: Q = Y.expand();Q                             # Розкриття дужок у виразі
Out[17]:                                              ….для функції Y(z) та приведення
4.0*T/(44.0*z**3 - 160.0*z**2 + 228.0*z –             ….його до вигляду Q(z)
160.0 + 56.0/z - 8.0/z**2) - 12.0*T/(44.0*z**2
 - 160.0*z + 228.0 - 160.0/z + 56.0/z**2 –
 8.0/z**3) + 12.0*T/(44.0*z - 160.0 + 228.0/z –
 160.0/z**2 + 56.0/z**3 - 8.0/z**4)
In [18]: T = 0.5                                      # Внесення значення періоду
In [19]: d = dict({"T":0.5});d                        ….дискретизації T та його
Out[19]: {'T': 0.5}                                   ….подання словником
In [20]: M = Function('M')(z)                         # Символізація виразу M(z)
In [21]: M = Q.subs({'T':0.5});M                      # Підставлення значення
Out[21]:                                              ….періоду T у трансформований
2.0/(44.0*z**3 - 160.0*z**2 + 228.0*z –               ….до вигляду Q(z) вираз
160.0 + 56.0/z - 8.0/z**2) - 6.0/(44.0*z**2 –         …. для функції Y(z) та її
160.0*z + 228.0 - 160.0/z + 56.0/z**2 –               ….приведення до вигляду M(z)
8.0/z**3) + 6.0/(44.0*z - 160.0 + 228.0/z –
 160.0/z**2 + 56.0/z**3 - 8.0/z**4)
```

```
In [22]: M1= Function('M1')(z)
In [23]: M1 = cancel(M);M1
Out[23]:
1.0*(11616.0*z**4 - 11616.0*z**3 +
3872.0*z**2)/(85184.0*z**5 –
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)
In [24]: expr = M1
In [25]: C,D = fraction(expr)
In [26]: print(C)
11616.0*z**4 - 11616.0*z**3 + 3872.0*z**2
In [27]: print(D)
85184.0*z**5 - 309760.0*z**4 + 441408.0*z**3
- 309760.0*z**2 + 108416.0*z - 15488.0
In [28]: L1 = list(y);L1
Out[28]: [y0, y1, y2, y3, y4]

In [29]: f = Function ('f:16')(z)

In [30]: A0 = L1[0];A0
Out[30]:
y0
In [31]: A1 =L1[1]*z**(-1);A1
Out[31]:
y1/z
In [32]: A2 = L1[2]*z**(-2);A2
Out[32]:
y2/z**2
In [33]: A3 = L1[3]*z**(-3);A3
Out[33]:
y3/z**3
In [34]: A4 =L1[4]*z**(-4);A4
Out[34]:
y4/z**4
In [35]: r = Function ('r')(z)
In [36]: R = A0+A1+A2+A3+A4+r;R
Out[36]:
y0 + y1/z + y2/z**2 + y3/z**3 + y4/z**4 + r(z)
In [37]: L3 = C.as_ordered_terms( );L3
Out[37]: [11616.0*z**4, -11616.0*z**3,
 3872.0*z**2]
In [38]: L4 = D.as_ordered_terms( );L4
Out[38]:
[85184.0*z**5,
 -309760.0*z**4,
 441408.0*z**3,
```

Символізація виразу ***M1(z)***
Приведення членів виразу
….***M(z)*** до спільного
….знаменника та його
….трансформація до вигляду
….***M1(z),*** який є відношенням
….многочленів ***C(z)*** та ***D(z)***
Надання виразу ***M1(z)*** статусу
….функції ***expr***, придатної для
….факторизації цього виразу у
….вигляді многочленів ***C(z), D(z),***
….перший з яких є діленим, а
….другий – дільником, та їх
….друкування
Створення списку ***L1*** для
….послідовності
….***y[0], y[T], y[2T], y[3T], y{4T]***
Символізація функції ***f(z)***
….з індексами
Визначення структури
….нульової складової
….частки ***R(z)***
Визначення структури
….першої складової
….частки ***R(z)***
Визначення структури
….другої складової
….частки ***R(z)***
Визначення структури
….третьої складової
….частки ***R(z)***
Визначення структури
….четвертої складової
….частки ***R(z)***
Символізація залишку ***r(z)***
Визначення структури
….частки ***R(z)***

Трансформація діленого ***C(z)***
….у список ***L3*** його структурних
….складових
Трансформація дільника ***D(z)***
….у список ***L4*** його структурних
….складових

```
 -309760.0*z**2,
 108416.0*z,
 -15488.0000000000]
In [39]: L3[0]                                  # Виклик початкового члена
Out[39]:                                        ….L3[0] зі списку L3
11616.0*z**4                                    ….складових діленого C(z)
In [40]: L4[0]                                  # Виклик початкового члена
Out[40]:                                        ….L4[0] зі списку L4
85184.0*z**5                                    ….складових дільника D(z)
In [41]: L3[0]/L4[0]                            # Ділення початкового члена
Out[41]:                                        ….діленого на початковий
0.136363636363636/z                             ….член дільника та
In [42]: A1 = L3[0]/L4[0];A1                    ….ідентифікація результату
Out[42]:                                        ….у вигляді структурної
0.136363636363636/z                             ….складової A1 частки R(z)
In [43]: y1 = A1*z;y1                           # Обчислення першої
Out[43]:                                        ….складової y1 списку L1
0.136363636363636
In [44]: A1*D                                   # Перемноження першої
Out[44]:                                        ….складової A1 частки R(z)
0.136363636363636*(85184.0*z**5 –               ….на дільник D(z)
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z
In [45]: f0 = C-A1*D;f0                         # Формування нового
Out[45]:                                        ….діленого f0 у вигляді
11616.0*z**4 - 11616.0*z**3 + 3872.0*z**2       ….різниці між діленим
 - 0.136363636363636*(85184.0*z**5 –            ….C(z) попереднього етапу
 309760.0*z**4 + 441408.0*z**3 –                ….ділення та результатом
309760.0*z**2 + 108416.0*z - 15488.0)/z         ….перемноження A1 на D(z)
In [46]: f1 = f0.expand ();f1                   # Спрощення діленого f0
Out[46]:                                        ….та приведення його
-1.81898940354586e-12*z**4 + 30624.0*z**3       ….до форми f1, придатної
- 56320.0*z**2 + 42240.0*z - 14784.0 + \        ….для здійснення
2112.0/z                                        ….наступного етапу ділення
In [47]: L6 = f1.as_ordered_terms( );L6         # Трансформація діленого f1(z)
Out[47]:                                        ….у список L6 його структурних
[-1.81898940354586e-12*z**4,                    ….складових
 30624.0*z**3,
 -56320.0*z**2,
 42240.0*z,
 -14784.0000000000,
 2112.0/z]
In [48]: L6.pop(0)                              # Вилучення зі списку L6
Out[48]:                                        ….початкового члена, що має
-1.81898940354586e-12*z**4                      ….коефіцієнт на рівні
```

```
In [49]: L6                                          ….технічного нуля та друк
Out[49]: [30624.0*z**3, -56320.0*z**2,               ….цього списку у тій же формі,
 42240.0*z, -14784.0000000000, 2112.0/z]             ….але без вилученого члена
In [50]: f4 = L6[0]+L6[1]+L6[2]+L6[3]+L6[4];f4       # Формування нового діленого
Out[50]:                                             ….f4 для третього етапу
30624.0*z**3 - 56320.0*z**2 + 42240.0*z              ….ділення
- 14784.0 + 2112.0/z
In [51]: A2=L6[0]/L4[0];A2                           # Ділення нового початкового
Out[50]:                                             ….члена L6[0] нового діленого
0.359504132231405/z**2                               ….f4 на початковий член L4[0]
In [51]: y2=A2*z**2;y2                               ….дільника D та ідентифікація
Out[51]:                                             ….результату у вигляді
0.359504132231405                                    ….складової A2 частки R(z)
                                                     ….і обчислення другої
                                                     …. складової y2 списку L1
In [52]: A2*D                                        # Перемноження другої
Out[52]:                                             ….складової A2 частки R(z)
0.359504132231405*(85184.0*z**5 –                    ….на дільник D(z)
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z**2
In [53]: f5=f4-A2*D;f5                               # Формування нового
Out[53]:                                             ….діленого f5 у вигляді
30624.0*z**3 - 56320.0*z**2 + 42240.0*z –            ….різниці між діленим
 14784.0 + 2112.0/z - 0.359504132231405*\            ….f4 попереднього етапу
(85184.0*z**5 - 309760.0*z**4 +                      ….ділення та результатом
441408.0*z**3 - 309760.0*z**2 +                      ….перемноження A2 на D(z)
108416.0*z - 15488.0)/z**2
In [54]: f6=f5.expand();f6                           # Спрощення діленого f5
Out[54]:                                             ….та приведення його
-3.63797880709171e-12*z**3 + 55040.0*z**2            ….до форми f6, придатної
 - 116448.0*z + 96576.0 - 36864.0/z +                ….для здійснення
 5568.0/z**2)                                        ….наступного етапу ділення
In [55]: L7 = f6.as_ordered_terms( );L7               # Трансформація діленого f6(z)
Out[55]:                                             ….у список L7 його структурних
[-3.63797880709171e-12*z**3,                         ….складових
 55040.0*z**2,
 -116448.0*z,
 96576.0000000000,
 -36864.0/z,
 5568.0/z**2]
In [56]: L7.pop(0)                                   # Вилучення зі списку L7
Out[56]:                                             ….початкового члена, що має
-3.63797880709171e-12*z**3                           ….коефіцієнт на рівні
In [57]: L7                                          ….технічного нуля та друк
```

```
Out[57]: [55040.0*z**2, -116448.0*z,
96576.0000000000, -36864.0/z, 5568.0/z**2]
In [58]: f7 =L7[0]+L7[1]+L7[2]+L7[3]+L7[4];f7
Out[58]:
55040.0*z**2 - 116448.0*z + 96576.0 –
36864.0/z + 5568.0/z**2
In [59]: A3 =L7[0]/L4[0];A3
Out[59]:
0.646130728775357/z**3
In [60]: y3=A3*z**3;y3
Out[60]:
0.646130728775357



In [61]: A3*D
Out[61]:
0.646130728775357*(85184.0*z**5 –
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z**3
In [62]: f8 = f7-A3*D;f8
Out[62]:
55040.0*z**2 - 116448.0*z + 96576.0 – \
36864.0/z + 5568.0/z**2 –
0.646130728775357*(85184.0*z**5 –
 309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z**3
In [63]: f9 = f8.expand();f9
Out[63]:
83697.4545454546*z - 188631.272727273
+ 163281.454545455/z –
64482.9090909091/z**2 +
10007.2727272727/z**3
In [64]: L8 = f9.as_ordered_terms( );L8
Out[64]:
[83697.4545454546*z,
 -188631.272727273,
 163281.454545455/z,
 -64482.9090909091/z**2,
 10007.2727272727/z**3]
In [65]: A4 =L8[0]/L4[0];A4
Out[65]:
0.982549006215423/z**4
In [66]: y4 = A4*z**4;y4
Out[66]:
0.982549006215423
```

….цього списку у тій же формі,
….але без вилученого члена
Формування нового діленого
….***f7*** для четвертого етапу
….ділення

Ділення нового початкового
….члена ***L7[0]*** нового діленого
….***f7*** на початковий член ***L4[0]***
….дільника ***D*** та ідентифікація
….результату у вигляді
….складової ***A3*** частки ***R(z)***
….і обчислення третьої
…. складової ***y3*** списку ***L1***
Перемноження третьої
….складової ***A3*** частки ***R(z)***
….на дільник ***D(z)***

Формування нового
….діленого ***f8*** у вигляді
….різниці між діленим
….***f7*** попереднього етапу
….ділення та результатом
….перемноження ***A3*** на ***D(z)***

Спрощення діленого ***f8***
….та приведення його
….до форми ***f9,*** придатної
….для здійснення
….наступного етапу ділення

Трансформація діленого ***f9***
….у список ***L8*** його структурних
….складових

Ділення нового початкового
….члена ***L8[0]*** нового діленого
….***f9*** на початковий член ***L4[0]***
….дільника ***D*** та ідентифікація
….результату у вигляді
….складової ***A4*** частки ***R(z)***

```
                                                    ….і обчислення четвертої
                                                    ….складової y4 списку L1
In [67]: A4*D                                       # Перемноження четвертої
Out[67]:                                            ….складової A4 частки R(z)
0.982549006215423*(85184.0*z**5 –                   ….на дільник D(z)
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z**4
In [68]: f10=f9-A4*D;f10                            # Формування нового
Out[68]:                                            ….діленого f10 у вигляді
83697.4545454546*z - 188631.272727273               ….різниці між діленим
+ 163281.454545455/z –                              ….f9 попереднього етапу
64482.9090909091/z**2 + 10007.2727272727/           ….ділення та результатом
z**3 - 0.982549006215423*(85184.0*z**5 –            ….перемноження A4 на D(z)
309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)/z**4
In [69]: f11=f10.expand();f11                       # Спрощення діленого f10
Out[69]:                                            ….та приведення його
115723.107438017 - 270423.537190083/z               ….до форми f11, придатної
 + 239871.47107438/z**2 - 96516.7603305786/         ….для здійснення
z**3 + 15217.7190082645/z**4                        ….наступного етапу ділення
In [70]: r=f11/D;r                                  # Визначення залишку від
Out[70]:                                            ….ділення
(115723.107438017 - 270423.537190083/z
+ 239871.47107438/z**2 - 96516.7603305786/
z**3 + 15217.7190082645/z**4)/(85184.0*z**5
 - 309760.0*z**4 + 441408.0*z**3 –
309760.0*z**2 + 108416.0*z - 15488.0)
In [71]: L1[0]=0;L1                                 # Внесення значення y0
Out[71]: [0, y1, y2, y3, y4]                        ….в список L1
In [72]: L1[1]=y1;L1                                # Внесення значення y1
Out[72]: [0, 0.136363636363636, y2, y3, y4]         ….в список L1
In [73]: L1[2]=y2;L1                                # Внесення значення y2
Out[73]: [0, 0.136363636363636,                     ….в список L1
0.359504132231405, y3, y4]
In [74]: L1[3]=y3;L1                                # Внесення значення y3
Out[74]: [0, 0.136363636363636,                     ….в список L1
0.359504132231405, 0.646130728775357, y4]
In [75]: L1[4]=y4;L1                                # Внесення значення y4
Out[75]: [0, 0.136363636363636,                     ….в список L1
0.359504132231405, 0.646130728775357,
0.982549006215423]
In [76]: import matplotlib                          # Виклик ППП matplotlib
In [77]: import matplotlib.pyplot as plt            # Виклик модуля pyplot
In [78]: import numpy as np                         # Виклик ППП numpy
In [79]: plt.plot(L1)                               # Друк графіка y[k]
Out[79]: [<matplotlib.lines.Line2D at 0x22cea175100>]
```

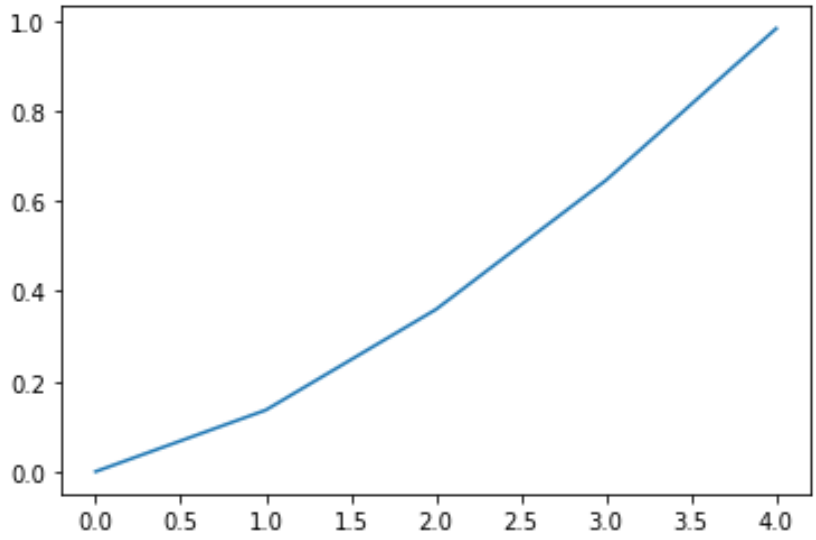


Рисунок 9.1 – Графік реакції об’єкта дослідження ***y[k], k = 0, 1, 2, 3, 4; T = 0,5*** у відносному часі ***k***

Кінець програми 29

9.4 Характеристики дискретних операторів, за допомогою яких синтезуються регресійні моделі часових рядів

У кінці восьмого підрозділу у базовому навчальному посібнику [2] серед інших стосовно дослідження операторів сформульовані і такі питання:

1. Що собою являє оператор лінійного фільтра?
2. Що собою являє білий шум? Які його основні властивості?
3. Що таке регресія і авторегресія? Яку форму мають їх оператори?
4. Синтезуйте модель часового ряду у формі авторегресії.
5. Як пов’язані між собою оператор лінійного фільтра та оператор авторегресії для часового ряду?
6. Що собою являють ковзне середнє часового ряду і оператор ковзного середнього?
7. Дайте означення автоковаріації і автокореляції часового ряду. Як знайти їх числові оцінки? Які основні їх властивості ви знаєте?
8. Для чого потрібні і як виводяться рівняння Юла – Уокера?
9. Як розв’язати рівняння Юла – Уокера?
10. За реалізацією часового ряду синтезуйте його модель у формі авторегресії з ідентифікацією цієї моделі.
11. Побудуйте модель часового ряду у формі авторегресії – ковзного середнього.
12. Яким чином можна нестаціонарний часовий ряд трансформувати у стаціонарний?
13. Побудуйте модель нестаціонарного часового ряду у формі авторегресії – проінтегрованого ковзного середнього.

Тож із відповідей на ці питання ми і продовжимо викладення змісту дев'ятого розділу другої частини нашого «Навчального посібника для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python».

Що стосується суті відповідей на кожне з цих питань, то її ми розкриватимемо послідовно, використовуючи матеріал, викладений в роботі [2]. Отже, як і при викладенні матеріалу попереднього підрозділу, при викладенні матеріалу і цього підрозділу почнемо наше викладення з розлогої цитати, взятої з нашого базового навчального посібника [2], в якій змінена не лише нумерація формул, але і опущено багато проміжних викладок, що робить цей матеріал не стільки цитатою, скільки її вільним переказом, через що ми текст цієї цитати не беремо в лапки.

Під час системного аналізу динамічних процесів, що мають випадковий характер, та створенні інформаційних технологій, придатних для реалізації цього аналізу, суттєва роль відводиться прогнозуванню розвитку цих процесів у часі.

На сьогоднішній день найбільш ефективними математичними моделями, за допомогою яких можна прогнозувати розвиток процесів, є ті, під час побудови яких використовуються часові ряди.

Нагадаємо, що **часовий ряд – це сукупність значень випадкового процесу, взятих через однакові проміжки часу *t*. Позначимо цю сукупність символом** z_t. Отже z_t – це є дискретний у часі випадковий процес.

Задача прогнозу полягає в тому, щоб, знаючи значення процесу у момент *t,* спрогнозувати його значення в момент $t + l$, де l – час упередження. Щоб відрізняти прогнозне значення процесу від дійсного, дійсне значення часового ряду у момент $t + l$ позначають символом z_{t+l}, а прогнозне – $z_t\,(l)$.

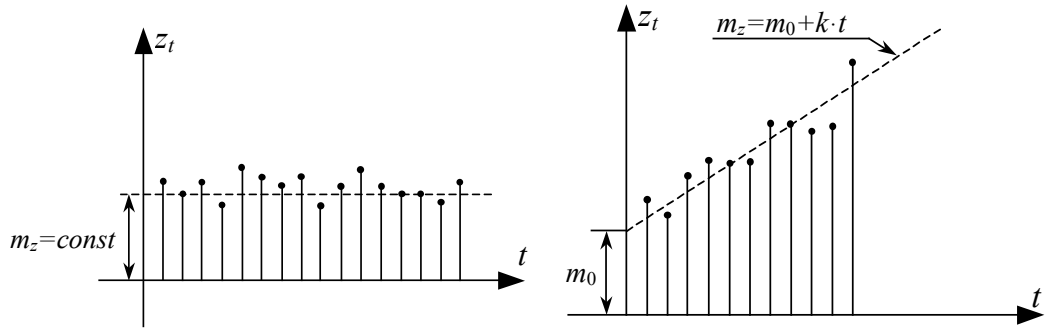
Зрозуміло, що точно спрогнозувати значення випадкового процесу, яким є часовий ряд, неможливо в принципі, а тому прогноз здійснюють, домагаючись мінімуму якогось функціонала, вибраного за критерій адекватності моделі.

Якщо значення t є малим (1, 2 кроки), то одним із таких критеріїв можна взяти дисперсію відхилення $z_t\,(l)$ від z_{t+l}, яка має бути для оптимальної моделі прогнозу мінімальною, тобто,

$$E\left\{\left(z_{t+l} - z_t\left(l\right)\right)^2\right\} \;\rightarrow\; \min\,, \tag{9.41}$$

де E – символ операції математичного очікування.

Як і будь-який інший випадковий процес, часовий ряд z_t може бути стаціонарним (рис. 9.2, а) або нестаціонарним (рис. 9.2, б).



а) б)

Рисунок 9.2 – Графіки реалізації стаціонарного (а) та нестаціонарного (б) часових рядів

Для стаціонарного часового ряду характерною є рівновага його значень z_t біля середнього значення m_z, яке є константою, як показано на рис. 9.2, а).

Для нестаціонарного часового ряду ковзне середнє значення $m_z(t)$ процесу є функцією часу t, як показано на рис. 9.2, б).

Введемо низку корисних операторів, які будуть потрібними у подальшому.

1. Оператор B зсуву назад на одну одиницю часу

$$z_{t-1} = Bz_t. \tag{9.42}$$

Зрозуміло, що, згідно з виразом (9.42), справедливим є і вирази:

$$z_{t-2} = B\, z_{t-1} = B^2\, z_t, \tag{9.43}$$

$$z_{t-m} = B^m z_t. \tag{9.44}$$

2. Різницевий оператор ∇ із зсувом назад на одну одиницю часу

$$\nabla\, z_t = z_t - z_{t-1} = z_t - B\, z_t = (1-B)z_t,. \tag{9.45}$$

$$\nabla = 1 - B. \tag{9.46}$$

3. Оператор суми S

$$\begin{aligned} S\, z_t &= z_t + z_{t-1} + z_{t-2} + \cdots = z_t + B\, z_t + B^2 z_t + \cdots = \\ &= (1 + B + B^2 + \cdots) z_t = \frac{1}{1-B} z_t = (1-B)^{-1} z_t = \nabla^{-1} z_{t,,}, \end{aligned} \tag{9.47}$$

$$S = \nabla^{-1}. \tag{9.48}$$

Отже, оператор суми є оберненим різницевому оператору зі зсувом назад.

Звертаємо увагу на те, що під час виведення співвідношення (9.47) було використано формулу суми членів нескінченно спадної геометричної прогресії зі знаменником B, який, за умови розгляду його як числа та умови збіжності ряду, має бути меншим одиниці, як це і засвідчує вираз (9.46).

Далі нагадаємо, що **послідовність некорельованих і розподілених нормально випадкових імпульсів a_t з нульовим середнім значенням і дисперсією**

$$\sigma_a^2 = const \tag{9.49}$$

називають дискретним білим шумом.

Спробуємо використати імпульси білого шуму a_t для побудови моделі часового ряду z_t у такий спосіб

$$z_t = \mu + a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \ \dots\ , \tag{9.50}$$

де μ – рівень відліку (середнє значення) часового ряду z_t, а ψ_k, $k = 1, 2, \dots$ – коефіцієнти ваги імпульсів білого шуму, з якими вони входять до суми (9.50).

Здійснимо операцію центрування часового ряду z_t, віднявши від z_t середнє значення μ. Для центрованого часового ряду

$$\tilde{z}_t = z_t - \mu. \tag{9.51}$$

З врахуванням (9.51) та (9.44) із виразу (9.50) отримаємо

$$\tilde{z_t} = a_t + \psi_1 a_{t-1} + \psi_2 a_{t-2} + \cdots = a_t + \psi_1 B a_t + \psi_2 B^2 a_t + \cdots = \\ = (1 + \psi_1 B + \psi_2 B^2 + \cdots) a_t = \Psi(B) a_t, \tag{9.52}$$

де

$$\Psi(B) = 1 + \psi_1 B + \psi_2 B^2 + \ \ldots \ . \tag{9.53}$$

Вираз $\Psi(B)$ є ***оператором фільтра***, який перетворює послідовність імпульсів білого шуму a_t у часовий ряд з заданими властивостями (рис. 9.3), тобто, ***ставить у відповідність дискретній стохастичній функції з однієї нульвимірної множини іншу дискретну стохастичну функцію з іншої нульвимірної множини.***

Коефіцієнти оператора фільтра ψ_k, $k = 1, 2, \ldots$ підбираються у процедурі мінімізації критерію (9.41) при $l = 0$.

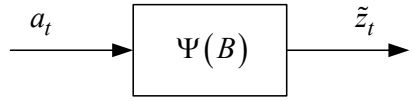


Рисунок 9.3 – Структурна схема лінійного фільтра

В моделі лінійного фільтра (9.52) значення часового ряду z_t визначаються через зважену суму поточного та попередніх імпульсів білого шуму a_t.

Характерною особливістю оператора $\Psi(B)$ фільтра, заданого виразом (9.53), є те, що він теоретично має нескінченну кількість членів, що створює певні незручності у разі його практичного використання.

Тому цікавою виявилась пропозиція будувати модель часового ряду z_t на основі скінченної множини потужністю q зважених імпульсів білого шуму a_t у вигляді

$$\tilde{z_t} = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \cdots - \theta_q a_{t-q} = a_t - \theta_1 B a_t - \theta_2 B^2 a_t - \cdots - \\ - \theta_q a_{t-q} = \Theta(B) a_t, \tag{9.54}$$

де $\Theta(B)$ – оператор ковзного середнього

$$\Theta(B) = 1 - \theta_1 B - \theta_2 B^2 - \ \ldots - \theta_q B^q \tag{9.55}$$

порядку q, який фактично теж є оператором фільтра, але з обмеженою кількістю складових, тобто «укороченим» оператором фільтра.

Співвідношення (9.54) задає модель стаціонарного часового ряду z_t, у якій використовується оператор ковзного середнього (9.55), а тому математики домовились цю модель називати моделлю ковзного середнього порядку q (скорочено: модель КС(q)).

Покажемо, як оператор фільтра пов'язаний з оператором авторегресії.

З філософської точки зору модель регресії – це модель «з оглядкою назад, у бік, звідки прийшов»; тобто, це модель, яка встановлює значення якоїсь координати процесу у даний момент часу за її незалежними складовими, визначеними у попередній момент. Кількість врахованих складових визначає порядок регресії.

Виходячи з даного трактування, ***модель авторегресії – це модель, яка встановлює значення якоїсь координати процесу у даний момент часу на основі своїх попередніх значень. Кількість врахованих попередніх значень визначає порядок авторегресії.***

Для центрованого часового ряду $\tilde{z}_t$ модель авторегресії порядку p (скорочено: АР(p)) можна записати у вигляді

$$\tilde{z}_t = \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ \ldots + \phi_p \tilde{z}_{t-p} + a_t , \tag{9.56}$$

де a_t – імпульс білого шуму, означення якого наведено вище.

Переносячи усі члени рівняння (9.56), окрім імпульсу білого шуму, в ліву частину цього рівняння та враховуючи співвідношення (9.44), матимемо

$$\Phi(B)\tilde{z}_t = a_t , \tag{9.57}$$

де $\Phi(B)$ – ***оператор авторегресії порядку*** p який ***має вигляд***

$$\Phi(B) = 1 - \phi_1 B - \phi_2 B^2 - \ldots - \phi_p B^p \ . \tag{9.58}$$

Помноживши зліва обидві частини рівняння (9.57) на оператор $\Phi^{-1}(B)$, обернений до оператора $\Phi(B)$, отримаємо

$$\tilde{z}_t = \Phi^{-1}(B) a_t . \tag{9.59}$$

Порівнюючи вирази (9.59) та (9.52) бачимо, що

$$\Psi(B) = \Phi^{-1}(B) . \tag{9.60}$$

Отже, синтезувавши за реалізацією досліджуваного часового ряду z_t оператор авторегресії $\Phi(B)$, що зробити, як буде показано далі, нескладно, та визначивши оператор $\Phi^{-1}(B)$, обернений до $\Phi(B)$, що теж досить просто, тим самим одночасно визначаємо і оператор $\Psi(B)$ лінійного фільтра, який з білого шуму a_t формує часовий ряд z_t з заданими властивостями.

Під час розв'язання задачі ідентифікації моделі часового ряду z_t на основі оператора авторегресії порядку p потрібно визначити $p+2$ невідомих, якими є коефіцієнти $\phi_1, \phi_2, \ldots, \phi_p$ оператора $\Phi(B)$, середнє значення μ процесу z_t та дисперсію σ_a^2 білого шуму a_t.

Про те, як розв'язати цю задачу, мова піде після того, як ми ***визначимось з поняттями автоковаріації та автокореляції часового ряду.***

Автоковаріацією γ_k ***часового*** **ряду** z_t **з затримкою** k **називають вира**з

$$\gamma_k = \operatorname{cov}\{z_t, z_{t+k}\} = E\{(z_t - \mu)\cdot(z_{t+k} - \mu)\}, \tag{9.61}$$

в якому E – символ обчислення математичного очікування від виразу, що стоїть у фігурних дужках.

Із (9.61) витікає, що

$$\gamma_0 = E\left\{(z_t - \mu)^2\right\} = \sigma_z^2 \tag{9.62}$$

– дисперсія часового ряду z_t.

Для отримання статистичної оцінки γ_k^* автоковаріації γ_k, визначеної виразом (9.61), використовують вираз

$$\gamma_k^* = \frac{1}{N-k}\sum_{t=1}^{N-k}\left(z_t-\mu\right)\cdot\left(z_{t+k}-\mu\right). \tag{9.63}$$

Автоковаріація γ_k ***характеризує ступінь лінійного зв'язку між значеннями часового ряду*** z_t ***та*** z_{t+k}**,** і чисельно цей ступінь не залежить від того зсуваємо ми двочлен, що стоїть в перших дужках виразу (9.63), відносно двочлена, що стоїть в других дужках виразу (9.63), вліво чи вправо, а тому

$$\begin{cases}|\gamma_k| \le \gamma_0,\\ \gamma_k = \gamma_{-k}.\end{cases} \tag{9.64}$$

Автокореляцією ρ_k **часового ряду** z_t **із затримкою** k **називають вираз**

$$\rho_k = \frac{\gamma_k}{\gamma_0} = \frac{E\left\{\left(z_t-\mu\right)\cdot\left(z_{t+k}-\mu\right)\right\}}{E\left\{\left(z_t-\mu\right)^2\right\}}. \tag{9.65}$$

А як випливає з виразів (9.62) та (9.64), для автокореляції ρ_k справедливими є співвідношення:

$$\begin{cases}|\rho_k| \le \rho_0,\\ \rho_0 = 1,\\ \rho_k = \rho_{-k}.\end{cases} \tag{9.66}$$

Вся сукупність значень γ_k ***задає автоковаріаційну функцію часового ряду*** z_t***, яка належить до класу решітчастих функцій. Аналогічно, сукупність всіх значень*** ρ_k ***задає автокореляційну функцію часового ряду*** z_t**.**

Приклад графіка автокореляційної функції ρ_k наведено на рис. 9.4.

А тепер повернемось до сформульованої уже вище задачі визначення коефіцієнтів оператора авторегресії, тобто отримаємо відповідь на питання: ***«Як визначити коефіцієнти авторегресії моделі часового ряду у формі*** $AP(p)$***?».***

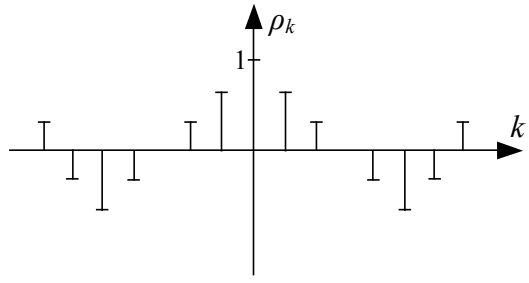


Рисунок 9.4 – Один із можливих графіків автокореляційної функції ρ_k, $k = \overline{-N, N}$

Покажемо, що відповідь на це питання можна отримати, розв'язавши рівняння Юла – Уокера, для синтезу яких спочатку помножимо вираз (9.56) на $\tilde{z}_{t-k}$ та отримаємо вираз

$$\tilde{z}_{t-k}\tilde{z}_t = \phi_1\tilde{z}_{t-k}\tilde{z}_{t-1} + \phi_2\tilde{z}_{t-k}\tilde{z}_{t-2} + \ \ldots + \phi_p\tilde{z}_{t-k}\tilde{z}_{t-p} + \tilde{z}_{t-k}a_t, \tag{9.67}$$

в якому зробимо заміну дискретної змінної, поклавши $t-k=\lambda$, а потім визначимо математичне очікування E від обох частин цього рівняння, в результаті чого матимемо систему рівнянь

$$\gamma_k = \phi_1\gamma_{k-1} + \phi_2\gamma_{k-2} + \ \ldots + \phi_p\gamma_{k-p} \tag{9.68}$$

для всіх k від 1 до p.

Але при $k=0$, з урахуванням виразу (9.64), отримаємо ще одне рівняння

$$\gamma_0 = \phi_1\gamma_1 + \phi_2\gamma_2 + \ \ldots + \phi_p\gamma_p + \sigma_a^2. \tag{9.69}$$

Відсутність результатів обчислення математичного очікування $E\{\tilde{z}_\lambda \cdot a_{\lambda+k}\}$ у рівняннях (9.68) та їх присутність у рівнянні (9.69) у вигляді дисперсії білого шуму σ_a^2 пояснюється тим, що, згідно з властивостями білого шуму, кожний його імпульс корелюється лише сам із собою і зовсім не корелюється ні з яким іншим, навіть розміщеним у часі поряд, тому

$$E\{\tilde{z}_\lambda \cdot a_{\lambda+k}\} = \begin{cases} 0, & при \quad k \neq 0 \\ \sigma_a^2, & при \quad k = 0 \end{cases}. \tag{9.70}$$

З рівняння (9.69) з урахуванням виразу (9.62) матимемо

$$\sigma_a^2 = \sigma_z^2 - \phi_1\gamma_1 - \phi_2\gamma_2 - \ \ldots - \phi_p\gamma_p. \tag{9.71}$$

Тож, якщо для реалізації стаціонарного часового ряду z_t довжиною N вже обчислені оціночні середнє значення μ і дисперсія σ_z^2 за відомими виразами:

$$\mu = \frac{1}{N}\sum_{t=1}^{N} z_t, \tag{9.72}$$

$$\sigma_z^2 = \frac{1}{N-1}\sum_{t=1}^{N}(z_t-\mu)^2, \tag{9.73}$$

знайдені якимсь чином коефіцієнти $\phi_1, \phi_2, \ldots, \phi_p$ і обчислені за виразом (9.63) автоковаріації $\gamma_1, \gamma_2, \ldots, \gamma_p$, тоді знайти дисперсію білого шуму σ_a^2 за виразом (9.71) зовсім нескладно.

Перш ніж навести алгоритм отримання числових значень коефіцієнтів $\phi_1, \phi_2, \ldots, \phi_p$, розділимо всі рівняння системи (9.68) на γ_0, в результаті чого отримаємо систему рівнянь вже не відносно автоковаріацій γ_k, $k=\overline{1,p}$, а відносно автокореляцій ρ_k, $k=\overline{1,p}$, тобто систему

$$\begin{cases} \phi_1\rho_0 + \phi_2\rho_1 + \phi_3\rho_2 + \ldots + \phi_p\rho_{p-1} = \rho_1, \\ \phi_1\rho_1 + \phi_2\rho_0 + \phi_3\rho_1 + \ldots + \phi_p\rho_{p-2} = \rho_2, \\ \phi_1\rho_2 + \phi_2\rho_1 + \phi_3\rho_0 + \ldots + \phi_p\rho_{p-3} = \rho_3, \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ \phi_1\rho_{p-1} + \phi_2\rho_{p-2} + \phi_3\rho_{p-3} + \ldots + \phi_p\rho_0 = \rho_p, \end{cases} \tag{9.74}$$

або, у матричній формі,

$$M \cdot \Phi = \rho, \tag{9.75}$$

де матриці мають розмірність $M(p \times p)$, $\Phi(p \times 1)$, $\rho(p \times 1)$.

У рівняннях (9.74).(9.75) невідомими є значення коефіцієнтів $\phi_1, \phi_2, \ldots, \phi_p$.

Для розв'язання системи (9.75) спочатку визначаємо матрицю M^{-1}, яка є оберненою до матриці *M*. Потім множимо матричне рівняння (9.75) зліва на M^{-1}. У результаті цього отримуємо матричне рівняння

$$\Phi = M^{-1}\rho. \tag{9.76}$$

Рівняння (9.74), доповнені рівнянням (9.71), носять назву рівнянь Юла – Уокера. Їх розв'язок у вигляді (9.76) дозволяє за попередньо обрахованими автокореляціями ρ_k, $k = \overline{1, p}$ ***часового ряду*** z_t ***визначити вектор*** $\Phi(p \times 1)$ ***коефіцієнтів*** $\phi_1, \phi_2, \ldots, \phi_p$ ***оператора авторегресії у моделі АР***(p)**.**

В моделі часового ряду z_t на основі авторегресії порядку *p* у формуванні поточного значення ряду бере участь лише один поточний імпульс білого шуму a_t. Природно припустити, що якщо від цього імпульсу a_t відняти зважену суму *q* попередніх значень білого шуму, то отримаємо модель, яка буде враховувати більш «тонкі» моменти випадкового процесу і більш адекватно віддзеркалюватиме його властивості, оскільки крім авторегресії ця модель буде враховувати ще й ковзне середнє процесу.

Така модель часового ряду z_t носить назву моделі авторегресії – ковзного середнього порядку (p, q) (скорочено: модель АРКС(p, q)) і має вигляд

$$\begin{aligned} \tilde{z}_t &= \phi_1 \tilde{z}_{t-1} + \phi_2 \tilde{z}_{t-2} + \ \ldots + \phi_p \tilde{z}_{t-p} + \\ &+ a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ \ldots - \theta_q a_{t-q}. \end{aligned} \tag{9.77}$$

Перенісши всі члени з $\tilde{z}_{t-i}$, $i = \overline{1, p}$ у ліву частину рівняння (9.77) та виконавши вже відомі перетворення, отримаємо рівняння

$$\Phi(B)\tilde{z}_t = \Theta(B)a_t, \tag{9.78}$$

в якому оператори $\Phi(B)$ та $\Theta(B)$ визначаються виразами (9.58), (9.55).

Рівняння (9.78) і є основною формою моделі часового ряду z_t на основі АРКС(p, q).

Для ідентифікації цієї моделі потрібно визначити $p + q + 2$ невідомих, якими є коефіцієнти ϕ_i, $i = \overline{1, p}$ оператора $\Phi(B)$, коефіцієнти θ_j, $j = \overline{1, q}$ оператора $\Theta(B)$, середнє значення μ процесу z_t та дисперсія σ_a^2 білого шуму a_t.

А далі сконцентруємо увагу на наших роботах [8], [11], в яких класична теорія синтезу та ідентифікації математичних моделей дискретних лінійних стохастичних динамічних об'єктів доповнена нами ще одним методом, який узагальнює класичну методику Юла – Уокера. Нагадаємо, що викладення цього матеріалу в оглядовому підрозділі роботи [8] починається з того, що ідентифікувати модель дискретного лінійного стохастичного динамічного об'єкта у вигляді авторегресії порядку p, тобто, модель АР(p), яка має вигляд (9.77), де z_t – центроване значення часового ряду в момент часу *t,* a_t – імпульс білого шуму з дисперсією σ_a^2, згенерований в той же момент часу *t*, можна, застосовуючи відому методику Юла – Уокера [9.10], згідно з якою числові

значення коефіцієнтів регресії ϕ_i, i = 1, 2, …, p знаходяться з матричного рівняння (9.76), в якому матриця M^{-1}є оберненою до матриці M, а матриці ϕ, M, ρ, як витікає з системи (9.74), мають вигляд :

$$\phi = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \dots \\ \phi_p \end{bmatrix}, \quad M = \begin{bmatrix} 1. \rho_1. \rho_2. \rho_3 \dots \rho_{p-1} \\ \rho_1. 1. \rho_1. \rho_2 \dots \rho_{p-2} \\ \rho_2. \rho_1. 1. \rho_1 \dots \rho_{p-3} \\ \dots\dots\dots\dots\dots\dots \\ \rho_{p-1}\rho_{p-2}\rho_{p-3} \dots 1 \end{bmatrix}, \quad \rho = \begin{bmatrix} \rho_1 \\ \rho_2 \\ \rho_3 \\ \dots \\ \rho_p \end{bmatrix}, \tag{9.79}$$

де ρ_i, i = *1, 2, ..., p* – це автокореляції центрованого часового ряду z_t, які розраховуються за виразом (9.65), в якому $\gamma_i, i = 0,1,2,\dots,p$ – автоковаріації, що розраховуються для центрованого часового ряду z_tза виразом (9.63) та використовуються, окрім розв'язання матричного рівняння (9.75), і для обчислення дисперсії σ_a^2 білого шуму a_t за виразом (9.71), в який також підставляються результати (9.76) розв'язання матричного рівняння (9.75).

Ще раз підкреслюємо, що в наших роботах [8], [11] усі вирази, за допомогою яких розв'язуються рівняння Юла – Уокера, задають лише вихідні передумови для задачі синтезу удосконаленого нами методу ідентифікації моделі авторегресії-ковзного середнього АРКС(p,q) з довільними значеннями порядків p, q, структура якої для центрованого часового ряду z_tмає вигляд (9.77).

Одразу ж відзначимо, що в наших роботах [8], [11] при виведенні розрахункових співвідношень удосконаленого нами методу ідентифікації моделі авторегресії-ковзного середнього АРКС(p,q), як і в алгоритмі Юла – Уокера, теж використана умова відсутності кореляції між сусідніми імпульсами білого шуму, що має вигляд (9.70).

На першому етапі синтезу нашого удосконаленого методу ідентифікації моделі АРКС(p,q) та його алгоритму ми в роботі [8] конкретизували значення параметрів на такому рівні, на якому уже можна буде робити узагальнення, але які ще не приводять до занадто громіздких математичних виразів. Тобто, якби ми задали p = 1, q = 1, або p = 2, q = 2, то отриманої інформації для узагальнення було б недостатньо, а тому ми задали p = 3 і q = 3. Для цього значення параметрів ми в роботі [8] вираз (9.77) конкретизували у вигляді моделі АРКС(3,3), для якої справедливим є рівняння

$$z_t = \phi_1 z_{t-1} + \phi_2 z_{t-2} + \phi_3 z_{t-3} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \theta_3 a_{t-3}. \tag{9.80}$$

Домножуючи обидві частини рівняння (9.80) по черзі на z_t, z_{t-1}, z_{t-2}, z_{t-3}, $z_{t-4}, z_{t-5}, z_{t-6}$, усереднюючи отримані добутки за виразом для автоковаріації (9.63) і враховуючи умови (9.70), ми в роботі [8] отримали систему рівнянь, яка має вигляд:

$$\begin{cases} \{q_0 = g_1 q_1 + g_2 q_2 + g_3 q_3 + \sigma_a^2 - p_1(g_1 - p_1)\sigma_a^2 - p_2(g_2 - p_2)\sigma_a^2 - p_3(g_3 - p_3)\sigma_a^2\}, \\ \{q_1 = g_1 q_0 + g_2 q_1 + g_3 q_2 - p_1\sigma_a^2 - p_2(g_1 - p_1)\sigma_a^2 - p_3(g_2 - p_2)\sigma_a^2\}, \\ \{q_2 = g_1 q_1 + g_2 q_0 + g_3 q_1 - p_2\sigma_a^2 - p_3(g_1 - p_1)\sigma_a^2\}, \\ \{q_3 = g_1 q_2 + g_2 q_1 + g_3 q_0 - p_3\sigma_a^2\}, \end{cases} \tag{9.81}$$

$$\begin{cases} q_4 = g_1 q_3 + g_2 q_2 + g_3 q_1, \\ q_5 = g_1 q_4 + g_2 q_3 + g_3 q_2, \\ q_6 = g_1 q_5 + g_2 q_4 + g_3 q_3, \end{cases} \tag{9.82}$$

Аналізуючи отриману систему рівнянь (9.81), (9.82), ми ще в роботі [8] звернули увагу на те, що з підсистеми (9.82) цієї системи рівнянь, застосовуючи до даної підсистеми

методику Юла – Уокера, можна знайти усі три параметри авторегресії, а також звернули увагу на те, що у цьому випадку матриці, що використовуються у методиці Юла – Уокера, матимуть вигляд:

$$g = \begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix}, \quad M = \begin{bmatrix} q_3 . q_2 . q_1 \\ q_4 . q_3 . q_2 \\ q_5 . q_4 . q_3 . \end{bmatrix}, \quad q = \begin{bmatrix} q_4 \\ q_5 \\ q_6 \end{bmatrix}, \tag{9.83}$$

а аналогом матричного розв'язку (9.76) буде матричний вираз

$$g = M^{-1} q, \tag{9.84}$$

у якому M^{-1} матриця, обернена до матриці *M*, заданої виразом (9.83).

А далі ми в роботі [8] вчинили так. Для кожного рівняння підсистеми (9.81) сконструювали числові значення алгебраїчної суми складових, які позначили як А, В, С, Д, використовуючи вирази:

$$\{q_0 - g_1 q_1 - g_2 q_2 - g_3 q_3 = \text{А}\}, \tag{9.85}$$

$$\{q_1 - g_1 q_0 - g_2 q_1 - g_3 q_2 = \text{В}\}, \tag{9.86}$$

$$\{q_2 - g_1 q_1 - g_2 q_0 - g_3 q_1 = \text{С}\}, \tag{9.87}$$

$$\{q_3 - g_1 q_2 - g_2 q_1 - g_3 q_0 = \text{Д}\}, \tag{9.88}$$

за допомогою яких та за допомогою співвідношень:

$$\sigma_{\text{а}}^2 = -\frac{\text{Д}}{p_3}, \tag{9.89}$$

$$p_3 = \frac{\text{Д} p_2}{\text{С} - \text{Д}(g_1 - p_1)} \tag{9.90}$$

цю підсистему (9.81) трансформували до вигляду

$$\begin{cases} f_1(p_1, p_2) = 0, \\ f_2(p_1, p_2) = 0, \end{cases} \tag{9.91}$$

де

$$f_1(p_1, p_2) = \text{А} p_2 (\text{С} - \text{Д}(g_1 - p_1)) - (\text{С} - \text{Д}(g_1 - p_1))^2 (-1 + p_1(g_1 - p_1) +$$
$$+ p_2(g_2 - p_2)) - \text{Д} p_2 (g_3 (\text{С} - \text{Д}(g_1 - p_1)) - \text{Д} p_2),$$
$$f_2(p_1, p_2) = \text{В} p_2 - \big(\text{С} - \text{Д}(g_1 - p_1)\big)\big(p_1 + p_2(g_1 - p_1)\big) - \text{Д} p_2 (g_2 - p_2).$$

Розв'язуючи систему рівнянь (9.91), знаходимо параметри p_1, p_2, а зі співвідношень (9.90) та (9.89) знаходимо параметри p_3, σ_a^2.

Але, якщо в процесі визначення параметрів $p_1,\ p_2, p_3$ виявиться, що $p_3 = 0$, дисперсію білого шуму потрібно визначати з першого рівняння системи (9.81) з урахуванням виразу (9.85), тобто використовуючи вираз

$$\sigma_a^2 = \frac{A}{1 - p_1(g_1 - p_1) - p_2(g_2 - p_2) - p_3(g_3 - p_3)}. \tag{9.92}$$

А завершуючи викладення нашого удосконаленого методу ідентифікації, ми в роботах [8], [11] акцентували увагу на те, що, на відміну від алгоритму ідентифікації моделі АРКС(p,q) за методикою, викладеною в роботах [9,10], де і параметри регресії, і параметри ковзного середнього пропонується визначати, використовуючи одні і ті ж значення

автоковаріацій та застосовуючи процедуру мінімізації суми квадратів відхилень для пошуку оцінок параметрів ковзного середнього при 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, а 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.

Всі моделі часових рядів, що побудовані вище, базувались на умові стаціонарності цих рядів. Але у повсякденному житті постійно стикаємося і з нестаціонарними випадковими процесами. Наприклад, це процеси пуску або гальмування будь-якого технологічного обладнання, яке реалізує технологічний процес стохастичного характеру.

Покажемо, що такі нестаціонарні випадкові процеси, які при їх дискретизації перетворюються на часові ряди, можна досить адекватно описувати за допомогою моделі, в якій закладені оператори авторегресії – проінтегрованого ковзного середнього.

Для їх синтезу припустимо, що у моделі АРКС(p,q), поданій виразом (9.78), оператор $\Phi(B)$ має *d* кратних коренів, що дорівнюють одиниці.

У цьому випадку, згідно з теоремою Вієтта, оператор $\Phi(B)$ можна записати у вигляді

$$\Phi(B)=(1-B)^d\cdot\left(1-\phi_1^*B-\phi_2^*B^2-\ \ldots-\phi_l^*B^l\right), \tag{9.93}$$

де

$$d+l=p\cdot \tag{9.94}$$

Позначимо

$$\Phi^*(B)=1-\phi_1^*B-\phi_2^*B^2-\ \ldots-\phi_l^*B^l\cdot \tag{9.95}$$

З урахуванням виразів (9.93) та (9.95) рівняння (9.78) можна переписати так

$$\Phi^*(B)\cdot(1-B)^d\,\tilde{z}_t=\Theta(B)a_t\cdot \tag{9.96}$$

Оскільки, згідно з виразом (9.46),

$$(1-B)^d=\nabla^d, \tag{9.97}$$

тобто, $(1-B)^d$ є різницевим оператором із зсувом назад порядку *d*, то рівняння

$$(1-B)^d\,z_t=\nabla^d z_t \tag{9.98}$$

задає нову змінну w_t, яка пов'язана з z_t співвідношенням

$$w_t=\nabla^d z_t\cdot \tag{9.99}$$

Підставляючи вираз (9.99) у рівняння (9.96), отримуємо

$$\Phi^*(B)w_t=\Theta(B)a_t\cdot \tag{9.100}$$

Очевидно, що вираз (9.100) задає модель АРКС (l,q) відносно w_t, яке можна переписати і так

$$w_t = \phi_1^* w_{t-1} + \phi_2^* w_{t-2} + \ \ldots + \phi_l^* w_{t-l} + \\ + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \ \ldots - \theta_q a_{t-q}. \tag{9.101}$$

Рівняння (9.99), (9.100) задають модель нестаціонарного часового ряду z_t у вигляді авторегресії – проінтегрованого ковзного середнього порядку (l,q,d). Скорочено: модель АРПКС (l,q,d).

Звертаємо увагу на те, що вже перша різниця ∇z_t значень будь-якого нестаціонарного часового ряду z_t має менший ступінь нестаціонарності, ніж часо-вий ряд z_t. Ще менший ступінь нестаціонарності матиме друга різниця $\nabla^2 z_t$, яка є різницею перших різниць $\nabla(\nabla z_t)$ цього часового ряду z_t.

Підвищуючи порядок d різниці $\nabla^d z_t$, рано чи пізно дійдемо до такого її значення w_t, яке вже являтиме собою стаціонарний часовий ряд відносно w_t. На рис. 9.5 наведена графічна інтерпретація цього факту.

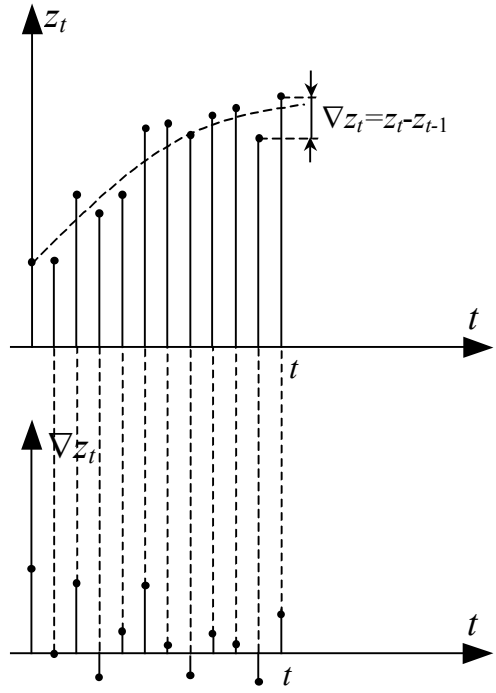


Рисунок 9.5 – графічна інтерпретація перетворення нестаціонарного часового ряду z_t у стаціонарний часовий ряд для його різниці ∇z_t

Зрозуміло, що з моделі АРПКС(l,q,d) (9.103), (9.104) при $d=0$ отримаємо модель АРКС(p,q).

Дамо пояснення чому у назві моделі АРПКС(l,q,d) має місце слово «проінтегроване» стосовно ковзного середнього.

Нагадаємо, що оберненим оператором для ∇ є оператор суми S (9.48). Тому, отримавши w_t з рівняння (9.100), для переходу до часового ряду z_t потрібно координату w_t просумувати d разів, оскільки, помножуючи рівняння (9.85) зліва на ∇^{-d}, матимемо

$$z_t = \nabla^{-d} w_t = S^d w_t . \tag{9.102}$$

Зрозуміло, що найскладнішим завданням під час використання моделі АРПКС (l,q,d) є визначення числового значення параметра інтегрування d, або, в інших термінах, визначення кількості різниць, які потрібно послідовно взяти від нестаціонарного часового ряду z_t, щоб перетворити його у стаціонарний ряд відносно якоїсь різниці цього ряду. Очевидно, що його потрібно розв'язувати шляхом підстановки значення z_t, отриманого за виразом (9.102), та експериментального значення цієї координати в критеріальний функціонал (9.41) і пошуку мінімального значення цього функціонала в наближеній інтерпретації.

Приклади застосування викладеного матеріалу для розв'язання конкретних задач розглянемо уже в програмах мовою Python/

9.5 Додаткові відомості з мови програмування Python, достатні для розв'язання задач, пов'язаних з застосуванням дискретних операторів класу регресійних

Аналізуючи вирази (9.50), (9.56), (9.59), (9.77), (9.80). (9.105) та згадуючи ті відомості з мови Python, які нами уже наведені у першій частині цього навчального посібника та у попередніх підрозділах частини другої, бачимо, що для реалізації цих виразів в програмах мовою Python уже викладеної нами інформації недостатньо, тому що ще не йшла мова про те, як реалізовувати програмно генерацію імпульсів випадкового характеру, які потрібно «підмішувати» до детермінованої частини авторегресійних моделей часових рядів для забезпечення адекватності цих моделей реальним дискретним стохастичним процесам, якими і є часові ряди, що у такий спосіб моделюються. Тож цей підрозділ ми і присвятимо викладенню інформації, присвяченої генеруванню чисел випадкового характеру, без використання якої неможливо синтезувати авторегресійні оператори для моделювання часових рядів, оскільки саме завдяки «зрізанню» цими числами детермінованої послідовності δ-функцій, що генерується в моменти часу, кратні періоду дискретизації часового ряду, ми і отримуємо можливість генерувати послідовність імпульсів з амплітудою, що визначається значенням згенерованого у цей же момент часу випадкового числа. При викладенні цієї інформації ми використовуватимемо відомості, почерпнуті з присвячених програмуванню мовою Python базових робіт [3], [5].

І почнемо ми викладення потрібного нам у-подальшому матеріалу з того, що акцентуємо увагу на наявності в Python двох модулів, що реалізують функції генерації випадкових чисел, а саме: модуля ***random***, який входить в усі три дистрибутиви мови Python та викликається при використанні будь-якого з них саме за цим іменем, а для виклику функцій з нього, як правило, застосовується символ ***rnd,*** що присвоюється цьому модулю

після його виклику; та однойменного модуля ***random*** в ППП ***numpy,*** для виклику якого обов'язково перед цим потрібно викликати сам ППП ***numpy,*** а потім викликати цей модуль потрібно, використовуючи ім'я ***numpy.random,*** або ***np.random***, якщо пакету ***numpy*** присвоєно символ ***np***. Обидва ці модулі містять в собі багато функцій, частина з яких дублюють однойменні функції модуля-конкурента, тож характеристика їх усіх зайняла б забагато сторінок, а тому ми, виходячи з наших цілей, розглянемо в межах кожного з цих двох модулів лише деякі з них.

Отже, важливими для нас, виходячи з наших цілей, в модулі ***random*** є функції: ***rnd.randrange*** (*start, stop, step*), яка вибирає випадкове ціле число ***y*** із діапазону [start, stop) випадкових цілих чисел, що відрізняються між собою не менше ніж на *step;* ***rnd.randint*** (*start, stop*), яка вибирає випадкове ціле число ***y*** із діапазону [start, stop] випадкових цілих чисел; ***rnd.uniform*** (*start, stop*), яка вибирає випадкове раціональне число ***y*** із інтервалу (start, stop); ***rnd.random*** ( ), яка вибирає випадкове раціональне число ***y*** із одиничного відрізка [0, 1]; ***rnd.choice*** ($\{y_i\}$), яка вибирає випадкове число $\boldsymbol{y_i}$ із послідовності $\{\boldsymbol{y_i}\}$, заданої у формі списку, кортежу або масиву; ***rnd.sample*** ($\{y_i\}, k$), яка створює список довжиною ***k*** випадково вибраних елементів із послідовності $\{y_i\}$; ***rnd.seed*** ( ), яка запускає генератор випадкових чисел, що реалізує одну з наведених вище функцій, з прив'язкою до системного часу. Як працюють вищезгадані функції продемонстровано у пояснювальному прикладі № 15.

Пояснювальний приклад № 15

```
In [1]: import random as rnd                 # Виклик модуля random як rnd
In [2]: rnd.seed( )                          # Виклик із модуля random
                                             ….функції seed () Ініціалізації
                                             ….генератора випадкових чисел
In [3]: rnd.randrange (3,30,4)               # Чотири варіанти вибору
Out[3]: 3                                    ….цілого випадкового числа
In [4]: rnd.randrange (3,30,4)               ….з заданої множини їх значень,
Out[4]: 7                                    ….починаючи з «трійки», кожне
In [5]: rnd.randrange (3,30,4)               ….з яких відрізняється на
Out[5]: 23                                   ….«чотири» одиниці, але не
In [6]: rnd.randrange(3,30,4)                ….збігається зі значенням
Out[6]: 15                                   ….правої межі множини
In [7]: rnd.randint (3,30)                   # Три варіанти вибору
Out[7]: 17                                   ….цілого випадкового числа
In [8]: rnd.randint (3,30)                   ….з заданої множини їх значень,
Out[8]: 29                                   ….що відрізняються на одиницю,
In [9]: rnd.randint(3,30)                    ….включно зі значеннями
Out[9]: 5                                    ….на обох межах
In [10]: rnd.uniform (3,30)                  # Три варіанти вибору
Out[10]: 20.536456598318562                  ….раціонального випадкового
In [11]: rnd.uniform (3,30)                  ….числа з заданого
Out[11]: 9.19975053747536                    ….інтервалу їх значень,
In [12]: rnd.uniform (3,30)                  ….без використання значень
Out[12]: 13.09875589535126                   ….на обох межах
In [13]: rnd.random ()                       # Три варіанти вибору
Out[13]: 0.841950248480422                   ….раціонального випадкового
```

```
In [14]: rnd.random ()                                  ….числа з одиничного відрізка
Out[14]: 0.5516994841556719                             ….їх значень включно зі
In [15]: rnd.random ()                                  ….значеннями на обох
Out[15]: 0.1517889391017414                             ….межах
In [16]: rnd.choice ([1,3,5,7,9,11,13,15])              # Три варіанти випадкового
Out[16]: 3                                              ….вибору елемента з заданої
In [17]: rnd.choice ([1,3,5,7,9,11,13,15])              ….списком або кортежом
Out[17]: 1                                              ….послідовності включно з її
In [18]: rnd.choice ([1,3,5,7,9,11,13,15])              ….значеннями на початку та
Out[18]: 13                                             ….та в кінці
In [19]: rnd.sample ([1,3,5,7,9,11,13,15],3)            # Три варіанти формування
Out[19]: [3, 5, 9]                                      ….списку з «трьох» елементів,
In [20]: rnd.sample ([1,3,5,7,9,11,13,15],3)            ….відібраних випадково з
Out[20]: [3, 5, 13]                                     ….заданої списком або
In [21]: rnd.sample ([1,3,5,7,9,11,13,15],3)            ….кортежом послідовності
Out[21]: [11, 3, 9]                                     ….включно зі значеннями
                                                        ….на початку та в кінці
```

Кінець пояснювального прикладу № 15

Що ж до модуля ***numpy.random*** в ППП ***numpy***, символьно позначеного як ***np***, то в ньому нас цікавитимуть лише функції, які генерують масиви випадкових чисел. І почнемо ми їх характеристику з функції, яка у варіанті ***np.random.sample*** **( )** генерує лише одне раціональне число з відрізку [0,1], у варіанті ***np.random.sample*** **(*k*)** генерує уже масив із ***k*** раціональних випадкових чисел з цього ж відрізка, а у варіанті ***np.random.sample*** **((*k,m*))** генерує уже навіть двовимірний масив раціональних випадкових чисел, який містить ***k*** рядків та ***m*** стовпців. Варто відзначити, що масив із ***k*** раціональних випадкових чисел з цього ж відрізка [0, 1] можна згенерувати також і функціями ***np.random.random*** **(*k*)** та ***np.random.rand*** **(*k*)**, а функція ***np.random.rand*** **((*k,m*))**, як і функція ***np.random.sample*** **((*k, m*))**, генерує двовимірний масив раціональних випадкових чисел із відрізка [0, 1], який містить ***k*** рядків та ***m*** стовпців. А масив із ***k*** цілих випадкових чисел з діапазону **(*p, q*)** можна згенерувати за допомогою функції ***np.random.randint*** **(*p, q, k*)**, якою у варіанті ***np.random.randint*** **(*p, q,* (*k, m*))** можна згенерувати і двовимірний масив цілих випадкових чисел із діапазону **(*p, q*)**, який містить ***k*** рядків та ***m*** стовпців. Ну і нарешті, якщо нам потрібно згенерувати одновимірний масив із ***k*** випадкових раціональних чисел, заданих на відрізку **[*a, b*]**, то це ми можемо зробити за допомогою функції ***np.random.uniform*** **(*a, b, k*)**, а якщо потрібно згенерувати двовимірний масив раціональних випадкових чисел із цього ж відрізка **[*a, b*]**, який містить ***k*** рядків та ***m*** стовпців, то це ми можемо зробити за допомогою функції ***np.random.uniform*** **(*a, b,* (*k, m*)).** Як працюють згадані вище функції продемонстровано у пояснювальному прикладі № 16.

Пояснювальний приклад № 16

```
In [1]: import numpy as np                              # Виклик ППП numpy як np
In [2]: np.random.sample( )                             # Виклик двічі підряд функції,
Out[2]: 0.4203889601376536                              ….яка генерує одне
In [3]: np.random.sample( )                             ….раціональне випадкове
Out[3]: 0.13018138113158662                             ….число з відрізка [0, 1]
In [4]: np.random.sample(3)                             # Виклик двічі підряд функції,
```

```
Out[4]: array([0.28426121, 0.74714427,
0.60585534])
In [5]: np.random.sample(3)
Out[5]: array([0.65490975, 0.19625638,
0.96158736])
In [6]: np.random.sample((3,2))
Out[6]:
array([[0.87792665, 0.50704843],
       [0.93988959, 0.07967668],
       [0.10346116, 0.43782785]])
In [7]: np.random.sample((3,2))
Out[7]:
array([[0.09138295, 0.13226296],
       [0.5639563 , 0.06166579],
       [0.54173701, 0.46364738]])
In [8]: np.random.random(3)
Out[8]: array([0.10687142, 0.61193353,
 0.7206429 ])
In [9]: np.random.random(3)
Out[9]: array([0.78718971, 0.88947328,
0.12267282])
In [10]: np.random.rand(3)
Out[10]: array([0.45619613, 0.78303445,
0.83212188])
In [11]: np.random.rand(3)
Out[11]: array([0.73639124, 0.68116394,
0.19805692])
In [12]: np.random.rand(3,2)
Out[12]:
array([[0.89741184, 0.82107899],
       [0.62675848, 0.87184645],
       [0.72502582, 0.19836619]])
In [13]: np.random.rand(3,2)
Out[13]:
array([[0.75999602, 0.3734667 ],
       [0.98139349, 0.305668  ],
       [0.06986268, 0.36318183]])
In [14]: np.random.randint (3,21,3)
Out[14]: array([13, 8, 11])
In [15]: np.random.randint(3,21,3)
Out[15]: array([10, 12, 17])
In [16]: np.random.randint (3,21,(3,2))
Out[16]:
array([[ 8,  8],
       [ 7, 17],
       [ 4, 14]])
In [17]: np.random.randint(3,21,(3,2))
```

```
….яка генерує масив із
….трьох раціональних
….випадкових чисел
….із відрізка [0, 1]

# Виклик двічі підряд функції,
….яка генерує двовимірний
….масив із раціональних
….випадкових чисел із
….відрізка [0, 1], розміщених
….у трьох рядках та
….двох стовпцях

# Виклик двічі підряд ще
….однієї функції,
….яка генерує масив із
….трьох випадкових
….раціональних чисел
….із відрізка [0, 1]
# Виклик двічі підряд ще й
….третьої функції,
….яка генерує масив із
….трьох випадкових
….раціональних чисел
….із відрізка [0, 1]
# Використання цієї ж
….третьої функції для
….генерації двовимірного
….масиву з випадкових
….раціональних чисел
….із відрізка [0, 1], розміщених
….у трьох рядках та
….двох стовпцях, двічі
….підряд

# Виклик двічі підряд функції,
….яка генерує масив із трьох
….цілих чисел із діапазону (3, 21)

# Використання цієї ж
….функції для генерації
….двічі підряд двовимірного
….масиву з випадкових
.цілих чисел із діапазону (3, 21),
….розміщених у трьох рядках
```

```
Out[17]:                                                    ….та двох стовпцях
array([[ 5, 12],
       [14, 16],
       [12,  4]])
In [18]: np.random.uniform(3,21,3)                          # Виклик двічі підряд функції,
Out[18]: array([ 3.63378103, 17.82438909,                   ….яка генерує масив із трьох
 5.26676606])                                               ….раціональних випадкових
In [19]: np.random.uniform(3,21,3)                          ….чисел із
Out[19]: array([ 7.53544443,  9.18792441,                   ….діапазону (3, 21)
17.42777036])
In [20]: np.random.uniform(3,21,(2,3))                      # Використання цієї ж
Out[20]:                                                    ….функції для генерації
array(                                                      ….двічі підряд двовимірного
[[ 4.14709428,  4.38411862, 11.56849929],                   ….масиву з випадкових
[19.87322246, 18.28741987, 17.48519693]])                   ….раціональних чисел
In [21]: np.random.uniform(3,21,(2,3))                      ….із діапазону (3, 21),
Out[21]:                                                    ….розміщених у двох рядках
array(                                                      ….та трьох стовпцях
[[16.70131452, 16.44907401, 11.93865979],
[ 3.37637479,  3.24072743, 12.26041777]])
```

Кінець пояснювального прикладу № 16

9.6 Задачі щодо застосування дискретних операторів класу регресійних в програмах мовою Python

Програма мовою Python для розв'язання задачі, пов'язаної з авторегресійною ідентифікацією стаціонарного часового ряду

y_t = {5., 8., 3., 4., 6., 3., 2., 7., 5., 4., 3., 6., 4., 5., 3., 8., 6., 4., 3., 5., 4., 2., 7., 4., 5., 3., 6., 3., 4., 5.}, заданого в N точках $t \in [0, N-1]$ при N = 30, з використанням моделі АР(3) зі структурою:

$$y_t = b + m_t,$$

$$b = \frac{1}{N}\sum_{i=1}^{N} y_i,$$

$$m_t = g_1 m_{t-1} + g_2 m_{t-2} + g_3 m_{t-3} + a_t$$

та алгоритму Юла – Уокера визначення її параметрів

(Програма 30)

```
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)
Out[5]: 137.0
```

Використання сформованої
….функції для визначення суми
….членів часового ряду,
….заданого списком ***L***

```
In [6]: b=_/N;b
Out[6]: 4.566666666666666
In [7]: b=b.round(3);b
Out[7]: 4.567
In [8]: L1=L-b;L1
Out[8]:
array([ 0.433,  3.433, -1.567, -0.567,  1.433,
 -1.567, -2.567,  2.433,  0.433, -0.567, -1.567,
  1.433, -0.567,  0.433, -1.567,  3.433,
      1.433, -0.567, -1.567,  0.433, -0.567,
-2.567,  2.433, -0.567,  0.433, -1.567,  1.433,
 -1.567, -0.567,  0.433])
```

Обчислення середнього
….значення ***b*** часового ряду
….та залишення в ньому лише
….трьох знаків після коми
Центрування часового ряду
….шляхом віднімання від
….кожного його члена
середнього значення
….цього ряду

```
In [9]: def fun (x):
            return np.dot(x,x)
```

Формування функції, яка
….визначає скалярний добуток
….масиву ***x*** з самим собою
…. в ***numpy***

```
In [10]: fun(L1)
Out[10]:
77.36667
```

Використання сформованої
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….заданого списком ***L1,*** з самим
….собою

```
In [11]: q0=_/N; q0
Out[11]:
 2.5788889999999998
In [12]: q0=q0.round(3); q0
Out[12]:
 2.579
```

Обчислення дисперсії ***q0***
….часового ряду та залишення
….в ній лише трьох знаків
….після коми

```
In [13]: L2=L1[:-1];L2
Out[13]:
array([ 0.433,  3.433, -1.567, -0.567,  1.433,
-1.567, -2.567,  2.433,  0.433, -0.567, -1.567,
  1.433, -0.567,  0.433, -1.567,  3.433,  1.433,
-0.567, -1.567,  0.433, -0.567, -2.567,  2.433,
-0.567,  0.433, -1.567,  1.433, -1.567, -0.567])
```

Вилучення зі списку ***L1***
….центрованого часового ряду
….останнього члена та
….трансформація його у
….список ***L2***

```
In [14]: L5=L1[1:];L5
Out[14]:
array([ 3.433, -1.567, -0.567,  1.433, -1.567,
-2.567,  2.433,  0.433, -0.567, -1.567,  1.433,
-0.567,  0.433, -1.567,  3.433,  1.433,  -0.567,
 -1.567,  0.433, -0.567, -2.567,  2.433, -0.567,
 0.433, -1.567,  1.433, -1.567, -0.567,  0.433])
```

Вилучення зі списку ***L1***
….центрованого часового ряду
….першого члена та
….трансформація його у
….список ***L5***

```
In [15]: def fun (x,y):
             return np.dot(x,y)

In [16]: fun(L2,L5)
Out[16]:
 -22.820818999999993

In [17]: q1=_/(N-1); q1
Out[17]:
 -0.786924793103448
In [18]: q1=q1.round(3);q1
Out[18]:
 -0.787
In [19]: L3=L2[:-1];L3
Out[19]:
array([ 0.433,  3.433, -1.567, -0.567,  1.433,
-1.567, -2.567,  2.433,  0.433, -0.567, -1.567,
 1.433, -0.567,  0.433, -1.567,  3.433,  1.433,
-0.567, -1.567,  0.433, -0.567, -2.567,  2.433,
 -0.567,  0.433, -1.567,  1.433, -1.567])
In [20]: L6=L5[1:];L6
Out[20]:
array([-1.567, -0.567,  1.433, -1.567, -2.567,
  2.433,  0.433, -0.567, -1.567,  1.433, -0.567,
  0.433, -1.567,  3.433,  1.433, -0.567, -1.567,
  0.433, -0.567, -2.567,  2.433, -0.567,  0.433,
 -1.567,  1.433, -1.567, -0.567,  0.433])
In [21]: fun(L3,L6)
Out[21]:
-14.874308000000001

In [22]: q2=_/(N-2);q2
Out[22]:
 -0.5312252857142857
In [23]: q2=q2.round(3);q2
Out[23]:
-0.531
In [24]: L4=L3[:-1];L4
Out[24]:
```

```
# Формування функції, яка
….визначає скалярний добуток
….масиву x з масивом y
# Використання сформованої
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….укороченого зліва і справа
….на одну одиницю у формі
….списків L2 та L5
# Обчислення автоковаріації q1
….часового ряду та залишення
….в ній лише трьох знаків
….після коми

# Вилучення зі списку L2
….укороченого на одиницю
….справа часового ряду
….останнього члена та
….трансформація його у
….список L3

# Вилучення зі списку L5
….укороченого на одиницю
….зліва часового ряду
….першого члена та
….трансформація його у
….список L6

# Використання сформованої
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….укороченого зліва і справа
….на дві одиниці у формі
….списків L3 та L6
# Обчислення автоковаріації q2
….часового ряду та залишення
….в ній лише трьох знаків
….після коми

# Вилучення зі списку L3
….укороченого на дві одиниці
```

```
array([ 0.433,  3.433, -1.567, -0.567,  1.433,
-1.567, -2.567,  2.433,  0.433, -0.567, -1.567,
 1.433, -0.567,  0.433, -1.567,  3.433,  1.433,
-0.567, -1.567,  0.433, -0.567, -2.567,  2.433,
 -0.567,  0.433, -1.567,  1.433])
In [25]: L7=L6[1:];L7                                  # Вилучення зі списку L6
Out[25]:                                               ….укороченого на дві одиниці
array([-0.567,  1.433, -1.567, -2.567,  2.433,         ….зліва часового ряду
  0.433, -0.567, -1.567,  1.433, -0.567,  0.433,       ….першого члена та
 -1.567,  3.433,  1.433, -0.567, -1.567,  0.433,       ….трансформація його у
 -0.567, -2.567,  2.433, -0.567,  0.433, -1.567,       ….список L7
  1.433, -1.567, -0.567,  0.433])
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]: r0=q0/q0;r0                                   # Обчислення автокореляції r0
Out[29]:
 1.0
In [30]: r1=q1/q0;r1                                   # Обчислення автокореляції r1
Out[30]:                                               ….та залишення в ній
 -0.30515703761147733                                  ….лише трьох знаків
In [31]: r1=r1.round(3);r1                             ….після коми
Out[31]:
 -0.305
In [32]: r2=q2/q0;r2                                   # Обчислення автокореляції r2
Out[32]:                                               ….та залишення в ній
 -0.2058937572702598                                   ….лише трьох знаків
In [33]: r2=r2.round(3);r2                             ….після коми
Out[33]:
 -0.206
In [34]: r3=q3/q0;r3                                   # Обчислення автокореляції r3
Out[34]:                                               ….та залишення в ній
 0.038386971694455214                                  ….лише трьох знаків
In [35]: r3=r3.round(3);r3                             ….після коми
Out[35]:
 0.038
```

```
In [36]: L9=[r0,r1,r2,r3];L9                    # Формування списку L9
Out[36]:                                        ….значень автокореляцій
 [1.0, -0.305, -0.206, 0.038]
In [37]: import sympy                           # Виклик ППП sympy
In [38]: from sympy import*                     ….та усіх його функцій
In [39]: r,r0,r1,r2,r3 = symbols('r r0 r1 r2 r3')  # Символізація автокореляцій
In [40]: M = symbols('M')                       # Символізація матриці Юла –
In [41]: M = Matrix([[r0,r1,r2],[r1,r0,r1],     ….Уокера та її форматування
[r2,r1,r0]]);M                                  ….в загальному вигляді
Out[41]:                                        ….для авторегресії АР(3)
Matrix([
[r0, r1, r2],
[r1, r0, r1],
[r2, r1, r0]])
In [42]: g,g1,g2,g3=symbols('g g1 g2 g3')       # Символізація параметрів
In [43]: g = Matrix([g1,g2,g3]);g               ….авторегресії та форматування
Out[43]:                                        ….матриці-стовпця з них у
Matrix([                                        ….загальному вигляді
[g1],
[g2],
[g3]])
In [44]: M =M.subs([(r0,1),(r1,-0.305),         # Ідентифікація матриці
(r2,-0.206)]);M                                 ….Юла – Уокера шляхом
Out[44]:                                        ….підставлення в неї
Matrix([                                        ….числових значень
[     1, -0.305, -0.206],                       ….автокореляцій
[-0.305,      1, -0.305],
[-0.206, -0.305,      1]])
In [45]: r=Matrix([r1,r2,r3]); r                # Форматування матриці-
Out[45]:                                        ….стовпця автокореляцій в
Matrix([                                        ….загальному вигляді
[r1],
[r2],
[r3]])
In [46]: r=r.subs([(r1,-0.305),(r2,-0.206),     # Ідентифікація матриці
(r3,0.038)]); r                                 ….автокореляцій шляхом
Out[46]:                                        ….підставлення в неї
Matrix([                                        ….їх числових значень
[-0.305],
[-0.206],
[ 0.038]])
In [47]: B=simplify(M.inv());B                  # Обчислення матриці,
Out[47]:                                        ….оберненої до матриці
Matrix([                                        ….Юла – Уокера
[ 1.23702975377247, 0.501685993913973,
  0.40784235742089],[0.501685993913973,
  1.30602845628752, 0.501685993913973],
```

```
[ 0.40784235742089, 0.501685993913973,
  1.23702975377247]])
In [48]: g=B*r; g                          # Розв'язання системи
Out[48]:                                   ….рівнянь Юла – Уокера, яка в
Matrix([                                   ….матричній формі має вигляд
[-0.465143380064886],                      ….M g = r, та отримання
0.402992022370261],                        …. матриці-стовпця g параметрів
[-0.180732103116296]])                     ….авторегресії АР(3)
In [49]: g=g.evalf(3); g                   # Скорочення до трьох знаків
Out[49]:                                   ….після коми в числових
Matrix([                                   ….значеннях параметрів
[-0.465],                                  ….авторегресії
[-0.403],
[-0.181]])
In [50]: g1=g[0,0]; g1                     # Ідентифікація параметра g1
Out[50]:
-0.465
In [51]: g2=g[1,0]; g2                     # Ідентифікація параметра g2
Out[51]:
-0.403
In [52]: g3=g[2,0]; g3                     # Ідентифікація параметра g3
Out[52]:
-0.181
In [53]: a = symbols ('a')                 # Символізація імпульсу a
                                           ….білого шуму
In [54]: ska = symbols ('ska')             # Символізація дисперсії ska
                                           ….білого шуму
In [55]: ska = q0-g1*q1-g2*q2-g3*q3; ska   # Обчислення дисперсії ska
Out[55]:                                   ….за алгоритмом Юла – Уокера
2.01681854248047
In [56]: skv = symbols ('skv')             # Символізація середньо-
In [57]: skv = ska**(0.5); skv             ….квадратичного відхилення
Out[57]:                                   ….skv білого шуму та його
1.42014736646605                           ….обчислення
In [58]: skv = skv.evalf(3); skv           # Скорочення до трьох знаків
Out[58]:                                   ….після коми в числовому
1.42                                       ….значенні skv
In [59]: a11,a22 = symbols ('a11 a22')     # Оголошення символьними
                                           ….меж діапазону генерації
                                           ….імпульсів білого шуму
In [60]: a11 = -3*skv; a11                 # Визначення нижньої a11 та
Out[60]:                                   ….верхньої a22 меж
-4.26                                      ….діапазону генерації імпульсів
In [61]: a22 = 3*skv; a22                  ….білого шуму за умови, що
Out[61]:                                   ….білий шум розподілений за
```

4.26

```
In [62]: import random as rnd

In [63]: m = symbols('m:10');m
Out[63]:
(m0, m1, m2, m3, m4, m5, m6, m7, m8, m9)
In [64]: l = symbols ('l:10');l
Out[64]:
 (l0, l1, l2, l3, l4, l5, l6, l7, l8, l9)
In [65]: m=list(m);m
Out[65]:
 [m0, m1, m2, m3, m4, m5, m6, m7, m8, m9]
In [66]: l=list(l);l
Out[66]:
 [l0, l1, l2, l3, l4, l5, l6, l7, l8, l9]
In [67]: d = symbols (‘d:10’); d
Out[67]:
(d0,d1,d2,d3,d4,d5,d6,d7,d8,d9)
In [68]: d = list (d); d
Out[68]:
[d0,d1,d2,d3,d4,d5,d6,d7,d8,d9]
In [69]: d[0] = rnd.uniform (-4.26,4.26); d[0]
Out[69]:
3.7878324717658174
In [70]: m[0]=g1*L1[29]+g2*L1[28]+\
g3*L1[27]+ d[0]; m[0]
Out[70]:
4.09812879744941
In [71]: L1 = np.append(L1,[m[0]]);L1
Out[71]:
array([0.43299999999999983, 3.433,
 -1.5670000000000002, -0.5670000000000002,
 1.4329999999999998, -1.5670000000000002,
 -2.567, 2.433, 0.43299999999999983,
-0.5670000000000002, -1.5670000000000002,
1.4329999999999998, -0.5670000000000002,
  0.43299999999999983, -1.5670000000000002,
 3.433,  1.4329999999999998,
-0.5670000000000002, -1.5670000000000002,
  0.43299999999999983, -0.5670000000000002,
-2.567, 2.433, -0.5670000000000002,
 0.43299999999999983, -1.5670000000000002,
  1.4329999999999998, -1.5670000000000002,
-0.5670000000000002,  0.43299999999999983,
 4.09812879744941], dtype=object)
```

….нормальним законом

```
# 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 d[0] для першої
….ітерації прогнозу
# Обчислення N+1 -го члена
….списку L1 з використанням
….моделі авторегресії AP(3)

# Додавання в кінець списку
….L1 нового члена m[0]
```

```
In [72]: l[0]=b+m[0]; l[0]                          # Обчислення N+1 -го члена
Out[72]:                                            ….списку L
8.66512879744941
In [73]: L = np.append(L,[l[0]]);L                  # Додавання в кінець списку
Out[73]:                                            ….L нового члена l[0]
array([5.0, 8.0, 3.0, 4.0, 6.0, 3.0, 2.0, 7.0, 5.0,
 4.0, 3.0, 6.0, 4.0, 5.0, 3.0, 8.0, 6.0, 4.0, 3.0,
5.0, 4.0, 2.0, 7.0, 4.0, 5.0, 3.0, 6.0, 3.0, 4.0,
 5.0, 8.66512879744941], dtype=object)
In [74]: d[1] = rnd.uniform (-4.26,4.26); d[1]      # Генерація випадкового
Out[74]:                                            ….імпульсу d[1] для другої
 -2.587298897242724                                 ….ітерації прогнозу
In [75]: m[1]= g1*L1[30]+g2*L1[29]+\                # Обчислення N+2 -го члена
g3*L1[28]+ d[1]; m[1]                               ….списку L1 з використанням
Out[75]:                                            ….моделі авторегресії AP(3)
-2.08074576204602
In [76]: L1 = np.append(L1,[m[1]])                  # Додавання в кінець списку
                                                    ….L1 нового члена m[1]
In [77]: l[1]=b+m[1]; l[1]                          # Обчислення N+2 -го члена
Out[77]:                                            ….списку L
2.48625423795398
In [78]: L=np.append(L,[l[1]])                      # Додавання в кінець списку
                                                    ….L нового члена l[1]
In [79]: d[2] = rnd.uniform (-4.26,4.26); d[2]      # Генерація випадкового
Out[79]:                                            ….імпульсу d[2] для третьої
 0.8071389548290568                                 ….ітерації прогнозу
In [80]: m[2]= g1*L1[31]+g2*L1[30]+\                # Обчислення N+3 -го члена
g3*L1[29]+ d[2]; m[2]                               ….списку L1 з використанням
Out[80]:                                            ….моделі авторегресії AP(3)
0.0451337059883733
In [81]: L1 = np.append(L1,[m[2]])                  # Додавання в кінець списку
                                                    ….L1 нового члена m[2]
In [82]: l[2]=b+m[2]; l[2]                          # Обчислення N+3 -го члена
Out[82]:                                            ….списку L
4.61213370598837
In [83]: L=np.append(L,[l[2]])                      # Додавання в кінець списку
                                                    ….L нового члена l[2]
In [84]: d[3] = rnd.uniform (-4.26,4.26); d[3]      # Генерація випадкового
Out[84]:                                            ….імпульсу d[3] для четвертої
3.5146713080878706                                  ….ітерації прогнозу
In [85]: m[3]= g1*L1[32]+g2*L1[31]+\                # Обчислення N+4 -го члена
g3*L1[30]+ d[3]; m[3]                               ….списку L1 з використанням
Out[85]:                                            ….моделі авторегресії AP(3)
3.59161472387185
In [86]: L1 = np.append(L1,[m[3]])                  # Додавання в кінець списку
```

```
In [87]: l[3]=b+m[3]; l[3]                                  ….L1 нового члена m[3]
Out[87]:                                                    # Обчислення N+4 -го члена
8.15861472387185                                            ….списку L
In [88]: L=np.append(L,[l[3]])

In [89]: d[4] = rnd.uniform (-4.26,4.26); d[4]              # Додавання в кінець списку
Out[89]:                                                    ….L нового члена l[3]
3.332153915638866                                           # Генерація випадкового
In [90]: m[4]= g1*L1[33]+g2*L1[32]+\                        ….імпульсу d[4] для п’ятої
g3*L1[31]+ d[4]; m[4]                                       ….ітерації прогнозу
Out[90]:                                                    # Обчислення N+5 -го члена
0.178801469969288                                           ….списку L1 з використанням
In [91]: L1 = np.append(L1,[m[4]])                          ….моделі авторегресії AP(3)

In [92]: l[4]=b+m[4]; l[4]                                  # Додавання в кінець списку
Out[92]:                                                    ….L1 нового члена m[4]
4.74580146996929                                            # Обчислення N+5 -го члена
In [93]: L=np.append(L,[l[4]])                              ….списку L

In [94]: d[5] = rnd.uniform (-4.26,4.26); d[5]              # Додавання в кінець списку
Out[94]:                                                    ….L нового члена l[4]
2.867785147080328                                           # Генерація випадкового
In [95]: m[5]= g1*L1[34]+g2*L1[33]+\                        ….імпульсу d[5] для шостої
g3*L1[32]+ d[5]; m[5]                                       ….ітерації прогнозу
Out[95]:                                                    # Обчислення N+6 -го члена
1.32898394299138                                            ….списку L1 з використанням
In [96]: L1 = np.append(L1,[m[5]])                          ….моделі авторегресії AP(3)

In [97]: l[5]=b+m[5]; l[5]                                  # Додавання в кінець списку
Out[97]:                                                    ….L1 нового члена m[5]
5.89598394299138                                            # Обчислення N+6 -го члена
In [98]: L=np.append(L,[l[5]])                              ….списку L

In [99]: d[6] = rnd.uniform (-4.26,4.26); d[6]              # Додавання в кінець списку
Out[99]:                                                    ….L нового члена l[5]
1.69270588124252                                            # Генерація випадкового
In [100]: m[6]= g1*L1[35]+g2*L1[34]+\                       ….імпульсу d[6] для сьомої
g3*L1[33]+ d[6]; m[6]                                       ….ітерації прогнозу
Out[100]:                                                   # Обчислення N+7 -го члена
-0.471995713797095                                          ….списку L1 з використанням
In [101]: L1 = np.append(L1,[m[6]])                         ….моделі авторегресії AP(3)

In [102]: l[6]=b+m[6]; l[6]                                 # Додавання в кінець списку
Out[102]:                                                   ….L1 нового члена m[6]
                                                            # Обчислення N+7 -го члена
                                                            ….списку L
```

```
4.09500428620290
In [103]: L=np.append(L,[l[6]])                  # Додавання в кінець списку
                                                 ….L нового члена l[6]
In [104]: d[7] = rnd.uniform (-4.26,4.26); d[7]  # Генерація випадкового
Out[104]:                                        ….імпульсу d[7] для восьмої
0.10021632009046666                              ….ітерації прогнозу
In [105]: m[7]= g1*L1[36]+g2*L1[35]+\            # Обчислення N+8 -го члена
g3*L1[34]+ d[7]; m[7]                            ….списку L1 з використанням
Out[105]:                                        ….моделі авторегресії AP(3)
-0.248149939266434
In [106]: L1 = np.append(L1,[m[7]])              # Додавання в кінець списку
                                                 ….L1 нового члена m[7]
In [107]: l[7]=b+m[7]; l[7]                      # Обчислення N+8 -го члена
Out[107]:                                        ….списку L
4.31885006073357
In [108]: L=np.append(L,[l[7]])                  # Додавання в кінець списку
                                                 ….L нового члена l[7]
In [109]: d[8] = rnd.uniform (-4.26,4.26); d[8]  # Генерація випадкового
Out[109]:                                        ….імпульсу d[8] для дев'тої
-2.232660689069062                               ….ітерації прогнозу
In [110]: m[8]= g1*L1[37]+g2*L1[36]+\            # Обчислення N+9 -го члена
g3*L1[35]+ d[8]; m[8]                            ….списку L1 з використанням
Out[110]:                                        ….моделі авторегресії AP(3)
-2.16719334714070
In [111]: L1 = np.append(L1,[m[8]])              # Додавання в кінець списку
                                                 ….L1 нового члена m[8]
In [112]: l[8]=b+m[8]; l[8]                      # Обчислення N+9 -го члена
Out[112]:                                        ….списку L
2.39980665285930
In [113]: L=np.append(L,[l[8]])                  # Додавання в кінець списку
                                                 ….L нового члена l[8]
In [114]: d[9] = rnd.uniform (-4.26,4.26); d[9]  # Генерація випадкового
Out[114]:                                        ….імпульсу d[9] для десятої
1.544009122634895                                ….ітерації прогнозу
In [115]: m[9]= g1*L1[38]+g2*L1[37]+\            # Обчислення N+10 -го члена
g3*L1[36]+ d[9]; m[9]                            ….списку L1 з використанням
Out[115]:                                        ….моделі авторегресії AP(3)
2.73738643318719
In [116]: L1 = np.append(L1,[m[9]])              # Додавання в кінець списку
                                                 ….L1 нового члена m[9]
In [117]: l[9]=b+m[9]; l[9]                      # Обчислення N+10 -го члена
Out[117]:                                        ….списку L
7.30438643318719
In [118]: L=np.append(L,[l[9]])                  # Додавання в кінець списку
                                                 ….L нового члена l[9]
```

```
In [119]: import matplotlib                          # Виклик ППП matplotlib
In [120]: import matplotlib.pyplot as plt            # Виклик модуля pyplot
In [121]: plt.plot(L1)                               # Друк графіка z_t
```

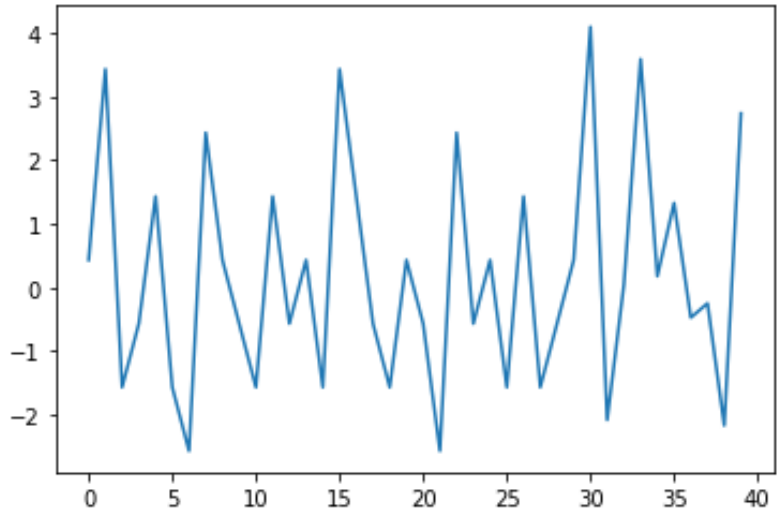


Рисунок 9.6 – Графік часового ряду m_t, який в діапазоні $t \in [0,30]$ заповнений отриманими експериментально значеннями, а за межами цього діапазону заповнений прогнозними значеннями, отриманими з використанням авторегресійної моделі, ідентифікованої з використанням отриманих експериментально значень

Кінець програми 30.

Програма мовою Python для розв'язання задачі, пов'язаної з авторегресійним моделюванням стаціонарного часового ряду
y_t= {5., 8., 3., 4., 6., 3., 2., 7., 5., 4., 3., 6., 4., 5., 3., 8., 6., 4., 3., 5., 4., 2., 7., 4., 5., 3., 6., 3., 4., 5.}, заданого в N точках $t \in [0, N-1]$ при N = 30, з використанням моделі АРКС(3,3) зі структурою:

$$y_t = b + m_t,$$

$$b = \frac{1}{N}\sum_{i=1}^{N} y_i,$$

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

та методу її ідентифікації, запропонованого професорами Мокіними

(Програма 31)

```
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):
            return np.sum(x)

In [5]: fun(L)
Out[5]: 137.0


In [6]: b=_/N;b
Out[6]: 4.566666666666666
In [7]: b=b.round(3);b
Out[7]: 4.567
In [8]: L1=L-b




In [9]: def fun (x):
            return np.dot(x,x)


In [10]: fun(L1)
Out[10]:
77.36667



In [11]: q0=_/N; q0
Out[11]:
 2.5788889999999998
In [12]: q0=q0.round(3); q0
Out[12]:
 2.579
In [13]: L2=L1[:-1]




In [14]: L5=L1[1:]




In [15]: def fun (x,y):
            return np.dot(x,y)

In [16]: fun(L2,L5)
```

```
# Формування функції, яка
….визначає суму членів
….числового масиву x в numpy
# Використання сформованої
….функції для визначення суми
….членів часового ряду,
….заданого списком L
# Обчислення середнього
….значення b часового ряду
….та залишення в ньому лише
….трьох знаків після коми
# Центрування часового ряду
….шляхом віднімання від
….кожного його члена
   середнього значення
….цього ряду
# Формування функції, яка
….визначає скалярний добуток
….масиву x з самим собою
…. в numpy
# Використання сформованої
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….заданого списком L1, з самим
….собою
# Обчислення дисперсії q0
….часового ряду та залишення
….в ній лише трьох знаків
….після коми


# Вилучення зі списку L1
….центрованого часового ряду
….останнього члена та
….трансформація його у
….список L2
# Вилучення зі списку L1
….центрованого часового ряду
….першого члена та
….трансформація його у
….список L5
# Формування функції, яка
….визначає скалярний добуток
….масиву x з масивом y
# Використання сформованої
```

```
Out[16]:
 -22.820818999999993
```
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….укороченого зліва і справа
….на одну одиницю у формі
….списків ***L2*** та ***L5***

```
In [17]: q1=_/(N-1); q1
Out[17]:
 -0.786924793103448
In [18]: q1=q1.round(3);q1
Out[18]:
 -0.787
```
Обчислення автоковаріації ***q1***
….часового ряду та залишення
….в ній лише трьох знаків
….після коми

```
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)
Out[21]:
-14.874308000000001
```
Використання сформованої
….функції для визначення
….скалярного добутку
….центрованого часового ряду,
….укороченого зліва і справа
….на дві одиниці у формі
….списків ***L3*** та ***L6***

```
In [22]: q2=_/(N-2);q2
Out[22]:
 -0.5312252857142857
In [23]: q2=q2.round(3);q2
Out[23]:
-0.531
```
Обчислення автоковаріації ***q2***
….часового ряду та залишення
….в ній лише трьох знаків
….після коми

```
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 подання у
Matrix([                                          …. загальному 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
(g1-p1)+D*p2*(g2-p2)/(C-D*(g1-p1));f11                      ….ідентифікація
                                                            ….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+\    # Розв’язання системи
p1)+ p2*(2.14+p2))*(4.31-2.97*(1.17\            ….f2(p1, p2) = 0, f22(p1, p2) = 0
+p1)))*(4.31-2.97*(1.17+p1))+2.97*p2*\          …. та отримання
 (0.297*(4.31-2.97*(1.17\                       ….коренів у вигляді
+p1))+2.97*p2),0.699*p2-\                       ….списку L15
(p1-p2*(1.17+p1))*(4.31-2.97*(1.17+\
p1))-2.97*p2*(2.14+p2)],p1,p2);L15
Out[84]:
[(0.281178451178451, 0.0)]
In [85]: p1=L15[0][0];p1                        # Визначення параметра
Out[85]:                                        ….p1 та 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]: ska,skv = symbols('ska skv')           # Символізація дисперсії і СКВ
                                                ….білого шуму
In [90]: ska=A/(1-p1*(g1-p1));ska               # Обчислення дисперсії
Out[90]:                                        ….білого шуму
0.350
In [91]: skv=ska**(0.5);skv                     # Обчислення СКВ
Out[91]:                                        ….білого шуму
0.591587344301330                               ….та залишення в його
In [92]: skv=skv.evalf(3);skv                   ….числовому значенні
Out[92]:                                        ….лише трьох знаків після коми
0.592
In [93]: a11,a22 = symbols('a11 a22')           # Символізація та обчислення
In [94]: a11=-2*skv;a11                         ….меж a11, a22 діапазону
Out[94]:                                        ….генерації імпульсів
-1.18                                           ….«нормального» білого шуму
In [95]: a22=2*skv;a22                          ….з довірчою ймовірністю 0,95
Out[95]:                                        ….для «підмішування» в модель
1.18                                            ….АРКС(3,1)
In [96]: p=[p0,p1];p                            # Формування списку параметрів
```

```
Out[96]:
[1.0, 0.280]
In [97]: import random as rnd

In [98]: m = symbols('m:5');m
Out[98]:
(m0, m1, m2, m3, m4)
In [99]: l = symbols ('l:5');l
Out[99]:
 (l0, l1, l2, l3, l4)
In [100]: m=list(m);m
Out[100]:
 [m0, m1, m2, m3, m4]
In [101]: l=list(l);l
Out[101]:
 [l0, l1, l2, l3, l4]
In [102]: d = symbols ('d:5'); d
Out[102]:
 (d0, d1, d2, d3, d4)
In [103]: d = list (d); d
Out[103]:
[d0,d1,d2,d3,d4]
In [104]: d0 = rnd.uniform (-1.18, 1.18); d0
Out[104]:
-0.9786481619579281
In [105]: m0= g1*L1[29]+g2*L1[28]+\
g3*L1[27]+ d0; m0
Out[105]:
-0.735726592133709
In [106]: L1 = np.append(L1,[m0])

In [107]: l0=b+m0; l0
Out[107]:
3.83127340786629
In [108]: L = np.append(L,[l0])

In [109]: d1 = rnd.uniform (-1.18, 1.18); d1
Out[109]:
-0.9602434653342734
In [110]: m1= g1*L1[30]+g2*L1[29]+\
g3*L1[28]+ d1-p1*d0; m1
Out[110]:
-0.922476284076637
In [111]: L1 = np.append(L1,[m1])

In [112]: l1=b+m1; l1
```

….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
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

```
Out[112]:
3.64452371592336
In [113]: L=np.append(L,[l1])

In [114]: d2 = rnd.uniform (-1.18, 1.18); d2
Out[114]:
0.1282150195429812
In [115]: m2= g1*L1[31]+g2*L1[30]+\
g3*L1[29]+ d2-p1*d1; m2
Out[115]:
3.18046983687152
In [116]: L1 = np.append(L1,[m2])

In [117]: l2=b+m2; l2
Out[117]:
7.74746983687152
In [118]: L=np.append(L,[l2])

In [119]: d3 = rnd.uniform(-1.18, 1.18); d3
Out[119]:
-1.133835327586167
In [120]: m3= g1*L1[32]+g2*L1[31]+\
g3*L1[30]+ d3-p1*d2; m3
Out[120]:
-3.12903725295206
In [121]: L1 = np.append(L1,[m3])

In [122]: l3=b+m3; l3
Out[122]:
1.43796274704794
In [123]: L=np.append(L,[l3])

In [124]: d4 = rnd.uniform(-1.18, 1.18); d4
Out[124]:
0.13261500744833254
In [125]: m4= g1*L1[32]+g2*L1[31]+\
g3*L1[30]+ d4-p1*d3; m4
Out[125]:
-1.50894475826974
In [126]: L1 = np.append(L1,[m4])

In [127]: l4=b+m4; l4
Out[128]:
3.05805524173026
In [129]: L=np.append(L,[l4])
```

….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
```

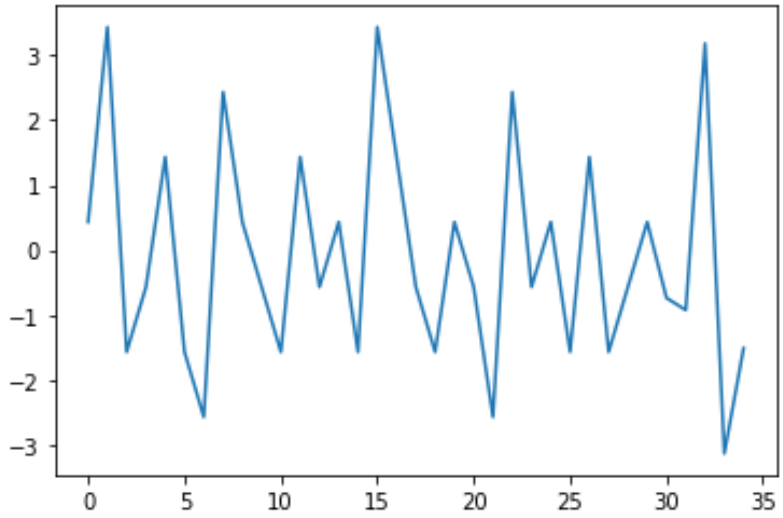


Рисунок 9.7 – Графік часового ряду m_t, який в діапазоні $t \in [0,30]$ заповнений експериментально отриманими значеннями, а за межами цього діапазону заповнений прогнозними значеннями, отриманими з використанням моделі АРКС (3, 1), ідентифікованої з використанням експериментально отриманих значень

Кінець програми 31.

Постскриптум. Вважаємо доцільним повторити уже висловлене нами раніше у роботах [8] та [11] твердження про те, що, на відміну від алгоритму ідентифікації моделі АРКС(n_{ap}, n_{kc}) за методикою, викладеною в роботах [9, 10], де і параметри авторегресії і параметри ковзного середнього пропонується визначати, використовуючи одні і ті ж значення автоковаріацій та застосовуючи процедуру мінімізації суми квадратів відхилень для пошуку оцінок параметрів ковзного середнього при 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, а 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.

Вважаємо за доцільне також акцентувати увагу на тому, що для розрахунку параметрів авторегресійної складової моделі АРКС(3, 3) в базовому класичному методі 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 цьому нашому 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 при розрахунку автоковаріації потрібно скорочувати цю реалізацію з обох боків на кількість членів, що дорівнює порядку автоковаріації. Крім того, потрібно пам'ятати один із 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 відрізка неухильно і стрімко зростає. Отже, на більш високу точність прогнозування наступних значень часового ряду при використанні моделі АРКС(3, 3), ідентифікованої за цим нашим методом, можна розраховувати лише при застосуванні в процесі ідентифікації моделі експериментальної реалізації часового ряду, що містить в собі не менше 60 його значень.

Програма мовою Python для розв'язання задачі, пов'язаної з авторегресійним моделюванням нестаціонарного часового ряду y_t = { 5., 8., 3., 4., 6., 8., 10., 7., 6., 9., 12., 8., 11., 15., 12., 10., 7., 8., 12., 15., 18., 20., 17., 14., 15., 17., 16., 19., 22., 25.}, заданого в N точках $t \in [0, N-1]$ при N = 30, з використанням моделі АРПКС(3, 0, 2) зі структурою:

$$v_t = y_t - y_{t-1}, w_t = v_t - v_{t-1},$$
$$w_t = g_1 w_{t-1} + g_2 w_{t-2} + g_3 w_{t-3} + a_t,$$
$$y_k = y_{-1} + \sum_{t=0}^{k} v_t, \qquad v_k = v_{-1} + \sum_{t=0}^{k} w_t$$

та методу її ідентифікації в класичній формі.

(Програма 32)

```
In [1]: import numpy as np                  # Виклик ППП numpy як np
In [2]: L=[5.,8.,3.,4.,6.,8.,10.,7.,6.,9.,12.,8.,\   # Внесення списку з N
11.,15.,12.,10.,7.,8.,12.,15.,18.,20.,17.,14.,       ….значень часового ряду
15.,17.,16.,19.,22.,25.]
In [3]: N=30                                # Фіксація N = 30
In [4]: L1 = np.diff(L);L1                  # Формування списку L1
Out[4]:                                     ….із масиву перших різниць
array([ 3., -5.,  1.,  2.,  2.,  2., -3., -1.,  3.,  3., \   ….часового ряду, заданого
-4.,  3.,  4., -3., -2., -3.,  1.,  4.,  3.,  3.,  2., -3.,\   ….списком L
 -3.,  1.,  2., -1.,  3.,  3.,  3.])
In [5]: def fun (x):                        # Формування функції, яка
            return np.sum(x)                ….визначає суму членів
                                            ….числового масиву x в numpy
In [6]: fun(L1)                             # Використання сформованої
Out[6]: 20.0                                ….функції для визначення суми
                                            ….членів часового ряду,
                                            ….заданого списком L1
In [7]: _/(N-1)                             # Обчислення середнього
Out[7]: 0.6896551724137931                  ….значення часового ряду,
```

```
                                                    ….заданого списком L1, і
                                                    ….оскільки воно не дорівнює
                                                    ….нулю, то ряд L1 перших
                                                    ….різниць ряду L
                                                    ….не є стаціонарним
In [8]: L11=np.diff(L1);L11                         # Формування списку L11
Out[8]:                                             ….із масиву других різниць
array([-8.,  6.,  1.,  0.,  0., -5.,  2.,  4.,  0.,\  ….часового ряду, заданого
 -7.,  7.,  1., -7.,  1., -1.,  4.,  3., -1.,  0., -1.,\  ….списком L
 -5.,  0.,  4.,  1., -3.,  4.,  0.,  0.])
In [9]: fun(L11)                                    # Використання сформованої
Out[9]: 0.0                                         ….функції для визначення суми
                                                    ….членів часового ряду,
                                                    ….заданого списком L11, і
                                                    ….оскільки вона дорівнює
                                                    ….нулю, то ряд L11 других
                                                    ….різниць ряду L стаціонарний
In [10]: def fun (x):                               # Формування функції, яка
             return np.dot(x,x)                     ….визначає скалярний добуток
                                                    ….масиву x з самим собою
                                                    ….в numpy
In [11]: fun(L11)                                   # Використання сформованої
Out[11]:                                            ….функції для визначення
390.0                                               ….скалярного добутку
                                                    ….центрованого часового ряду,
                                                    ….заданого списком L11, з
                                                    ….самим собою
In [12]: q0=_/(N-2); q0                             # Обчислення дисперсії q0
Out[12]:                                            ….часового ряду L11
13.928571428571429                                  ….та залишення
In [13]: q0=q0.round(3);q0                          ….в ній лише трьох знаків
Out[13]:                                            ….після коми
13.929
In [14]: L2=L11[:-1]                                # Вилучення зі списку L11
                                                    ….центрованого часового ряду
                                                    ….останнього члена та
                                                    ….трансформація його у
                                                    ….список L2
In [15]: L5=L11[1:]                                 # Вилучення зі списку L11
                                                    ….центрованого часового ряду
                                                    ….першого члена та
                                                    ….трансформація його у
                                                    ….список L5
In [16]: def fun (x,y):                             # Формування функції, яка
             return np.dot(x,y)                     ….визначає скалярний добуток
                                                    ….масиву x з масивом y
In [17]: fun(L2,L5)                                 # Використання сформованої
```

```
Out[17]:                                  ….функції для визначення
-102.0                                    ….скалярного добутку
                                          ….центрованого часового ряду,
                                          ….укороченого зліва і справа
                                          ….на одну одиницю у формі
                                          ….списків L2 та L5
In [18]: q1=_/(N-3); q1                   # Обчислення автоковаріації q1
Out[18]:                                  ….часового ряду та залишення
-3.7777777777777777                       ….в ній лише трьох знаків
In [19]: q1=q1.round(3);q1                ….після коми
Out[19]:
-3.778
In [20]: L3=L2[:-1]                       # Вилучення зі списку L2
                                          ….укороченого на одиницю
                                          ….справа часового ряду
                                          ….останнього члена та
                                          ….трансформація його у
                                          ….список L3
In [21]: L6=L5[1:]                        # Вилучення зі списку L5
                                          ….укороченого на одиницю
                                          ….зліва часового ряду
                                          ….першого члена та
                                          ….трансформація його у
                                          ….список L6
In [22]: fun(L3,L6)                       # Використання сформованої
Out[22]:                                  ….функції для визначення
-134.0                                    ….скалярного добутку
                                          ….центрованого часового ряду,
                                          ….укороченого зліва і справа
                                          ….на дві одиниці у формі
                                          ….списків L3 та L6
In [23]: q2=_/(N-4);q2                    # Обчислення автоковаріації q2
Out[23]:                                  ….часового ряду та залишення
-5.153846153846154                        ….в ній лише трьох знаків
In [24]: q2=q2.round(3);q2                ….після коми
Out[24]:
-5.154
In [25]: L4=L3[:-1]                       # Вилучення зі списку L3
                                          ….укороченого на дві одиниці
                                          ….справа часового ряду
                                          ….останнього члена та
                                          ….трансформація його у
                                          ….список L4
In [26]: L7=L6[1:]                        # Вилучення зі списку L6
                                          ….укороченого на дві одиниці
                                          ….зліва часового ряду
                                          ….першого члена та
```

```
                                                    ….трансформація його у
                                                    ….список L7
In [27]: fun(L4,L7)                                 # Використання сформованої
Out[27]:                                            ….функції для визначення
49.0                                                ….скалярного добутку
                                                    ….центрованого часового ряду,
                                                    ….укороченого зліва і справа
                                                    ….на три одиниці у формі
                                                    ….списків L4 та L7
In [28]: q3=_/(N-5); q3                             # Обчислення автоковаріації q3
Out[28]:                                            ….часового ряду
1.96
In [29]: r0=q0/q0;r0                                # Обчислення автокореляції r0
Out[29]:
 1.0
In [30]: r1=q1/q0;r1                                # Обчислення автокореляції r1
Out[30]:                                            ….та залишення в ній
-0.29059829059829057                                ….лише трьох знаків
In [31]: r1=r1.round(3);r1                          ….після коми
Out[31]:
-0.291
In [32]: r2=q2/q0;r2                                # Обчислення автокореляції r2
Out[32]:                                            ….та залишення в ній
-0.3964615384615385                                 ….лише трьох знаків
In [33]: r2=r2.round(3);r2                          ….після коми
Out[33]:
-0.396
In [34]: r3=q3/q0;r3                                # Обчислення автокореляції r3
Out[34]:                                            ….та залишення в ній
0.15076923076923077                                 ….лише трьох знаків
In [35]: r3=r3.round(3);r3                          ….після коми
Out[35]:
0.151
In [36]: L9=[r0,r1,r2,r3];L9                        # Формування списку L9
Out[36]:                                            ….значень автокореляцій
[1.0, -0.291, -0.396, 0.151]
In [37]: import sympy                               # Виклик ППП sympy
In [38]: from sympy import*                         ….та усіх його функцій
In [39]: r,r0,r1,r2,r3 = symbols('r r0 r1 r2 r3')   # Символізація автокореляцій
In [40]: M = symbols('M')                           # Символізація матриці Юла –
In [41]: M = Matrix([[r0,r1,r2],[r1,r0,r1],         ….Уокера та її форматування
[r2,r1,r0]]);M                                      ….в загальному вигляді
Out[41]:                                            ….для авторегресії AP(3)
Matrix([
[r0, r1, r2],
```

```
[r1, r0, r1],
[r2, r1, r0]])
In [42]: g,g1,g2,g3=symbols('g g1 g2 g3')          # Символізація параметрів
In [43]: g = Matrix([g1,g2,g3]);g                  ….авторегресії та форматування
Out[43]:                                           ….матриці-стовпця з них у
Matrix([                                           ….загальному вигляді
[g1],
[g2],
[g3]])
In [44]: M =M.subs([(r0,1),(r1,-0.291),            # Ідентифікація матриці
(r2,-0.396)]);M                                    ….Юла – т Уокера шляхом
Out[44]:                                           ….підставлення в неї
Matrix([                                           ….числових значень
[     1, -0.291, -0.396 ],                         ….автокореляцій
[ -0.291,      1, -0.291],
[-0.396, -0.291,      1 ]])
In [45]: r=Matrix([r1,r2,r3]); r                   # Форматування матриці-
Out[45]:                                           ….стовпця автокореляцій в
Matrix([                                           ….загальному вигляді
[r1],
[r2],
[r3]])
In [46]: r=r.subs([(r1,-0.291),(r2,-0.396),        # Ідентифікація матриці
(r3,0.151)]); r                                    ….автокореляцій шляхом
Out[46]:                                           ….підставлення в неї
Matrix([                                           ….їх числових значень
[ -0.291],
[ -0.396],
[ 0.151]])
In [47]: B=simplify(M.inv());B                     # Обчислення матриці,
Out[47]:                                           ….оберненої до матриці
Matrix([                                           ….Юла – Уокера
[ 1.50854880637025, 0.669522683244447, \
0.792216428146752],[0.669522683244447, \
 1.38966220164827, 0.669522683244447],
[0.792216428146752, 0.669522683244447, \
 1.50854880637025]])
In [48]: g=B*r; g                                  # Розв’язання системи
Out[48]:                                           ….рівнянь Юла – Уокера, яка в
Matrix([                                           ….матричній формі має вигляд
[ -0.584494004568384],                             ….M g = r, та отримання матриці-
[-0.644039407506937],                              ….стовпця g параметрів
[-0.267875093393598 ]])                            ….авторегресії АР(3)
In [49]: g=g.evalf(3); g                           # Скорочення до трьох знаків
Out[49]:                                           ….після коми в числових
Matrix([                                           ….значеннях параметрів
[-0.584 ],                                         ….авторегресії
```

```
[ -0.644],
[ -0.268]])
In [50]: g1=g[0,0]; g1                          # Ідентифікація параметра g1
Out[50]:
-0.584
In [51]: g2=g[1,0]; g2                          # Ідентифікація параметра g2
Out[51]:
-0.644
In [52]: g3=g[2,0]; g3                          # Ідентифікація параметра g3
Out[52]:
-0.268
In [53]: a = symbols ('a')                      # Символізація імпульсу a
                                                ….білого шуму
In [54]: ska = symbols ('ska')                  # Символізація дисперсії ska
                                                ….білого шуму
In [55]: ska = q0-g1*q1-g2*q2-g3*q3; ska        # Обчислення дисперсії ska
Out[55]:                                        ….за алгоритмом Юла – Уокера
7.99764599609375
In [56]: skv = symbols ('skv')                  # Символізація середнього
In [57]: skv = ska**(0.5); skv                  ….квадратичного відхилення
Out[57]:                                        ….skv білого шуму та його
2.82801096109859                                ….обчислення
In [58]: skv = skv.evalf(3); skv                # Скорочення до трьох знаків
Out[58]:                                        ….після коми в числовому
2.83                                            ….значенні skv
In [59]: a11,a22 = symbols ('a11 a22')          # Оголошення символьними
                                                ….меж діапазону генерації
                                                ….імпульсів білого шуму
In [60]: a11 = -2*skv; a11                      # Визначення нижньої a11 та
Out[60]:                                        ….верхньої a22 меж
-5.66                                           ….діапазону генерації імпульсів
In [61]: a22 = 2*skv; a22                       ….білого шуму за умови, що
Out[61]:                                        ….білий шум розподілений за
5.66                                            ….нормальним законом з
                                                ….довірчою ймовірністю 0,95
In [62]: import random as rnd                   # Виклик модуля генерації
                                                ….імпульсів білого шуму
In [63]: w = symbols('w:10');w                  # Індексна символізація
Out[63]:                                        ….параметра w
(w0, w1, w2, w3, w4, w5, w6, w7, w8, w9)
In [64]: v = symbols ('v:10');v                 # Індексна символізація
Out[64]:                                        ….параметра v
(v0, v1, v2, v3, v4, v5, v6, v7, v8, v9)
In [65]: w=list(w);w                            # Створення списку для
Out[65]:                                        ….параметра w
[w0, w1, w2, w3, w4, w5, w6, w7, w8, w9]
In [66]: v=list(v);v                            # Створення списку для
```

```
Out[66]:
[v0, v1, v2, v3, v4, v5, v6, v7, v8, v9]
In [67]: d = symbols (‘d:10’); d
Out[67]:
(d0,d1,d2,d3,d4,d5,d6,d7,d8,d9)
In [68]: d = list (d); d
Out[68]:
[d0,d1,d2,d3,d4,d5,d6,d7,d8,d9]
In [69]: y = symbols ('y:10');y
Out[69]:
(y0, y1, y2, y3, y4, y5, y6, y7, y8, y9)
In [70]: y=list(y);y
Out[70]:
[y0, y1, y2, y3, y4, y5, y6, y7, y8, y9]
In [71]: L21=[ ]
In [72]: L22=[ ]
In [73]: L23=[ ]
In [74]: d[0] = rnd.uniform (-5.66,5.66); d[0]
Out[74]:
 -3.949689744213309
In [75]: w[0]=g1*L11[27]+g2*L11[26]+\
g3*L11[25]+ d[0]; w[0]
Out[75]:
-5.02122294733831
In [76]: L23.append(w[0]);L23
Out[76]:
[-5.02122294733831]
In [77]: v[0]=L1[28]+w[0];v[0]
Out[77]:
-2.02122294733831
In [78]: L22.append(v[0]);L22
Out[78]:
[-2.02122294733831]
In [79]: y[0]=L[29]+v[0];y[0]
Out[79]:
22.9787770526617
In [80]: L21.append(y[0]); L21
Out[80]:
[22.9787770526617]
In [81]: d[1] = rnd.uniform (-5.66,5.66); d[1]
Out[81]:
-0.35599763798745787
In [82]: w[1]=g1*L23[0]+g2*L11[27]+\
g3*L11[26]+ d[1]; w[1]
Out[82]:
2.57876987566682
In [83]: L23.append(w[1]);L23
```

….параметра ***v***

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

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

Індексна символізація
….параметра ***y***

Створення списку для
….параметра ***y***

Створення пустого списку ***L21***
Створення пустого списку ***L22***
Створення пустого списку ***L23***
Генерація випадкового
….імпульсу ***d[0]*** для першої
….ітерації обчислень
Використовуючи модель
…АРПКС (3, 0, 2), обчислюємо
….перший член ***w[0]*** ряду
….других різниць часового ряду ***L***
Заносимо член ***w[0]*** в
….список ***L23***

Обчислюємо перший член ***v[0]***
….ряду перших різниць
….часового ряду ***L***
Заносимо член ***v[0]*** в
….список ***L22***

Обчислюємо перший
….прогнозований член ***y[0]***
….за межами часового ряду ***L***
Заносимо член ***y[0]*** в
….список ***L21***

Генерація випадкового
….імпульсу ***d[1]*** для другої
….ітерації обчислень
Використовуючи модель
…АРПКС (3, 0, 2), обчислюємо
….другий член ***w[1]*** ряду
….других різниць часового ряду ***L***
Заносимо член ***w[1]*** в

```
Out[83]:
[-5.02122294733831, 2.57876987566682]
In [84]: v[1]=L22[0]+w[1];v[1]
Out[84]:
0.557546928328509
In [85]: L22.append(v[1]);L22
Out[85]:
[-2.02122294733831, 0.557546928328509]
In [86]: y[1]=L21[0]+v[1];y[1]
Out[86]:
23.5363239809902
In [87]: L21.append(y[1]);L21
Out[87]:
[22.9787770526617, 23.5363239809902]
In [88]: d[2] = rnd.uniform (-5.66,5.66); d[2]
Out[88]:
5.4505096830373745
In [89]: w[2]=g1*L23[1]+g2*L23[0]+\
g3*L11[27]+ d[2]; w[2]
Out[89]:
7.17717253770830
In [90]: L23.append(w[2]);L23
Out[90]:
[-5.02122294733831, 2.57876987566682,\
 7.17717253770830]
In [91]: v[2]=L22[1]+w[2];v[2]
Out[91]:
7.73471946603681
In [92]: L22.append(v[2]);L22
Out[92]:
[-2.02122294733831, 0.557546928328509, \
7.73471946603681]
In [93]: y[2]=L21[1]+v[2];y[2]
Out[93]:
31.2710434470270
In [94]: L21.append(y[2]);L21
Out[94]:
[22.9787770526617, 23.5363239809902,\
 31.2710434470270]
In [95]: d[3] = rnd.uniform (-5.66,5.66); d[3]
Out[95]:
1.5127310202924091
In [96]: w[3]=g1*L23[2]+g2*L23[1]+\
g3*L23[0]+ d[3]; w[3]
Out[96]:
-2.99786690654250
In [97]: L23.append(w[3]);L23
```

….список ***L23***

Обчислюємо другий член ***v[1]***
….ряду перших різниць
….часового ряду ***L***
Заносимо член ***v[1]*** в
….список ***L22***

Обчислюємо другий
….прогнозований член ***y[1]***
….за межами часового ряду ***L***
Заносимо член ***y[1]*** в
….список ***L21***

Генерація випадкового
….імпульсу ***d[2]*** для третьої
….ітерації обчислень
Використовуючи модель
…АРПКС (3, 0, 2), обчислюємо
….третій член ***w[2]*** ряду
….других різниць часового ряду ***L***
Заносимо член ***w[2]*** в
….список ***L23***

Обчислюємо третій член ***v[2]***
….ряду перших різниць
….часового ряду ***L***
Заносимо член ***v[2]*** в
….список ***L22***

Обчислюємо третій
….прогнозований член ***y[2]***
….за межами часового ряду ***L***
Заносимо член ***y[2]*** в
….список ***L21***

Генерація випадкового
….імпульсу ***d[3]*** для четвертої
….ітерації обчислень
Використовуючи модель
…АРПКС (3, 0, 2), обчислюємо
….четвертий член ***w[3]*** ряду
….других різниць часового ряду ***L***
Заносимо член ***w[3]*** в

```
Out[97]:                                                    ….список L23
[-5.02122294733831, 2.57876987566682, \
7.17717253770830, -2.99786690654250]
In [98]: v[3]=L22[2]+w[3];v[3]                              # Обчислюємо четвертий член
Out[98]:                                                    ….v[3] ряду перших різниць
4.73685255949431                                            ….часового ряду L
In [99]: L22.append(v[3]);L22                               # Заносимо член v[3] в
Out[99]:                                                    ….список L22
[-2.02122294733831, 0.557546928328509, \
7.73471946603681, 4.73685255949431]
In [100]: y[3]=L21[2]+v[3];y[3]                             # Обчислюємо четвертий
Out[100]:                                                   ….прогнозований член y[3]
36.0078960065213                                            ….за межами часового ряду L
In [101]: L21.append(y[3]);L21                              # Заносимо член y[3] в
Out[101]:                                                   ….список L21
[22.9787770526617, 23.5363239809902, \
31.2710434470270, 36.0078960065213]
In [102]: d[4] = rnd.uniform (-5.66,5.66); d[4]             # Генерація випадкового
Out[102]:                                                   ….імпульсу d[4] для п’ятої
 -2.938526432470273                                         ….ітерації обчислень
In [103]: w[4]=g1*L23[3]+g2*L23[2]+\                        # Використовуючи модель
g3*L23[1]+ d[4]; w[4]                                       …АРПКС (3, 0, 2), обчислюємо
Out[103]:                                                   ….п’ятий член w[4] ряду
-6.49957209318491                                           ….других різниць часового ряду L
In [104]: L23.append(w[4]);L23                              # Заносимо член w[4] в
Out[104]:                                                   ….список L23
[-5.02122294733831,
 2.57876987566682,
 7.17717253770830,
 -2.99786690654250,
 -6.49957209318491]
In [105]: v[4]=L22[3]+w[4];v[4]                             # Обчислюємо п’ятий член
Out[105]:                                                   ….v[4] ряду перших різниць
-1.76271953369060                                           ….часового ряду L
In [106]: L22.append(v[4]);L22                              # Заносимо член v[4] в
Out[106]:                                                   ….список L22
[-2.02122294733831,
 0.557546928328509,
 7.73471946603681,
 4.73685255949431,
 -1.76271953369060]
In [107]: y[4]=L21[3]+v[4];y[4]                             # Обчислюємо п’ятий
Out[107]:                                                   ….прогнозований член y[4]
34.2451764728307                                            ….за межами часового ряду L
In [108]: L21.append(y[4]);L21                              # Заносимо член y[4] в
Out[108]:                                                   ….список L21
```

```
[22.9787770526617,
 23.5363239809902,
 31.2710434470270,
 36.0078960065213,
 34.2451764728307]
In [109]: L555 = L+L21                    # Об’єднуємо в один список
                                          ….L555 списки L та L21, якими
                                          ….задаються: базовий часовий
                                          ….ряд та 5 членів,прогнозованих
                                          ….моделлю АРПКС (3, 0, 2)
In [110]: import matplotlib               # Виклик ППП matplotlib
In [111]: import matplotlib.pyplot as plt # Виклик модуля pyplot як plt
In [112]: plt.plot(L555)                  # Побудова графіка часового
Out[138]: [<matplotlib.                   ….ряду з прогнозними членами
lines.Line2D at 0x2066cb6ce50>]           ….включно
```

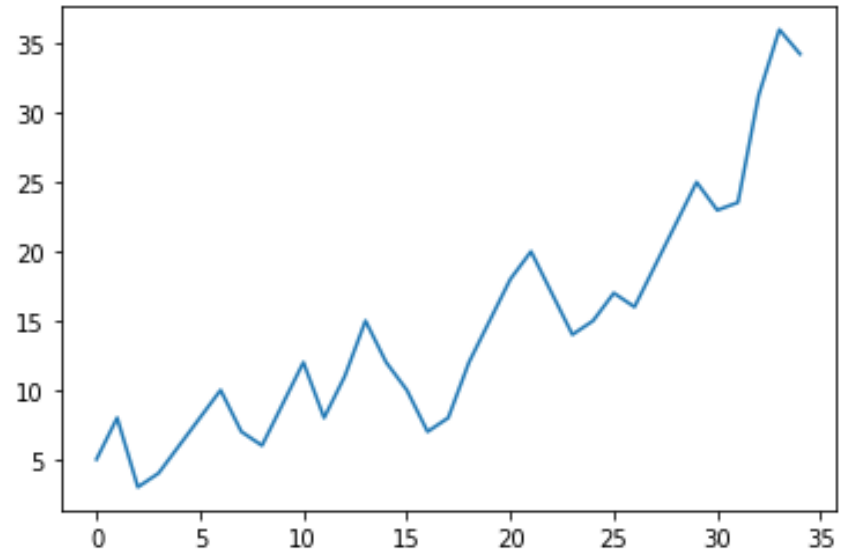


Рисунок 9.8 – Графік часового ряду y_t, який в діапазоні $t \in [0,30]$ заповнений експериментально отриманими значеннями, а за межами цього діапазону заповнений прогнозними значеннями, отриманими з використанням моделі АРПКС (3, 0, 2), ідентифікованої з використанням експериментально отриманих значень

Кінець програми 32.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Мокін Б. І., Мокін В. Б., Мокін О. Б. Навчальний посібник для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python, частина 1 : навч. пос. Вінниця : ВНТУ, 2022. 124 с.

2. Мокін Б. І., Мокін В. Б., Мокін О. Б. Функціональний аналіз, адаптований до прикладних задач в галузі інформаційних технологій : навчальний посібник.– Вінниця : ВНТУ, 2020 192 с.

3. Python. [Електронний ресурс]. Режим доступу : https://python.org/downloads/.

4. Бріггс Джейсон Р. Python для дітей (веселий вступ до програмування) / пер. з англ. Олександри Гордійчук. Львів : Видавництво старого Лева, 2019. 400 с.

5. Доля П. Г. Введение в научный Python. Харків : ХНУ ім. Каразіна, 2016. 265 с.

6. Мокін Б. І., Мокін О. Б., Шалагай Д. О. Про один із підходів наближеного обчислення інтегралів Стілтьєса і Лебега на мові Python в задачах системного аналізу з дискретними моделями. *Вісник Вінницького політехнічного інституту*, 2021, № 3 С. 61–68. DOI https://doi.org/10.31649/1997-9266-2021-156-3-61-68

7. Деч Густав. Руководство к практическому применению преобразования Лапласа и Z-преобразования. М. : Наука, 1971. 288 с.

8. Мокін О. Б., Мокін В. Б., Мокін Б. І. 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.* 2014. № 2. С. 1–6. Режим доступу : https://praci.vntu.edu.ua/index.php/praci/article/view/406/404.

9. Бокс Дж., Дженкинс Г. Анализ временных рядов. *Прогноз и управление*. Вып. 1. М. : Мир. 1974. 408 с.

10. Бокс Дж., Дженкинс Г. Анализ временных рядов. *Прогноз и управление*. Вып. 2. М. : Мир. 1974. 197 с

11 Мокін О. Б., Мокін В. Б., Мокін Б. І. Алгоритм методу ідентифікації моделі авторегресії-ковзного середнього, який узагальнює методику Юла – Уокера, та його програмна Python-реалізація. *Вісник Вінницького політехнічного інституту*. 2022. № 4. С. 61–68. DOI https://doi.org/10.31649/1997-9266-2022-163-4-41-55

Навчальне видання

Борис Іванович Мокін
Віталій Борисович Мокі
Олесандр Борисович Мокін

НАВЧАЛЬНИЙ ПОСІБНИК
для опанування студентами способів розв'язання задач з функціонального аналізу мовою Python
Частина 2

Навчальний посібник

Рукопис оформив *Б. Мокін*

Редактор *В. Дружиніна*

Оригінал-макет підготовлено у *Редакційно-видавничому відділі ВНТУ*

Підписано до друку 5.04.2023 р.
Формат 29,7×42 ¼. Папір офсетний.
Гарнітура Times New Roman.
Друк різографічний. Ум. друк. арк. 16,.17.
Наклад 30 пр. Зам. № 2023-007

Видавець та виготовлювач
Вінницький національний технічний університет,
редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: kivc.vntu@gmail.com
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.