НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК 004.021

«До захисту допущено»
Завідувач кафедри СПСКС

_____Віталій РОМАНКЕВИЧ
(підпис) (ім'я, прізвище)
"___"_____2020р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія

на тему: Прогнозування космічної погоди з використанням NARMAX моделей_____
_____

Виконав: студент ІІ курсу, групи КВ-92мп
Сапожніков Богдан Артурович _____ _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., професор Яценко В.О. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю доцент, с.н.с.,к.т.н. Юлія БОЯРІНОВА _____
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
за освітньо-професійною програмою
Спеціальність 123 Комп'ютерна інженерія (Системне програмування)

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ Віталій РОМАНКЕВИЧ
(підпис)
«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сапожнікову Богдану Артуровичу

1. Тема дисертації «Прогнозування космічної погоди з використанням NARMAX моделей»,
науковий керівник дисертації д.т.н., професор Яценко В.О.,
затверджені наказом по університету від «___» _____ 2020 р. № _____

2. Термін подання студентом дисертації 10 грудня 2020 р.

3. Об'єкт дослідження: NARMAX моделі для прогнозування космічної погоди

4. Предмет дослідження: космічна погода_____

5. Перелік завдань, які потрібно розробити: Структурно-параметрична ідентифікація NARMAX моделі. Розробка алгоритму прогнозування геомагнітного Dst індексу. Створення функціональної структури програмного забезпечення для автоматизації процесу ідентифікації та її

реалізація..

6. Перелік ілюстративного матеріалу: презентація

7. Перелік публікацій:

8. Дата видачі завдання 5 вересня 2019 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення літератури за тематикою дисертації	11.10.2019	
2.	Підготовка матеріалів першого розділу магістерської дисертації	16.09.2020	
3.	Підготовка матеріалів другого розділу магістерської дисертації	22.09.2020	
4.	Підготовка матеріалів третього розділу магістерської дисертації	26.09.2020	
5.	Підготовка матеріалів четвертого розділу магістерської дисертації	30.09.2020	
6.	Підготовка матеріалів п’ятого розділу магістерської дисертації	05.10.2020	
7.	Розробка моделі сітки Петрі	09.10.2020	
8.	Розробка моделі задачі реального часу	15.10.2020	
9.	Оформлення документації магістерської дисертації	21.11.2020	
10.	Попередній розгляд магістерської дисертації на кафедрі	26.11.2020	

Студент _____ Сапожніков Б.А.
(підпис)

Науковий керівник дисертації _____ Яценко В.О.
(підпис)

Реферат

Актуальність теми

У даний час кількість факторів, на які космічна погода має великий вплив швидко зростає. Цей ріст обумовлено, як збільшенням кількості електронного обладнання, що виводиться на навколоземну орбіту, так і зростаючою кількістю чутливого до електромагнітних коливань наземного обладнання, включаючи навіть звичайні електромережі.

На даний момент методи, що використовуються для передбачення космічної погоди мають велику кількість обмежень пов’язаних з невисокою точністю та обмеженим горизонтом прогнозу.

Мета і задачі дослідження

Структурно-параметрична ідентифікація NARMAX моделі. Розробка алгоритму прогнозування геомагнітного Dst індексу.

Створення функціональної структури програмного забезпечення для автоматизації процесу ідентифікації та її реалізація.

Об'єкт дослідження – NARMAX моделі для прогнозування космічної погоди.

Предмет дослідження – космічна погода.

Методи дослідження

Математичне моделювання, оптимізація та порівняльний аналіз.

Наукова новизна одержаних результатів

Запропоновано новий метод структурно-параметричної ідентифікації NARMAX моделей. Він дозволяє покращити точність прогнозу через зменшення ступеню поліному моделі. Також використання цього методу дозволяє збільшити горизонт прогнозу до 5-6 годин.

Розроблена NARMAX модель типу «вхід-вихід», що базується на описаному методі та включає у себе прогнозування Dst індексу з використанням даних про швидкість сонячного вітру та південної компоненти магнітного поля.

Обґрунтована структура програмного забезпечення для автоматичної структурно- параметричної ідентифікації моделі прогнозування космічної погоди на основі викладених методів. За цією структурою було розроблено програмне забезпечення на основі Python з використанням бібліотек mathplotlib та numpy.

Практична цінність одержаних результатів полягає в тому, що результуюча NARMAX модель дозволяє отримувати більш точний прогноз на більшому відрізку часу.

Апробація результатів дисертації.

Основні положення і результати роботи представлені та обговорені на:

- XIII конференція молодих вчених «ПМК-2020» у «Прикладна математика та комп’ютинг»;

Публікації.

За темою досліджень опубліковано 1 наукову працю - тези доповіді на конференції. Планується прийняти участь в міжнародній конференції

Структура та обсяг дисертації.

Магістерська дисертація складається з чотирьох розділів.

Для вирішення поставленої задачі у першому розділі атестаційної роботи викладено сфери застосування результатів передбачення космічної погоди. Проведено аналіз сучасних способів ідентифікації моделей та існуючих способів прогнозування. Окреслено принципи роботи досліджуваних методів прогнозування та їх ефективність.

У другому розділі роботи визначено основні математичні та алгоритмічні засоби, на яких базується викладений метод структурно-параметричної ідентифікації, визначено існуючі джерела експериментальних даних.

У третьому розділі обґрунтовуються засоби розробки та розглядаються особливості їх використання. Також описується функціональна структура розробленого програмного забезпечення.

Описана логіка його взаємодії з джерелами експериментальних даних та його інтерфейс для отримання результатів роботи.

У четвертому розділі подані результуючі NARMAX моделі та порівняльні результати їх роботи. Подане порівняння по ефективності прогнозування з стандартними методами ідентифікації моделі.

Ключові слова.

Прогнозування, космічна погода, NARMAX модель, принцип чорного ящика, горизонт прогнозу, структурно-параметрична ідентифікація.

Abstract

Actuality of theme

Currently, the number of factors that are greatly influenced by space weather is growing rapidly. This growth is due to both the increase in the number of electronic equipment launched into orbit and the growing number of sensitive to electromagnetic oscillations ground equipment, including even conventional power grids.

Currently, the methods used to predict space weather have a number of limitations due to low accuracy and limited forecast horizon.

The purpose and objectives of the study

Structural and parametric identification of the NARMAX model. Development of a geomagnetic Dst index prediction algorithm.

Creating a functional structure of software to automate the identification process and its implementation.

The object of study - NARMAX models for forecasting space weather.

The subject of research - space weather.

Research methods

Mathematical modeling, optimization and comparative analysis.

Scientific novelty of the obtained results

A new method of structural-parametric identification of NARMAX models is proposed. It improves the prediction accuracy by reducing the degree of the polynomial model. Also, the use of this method allows you to increase the forecast horizon to 5-6 hours.

An NARMAX input-output model has been developed based on the described method and includes Dst index prediction using data on the solar wind speed and the southern component of the magnetic field.

The structure of the software for automatic structural-parametric identification of the model of space weather forecasting on the basis of the stated

methods is substantiated. Python-based software using the mathplotlib and numpy libraries was developed using this structure.

The practical value of the obtained results is that the resulting NARMAX model allows to obtain a more accurate forecast over a longer period of time.

Approbation of dissertation results.

The main provisions and results of the work are presented and discussed at:

- XIII Conference of Young Scientists "PMK-2020" in "Applied Mathematics and Computing";

Publications.

1 scientific work was published on the topic of research - abstracts of the report at the conference. It is planned to take part in the international conference

The structure and scope of the dissertation.

The master's dissertation consists of four sections.

To solve this problem in the first section of the certification work outlines the scope of application of the results of space weather forecasting. The analysis of modern methods of model identification and existing methods of forecasting is carried out. The principles of operation of the studied forecasting methods and their efficiency are outlined.

In the second section of the work the basic mathematical and algorithmic means on which the stated method of structural-parametric identification is based are defined, the existing sources of experimental data are defined.

The third section substantiates the means of development and considers the features of their use. The functional structure of the developed software is also described. The logic of its interaction with experimental data sources and its interface for obtaining work results are described.

The fourth section presents the resulting NARMAX models and comparative results of their work. A comparison of forecasting efficiency with standard model identification methods is presented.

Keywords.

Forecasting, space weather, NARMAX model, black box principle, forecast horizon, structural-parametric identification.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ І ТЕРМІНІВ …… 12
ВСТУП…… 13
1 ОСОБЛИВОСТІ ПРОГНОЗУВАННЯ КОСМІЧНОЇ ПОГОДИ…… 14
1.1 Космічна погода…….. 14
1.2 Сфери впливу космічної погоди та актуальність…… 14
1.3 Характеристики моделі прогнозування подій …… 18
1.4 Основні показники космічної погоди ……..….. 19
1.5 Вимірювання електромагнітного потоку……. 22
1.6 Геомагнітне поле…… 23
1.7 Кліматологічні моделі прогнозування……..…. 25
1.8 Моделі типу вхід-вихід…… 25
1.9 Модель NARMAX…… 27
1.10 Вибір структури моделі…….. 29
Висновки до розділу…… 32
2 ЗАДАЧА ІДЕНТИФІКАЦІЇ NARMAX МОДЕЛІ…… 33
2.1 Задача вибору показників прогнозування…… 33
2.2 Моделі з кількома входами та ранжирування вхідних ……. даних ……..... 36
2.3 Зворотній зв'язок та нелінійність. …… 38
2.4 Моделювання підсистем ……. 39
2.5 Вибір джерел вхідних даних та їх корекція ……..…… 40
2.6 Фільтрація Калмана……..... 43
2.7 Іоносферна асиміляція даних …… 30
2.8 Перевірка моделі…… 45
Висновки до розділу…… 47
3 ІМПЛЕМЕНТАЦІЯ ЗАПРОПОНОВАНОГО СПОСОБУ 45
ІДЕНТИФІКАЦІЇ МОДЕЛЕЙ…… 47
3.1 Математичне моделювання…….. 41
3.2 Обґрунтування засобів розробки ПЗ…… 50
1.1.1 Порівняння середовищ розробки…… 50

1.1.2 Використання бібліотеки Matplotlib ……. 54
1.1.3 Робота з даними …… 55
3.1 Алгоритм структурно-параметричної ідентифікації моделі……. 55
3.2 Функціональна структура ПЗ…… 57
Висновки до розділу…… 81

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ 81
4.1 Порівняння результатів прогнозу з експериментальними……. даними……. 81
4.2 Порівняння результатів прогнозування різних методів структурно параметричної ідентифікації……. 89
Висновки до розділу…… 91
ВИСНОВКИ…… 94
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ……... 94
ДОДАТКИ …… 96

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

ПА – програмний алгоритм.

ПЗ– програмне забезпечення.

ОС – операційна система.

NARMAX – “The Nonlinear Autoregressive Moving Average”.

СД - Корональна діра - область на Сонці, де лінії магнітного поля вигинаються від Сонця. Це дозволяє сонячному вітру втекти з високою швидкістю і спричинити темні діри в короні, як це видно на сонячних знімках.

CIR – «Co-rotating Interaction Region»Термін дії області взаємодії, що обертається, для стиснутої межі між швидким і повільним сонячним вітром у високошвидкісному потоці сонячного вітру із стійких корональних дір під час численних обертань сонця.

Dst – «Disturbance storm time index». Позначає силу кільцевого струму навколо Землі, спричиненого сонячними протонами та електронами. Негативне значення Dst означає, що магнітне поле Землі ослаблене. Значення -50 нТл і нижче, як правило, розглядаються як умови геомагнітної шторму.

HSS - «High Speed Stream» Високошвидкісний потік HSS Потік сонячного вітру на швидкості вище середньої. Часто використовується для визначення сонячного вітру, що виходить із коронкової діри.

IMF – «Interplanetary Magnetic Field» Сонячне магнітне поле, яке несе сонячний вітер серед планет Сонячної системи.

IRIS – «Interface Region Imaging Spectrograph» Космічний апарат NASA для сонячного спостереження.

DRAO – “The Dominion Radio Astrophysical Observatory”

Вступ

Прогнозування космічних погодних явищ представляє головний виклик для моделі космічної фізики. Повинні бути задоволені не тільки фізичні обмеження, але й відповідні практичні питання, такі як своєчасність, точність та надійність. Для задоволення цих потреб сучасні синоптики та користувачі космічної погоди покладаються на велику різноманітність моделей космічної погоди, починаючи від простих регресій і закінчуючи досить складними інформаційними (емпіричними), фізичними та гібридними моделями. Як результат, прогнозування космічного середовища на основі моделей стабільно покращується. Особливо з початку 1990-х років, коли дані в режимі реального часу стали доступними в режимі он-лайн, залежать від часу вхідні дані, а пізніше асиміляція даних та відповідні методи, такі як фільтрація Калмана, значно покращили точність прогнозування. Валідація та верифікація моделі є важливими етапами оцінки повного спектру можливостей моделі космічної погоди. Я обговорюю вищезазначені концепції та ілюструю їх там, де це необхідно, використовуючи тематичні дослідження. На закінчення я підсумовую відповідні останні події та перспективи майбутнього.

Розділ 1 «ОСОБЛИВОСТІ ПРОГНОЗУВАННЯ КОСМІЧНОЇ ПОГОДИ»

1.1 Космічна погода

Вперше офіційно визначення поняття «космічна погода» дано в США в 1995 р при розробці національної програми NSWP - National Space Weather Program.

"Космічна погода" - це зміни умов на сонці, в сонячному вітрі, магнітосфері і іоносфері, які можуть вплинути на роботу і надійність бортових і наземних технологічних систем і загрожувати здоров'ю і життю людей.

В даний час сонячною активністю (СА) прийнято називати сукупність активних утворень (плям, протуберанців і т.п.), а також нестаціонарних динамічних явищ (спалахів, спливаючих магнітних потоків), що спостерігаються в атмосфері Сонця. СА характеризується, зокрема числом Вольфа, в основі якого лежить число сонячних плям і груп плям на видимій півсфері Сонця. З урахуванням чергування магнітної полярності сонячних плям фізично більш обґрунтований 22-річний цикл СА [2]. Також космічна погода включає велику кількість складних електродинамічних та плазмових фізичних явищ та їх вплив на техногенні технології та інфраструктуру.

1.2 Сфери впливу космічної погоди та актуальність

Існує багато зон у яких визначається вплив космічної погоди (рис 1.1), такі як: сонячна, іоносферна, геліосферна, магнітосферна та нейтральна. Їх можна поділити за важливістю для людей по кількості ресурсів, що знаходяться у цих зонах. Очевидно, що найбільша кількість та різноманітність технологічних активів, міститься у магнитосфері землі. Там знаходяться космічні станції, супутники та комічні шатли. Все це

обладнання характеризується як дуже цінне. Тому дуже важливо правильно оцінювати вплив на нього космічної погоди. Також велика частина обладнання, що знаходиться на Землі теж знаходиться під впливом різноманітних проявів космічної погоди. Це можуть буду електромережі мережі зв’язку тощо. Частина обладнання знаходиться у позаземному просторі: міжпланетні кораблі, роботизована техніка дослідження місяця тощо.

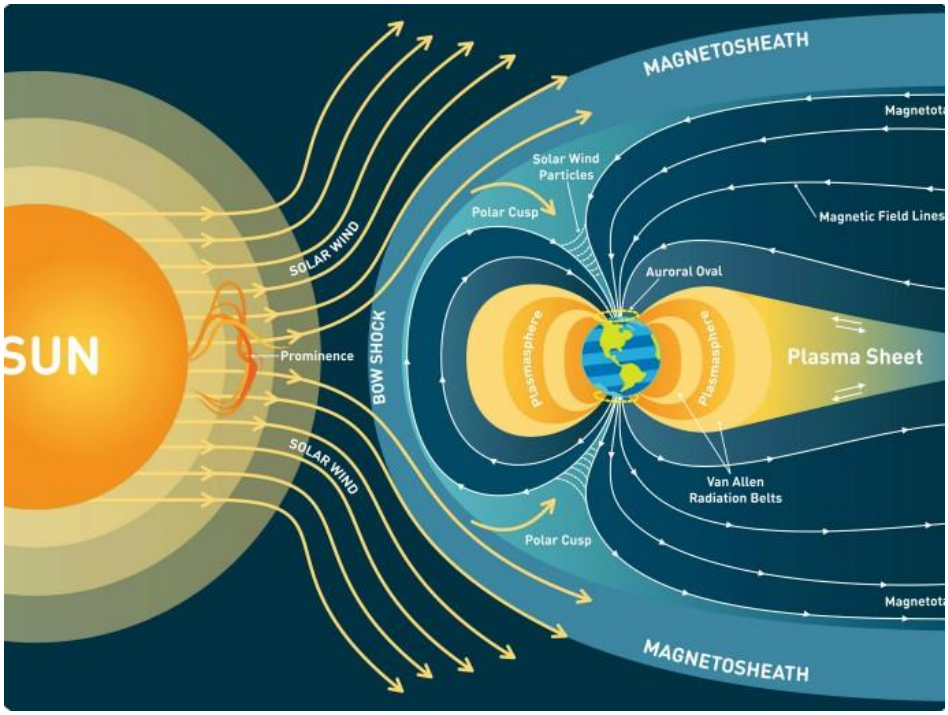

Рисунок 1.1 Загальний вигляд сфер, що мають відношення до поняття космічна погода.

Космічні дослідження надають переваги суспільству на Землі в таких сферах, як охорона здоров'я, транспорт, громадська безпека, промислове виробництво, енергетика та навколишнє середовище та інформаційні технології. Більш широкий перелік технологічних проривів включає покращені сонячні батареї, імплантовані серцеві монітори, світлодіодну протиракову терапію, бездротові інструменти, створення високотемпературних сплавів, що використовуються в турбінах

струминних двигунів, камери, що знаходяться в сучасних стільникових телефонах, компактні системи очищення води, глобальні пошуково-рятувальні системи та біомедичні технології.

Розмір космічної індустрії в 2016 склав 345 млрд дол. США. Доходи в цій сфері зросли більш ніж удвічі за останнє десятиліття, хоча в останні роки зростання уповільнилося. Доходи від супутникового зв'язку зосереджуються на двох споживчих ринках: супутникового телебачення та продуктів та послуг супутникової системи навігації. Загальний ринок виробництва супутників, запусків, що їх доставляють до космосу, а також відповідне обладнання та послуги збільшився з 2004 року більш в 4 рази.

Покращені прогнози космічної погоди захищають життєво важливі для людства інфраструктури. Економіка багатьох країн базується на системах зв'язку, розташованих у космосі та на землі, на які можуть негативно впливати примхи космічної погоди. Прогнозування цих подій є важливим для ефективного пом'якшення їх негативних наслідків.

Космічна погода має сильний вплив на рівень геомагнітної активності. Для її прогнозування розробляють нові моделі. Для її прогнозування використовують різні підходи, засновані на нейронних мережах та NARMAX моделях.

Розробка нових моделей прогнозування ведеться по всьому світу, не тільки за рахунок державних фондів, а й використовуючи фінансові фонди підприємств. Точність і тривалість прогнозів має дуже великий вплив на діяльність багатьох підприємств. Оскільки космічна погода може серйозно порушити таку діяльність, як супутникові операції, зв'язок та авіація. Інвестиції у розробку нових методів прогнозування космічної погоди також має цінність для енергетичних компаній, які попередньо отримуватимуть попередження про геомагнітні бурі, які можуть пошкодити їх електромережі. Усі ці сектори можуть отримати вигоду від своєчасного попередження про потенційні ризики, пов'язані з космічною

погодою, та вжити заходів для пом'якшення наслідків для технологічної інфраструктури. [1]

Отже, по-перше, космічна погода представляє собою діяльність певного регіону геопростору, а значить прогнозування космічної погоди може вестись для різних областей сонячної, геліосферної / міжпланетної, магнітосферної, іоносферної та нейтральної атмосфери. Ймовірно, найбільша кількість та різноманітність технологічних активів, на які впливає космічна погода, знаходиться в магнітосфері Землі (супутники, транспортні засоби, що повертаються, і космічні станції). Ці активи також зазвичай несуть дуже високі витрати, при виході з ладу. Менша кількість активів знаходиться на поверхні Землі (електромережі, трубопровідні системи, мережі зв'язку тощо) та у позаземному просторі (міжпланетні космічні кораблі та бази дослідження Місяця).

По-друге, в межах своєї області модель космічного середовища представляє один або кілька фізичних ефектів, які, як правило, є електродинамічними та плазмовими фізичними явищами. Наприклад, модель космічного випромінювання для поясів Ван Аллена повинна включати кілька фізично різних компонентів випромінювання, таких як космічні промені, енергетичні частинки сонячних променів, електрони радіаційного поясу та види іонів тощо.

По-третє, різні технологічні засоби в різному ступені вразливі до того ж самого фізичного ефекту. Наприклад, електромережі та трубопроводи по-різному реагують на наземне геоелектричне поле.

Кожне поєднання космічного погодного середовища, фізичного ефекту та технологічних засобів вимагає окрему модель прогнозування і представляє цілий спектр викликів та можливостей з точки зору збору даних, моделі та розробки теорії, моніторинг у реальному часі, а також перевірки результатів. Як наслідок, на сьогоднішній день моделі

прогнозування космічної погоди знаходяться на дуже різних стадіях еволюції.

Взаємодія космічних погодних систем може бути ефективно описана двома різними способами. В основному це можна розглядати як обмін інформацією. Наприклад, зміни полярності Bz міжпланетного магнітного поля (МВФ) можуть не мати суттєвих змін у внутрішньому енергетичному балансі сонячного вітру, наприклад, вони можуть відображати проходження конвектованої структури або наявність турбулентності.

Однак, у контексті взаємодії з магнітосферою Землі поворот МВФ на південь у точці вібрації L1 означає, що магнітне повторне підключення почнеться в прибережній екваторіальній області через 30-60 хвилин. Точна кількість енергії, переданої під час денного повторного підключення, не відома з такою високою точністю, як амплітуда результуючого іоносферного або магнітосферного струмів. Таким чином, амплітуда Bz IMF є корисною вхідною змінною для магнітосферної моделі, яка не враховує передачу енергії. Представлення взаємодій космічного середовища з точки зору динаміки інформації призводить до розробки інформаційних, або емпіричних, моделей. Інформаційні моделі, як правило, перші, що розробляються для даного космічного погодного середовища. Прості моделі - це нелінійні регресії та загальні середні значення поля чи іншої змінної, обумовлені місцем вимірювання та / або рівнем активності. Моделі включають напівемпіричні зображення розширення сонячного вітру магнітосферного магнітного поля а також високоширотного магнітосферного електричного поля.

1.3 Характеристики моделі прогнозування подій.

Результатом роботи моделі, враховуючи показники, що обумовлюють стан космічної погоди, є прогнозування (точне чи ні) для

конкретно вибраного показника. Модельні прогнози можна отримати як для майбутніх, так і для минулих подій.

Основними властивостями прогнозу є:

а) Своєчасність: прогноз повинен бути зроблений якомога раніше заздалегідь, щоб забезпечити достатньо часу для внесення користувачами оперативних змін. Різниця в часі між видачею прогнозу та фактичною подією називається часом виконання або фронтом прогнозування.

б) Точність: це узгодження між властивостями прогнозу та властивостями фактичної події. Залежно від застосування космічної погоди, відповідне властивість може бути відтворено як амплітуду, спектр, розподіл ймовірностей тощо.

с) Надійність: прогноз повинен досягати користувачів у будь-який час за будь-яких умов, тобто система прогнозу повинна мати зрозумілий та доступний інтерфейс для користування. Тут система прогнозування означає не лише модель космічного середовища, а й будь-який збір, обробку даних та зв’язок як апаратного, так і програмного забезпечення.[3]

1.4 Основні показники космічної погоди

Наступні сонячні явища є найважливішими джерелами мінливості космічної погоди, і тому їх моніторинг та прогнозування є важливою частиною оперативних служб ПЗ.

Сонячні плями, темні плями (при спостереженні в білому світлі) з концентрацією магнітного поля з нижчою температурою, ніж навколишня фотосфера, та розміром, як правило, від декількох до 20 мм на фотосфері. Кількість сонячних плям служило показником сонячної активності протягом багатьох століть, демонструючи циклічні коливання з періодом приблизно 11 років. Групи сонячних плям пов'язані з активними регіонами,

які найкраще спостерігаються під світлом, випромінюваним вище розташованою хромосферою та короною. Активні області - це місця, де сонячне магнітне поле, як правило, має складну структуру, що створює велику ймовірність виникнення вивержувальних подій (спалахів, викидів корональної маси).

Дефекти корони - темні ділянки в короні розміром до половини сонячного радіуса, пов'язані з відкритими лініями магнітного поля. Поширюючись у зовнішню корону по лініях магнітного поля, коронкові діри є джерелами високошвидкісних потоків на сонячному вітрі, які, в свою чергу, створюють пов'язані з цим космічні порушення погоди (геомагнітні шторми, прискорені частинки). Коронкові діри спільно обертаються із сонячною поверхнею, часто демонструючи ~ 27-денний рецидив.

Сонячні нитки та протуберанці, витягнуті з хромосфери у зовнішню корону, є одним із можливих джерел викидів корональної маси (CME), сонячного явища, що відповідає за найпотужніші геомагнітні та іоносферні бурі, впливаючи на численні активи у космосі та на землі .

Спалахи - це найшвидші вивержувальні сонячні явища, які виділяють енергію у багатьох частинах електромагнітного спектра. Електромагнітні випромінювання від спалахів є першими ознаками цих порушень, пов'язаних з космічною погодою. Рентгенівські випромінювання та випромінювання EUV порушують іоносферу Землі, тоді як раптові випромінювання сонячного радіо також можуть впливати безпосередньо на радіотехнологію. Досить часто великі спалахи асоціюються (розташовуються одночасно) з КМЕ та енергетичними частинками.

Сонячні енергетичні частинки (SEP) - це заряджені частинки (електрони, протони) сонячного походження, які часто ініціюються після (і, можливо, внаслідок) великого спалаху та / або прискорюються ударами,

що виникають під час поширення CME через сонячну корону та сонячний вітер. SEP впливають на іоносферу і можуть збільшити дозу випромінювання в навколоземному просторі та на багатьох висотах.

Сонячне магнітне поле визначає динаміку сонячних явищ і демонструє дуже складну поведінку. Магнітно-складні сонячно активні області є джерелами великих вивержувальних подій, таких як спалахи, CME та SEP. Дані про сонячні магнітні структури використовуються в службах реального часу для чисельного моделювання космічної погоди та для прогнозування поширення CME.

Як правило, спостереження за Сонцем включають зображення та магнітограми сонячного диска, зображення корони та геліосфери, а також інтегрований у диск потік, який спостерігається на певних довжинах хвиль, інтегрований в широких смугах довжини хвилі або в повному сонячному спектрі.

Сонячні зображення використовуються для визначення числа сонячних плям та груп сонячних плям; визначити місце розташування та властивості активних областей, спалахів, ниток розжарення, виступів та корональних дір, щоб оцінити їх потенційну геоефективність для прогнозування та оповіщення. Вони використовуються для характеристики магнітного поля у фотосфері та хромосфері, що, в свою чергу, використовується як вхідний сигнал у різні моделі прогнозування.

Загальні спостереження за рентгенівським та радіопотоками на різних частотах використовуються для моніторингу еволюції часу та класифікації найбільш короткочасних сонячних характеристик, таких як сонячні спалахи (потік рентгенівських променів) та радіовибухи сонця. Динамічні радіоспектри використовуються для подальшої характеристики спалахів, КМЕ та пов'язаних з ними ударних хвиль - через пов'язані сигнали радіовипромінювання типів II, III, IV. Сонячний потік на довжині хвилі 10,7 см також реєструється щодня і використовується для тривалих

кліматологічних досліджень та як проксі для інших сонячних параметрів, що використовуються в моделях атмосфери Землі. Крім того, слід зазначити, що вимірювання загальної сонячної освітленості (TSI), що широко використовуються в атмосферних програмах, також можуть бути використані для відстеження короткочасних і довгострокових змін Сонця в рамках досліджень космічної погоди.

1.5 Вимірювання електромагнітного потоку

Ці спостереження в основному використовуються для сповіщення про небезпечні умови, які можуть різко початися, такі як сонячні спалахи. вимірювання від 0,5 до 3 секунд із затримкою доступності від 1 до 5 хвилин необхідна для вимірювання потоку рентгенівських променів як засіб для сповіщення про пов'язаний з цим вплив на іоносферу.

Космічні потоки вимірювання потоку EUV та рентгенівського потоку доступні з експлуатаційних супутників NOAA / GOES. Альтернативні вимірювання в сусідніх смугах частот також проводяться науковими місіями, такими як Сонячна динамічна обсерваторія (SDO) або PROBA2 / LYRA, хоча вони не відповідають тим самим вимогам щодо своєчасності.

Для вимірювань сонячного електромагнітного випромінювання своєчасне оповіщення очевидно неможливе, але вимірювання в радіодомені на вибраних частотах (до 3 ГГц, як використовують конкретні програми) можуть бути корисними для підтвердження сонячного походження відмови або погіршення таких конкретних застосувань. Ці продукти даних, як правило, надаються як побічні продукти наземними обсерваторіями, що забезпечують динамічні радіоспектри (див. Нижче).

Для моніторингу довготривалої сонячної мінливості та введення в числові моделі космічного середовища та атмосфери використовуються вимірювання потоку на частоті 2800 МГц (10,7 см). Наразі їх надає лише

радіотелескоп Пентиктон. Слід забезпечити довготривалу безперервність та послідовність цих рядів даних.

Обговорювались альтернативи на інших частотах (наприклад, f30 ~ 1 ГГц), але вони не є рішенням проблеми довготривалої послідовності.

Радіовипромінювання, що вказують на появу радіовибухів, а також на швидкість ударів у короні та сонячному вітрі, вимагають спостереження за динамічними радіоспектрами (кольорові діаграми інтенсивності в залежності від часу та довжини хвилі / частоти) між 20 МГц та 2 ГГц з каденцією від 1 до 60 секунд та затримкою доступності на 1 та 5 хвилин.

Такі вимірювання, отримані за допомогою наземної інфраструктури, вимагають внеску обсерваторій по всьому світу, щоб досягти охоплення протягом доби. Мережі, що збирають такі дані з усього світу, існують, але в даний час не забезпечують загальнодоступності даних, що відповідають вищезазначеним критеріям. Мережа радіосонячних телескопів (RSTN), що експлуатується ВПС США, охоплює земну кулю в режимі реального часу, але не всі спектри реального часу доступні для всіх. Дані мережі eCallisto є загальнодоступними, але лише деякі станції надають інформацію в режимі реального часу.

Надання даних про сонячний потік EUV, потік рентгенівських променів та радіовипромінювання слід оцінювати як граничні до прийнятних.

1.6 Геомагнітне поле

Геомагнітне поле на поверхні Землі - це поєднання внутрішнього магнітного поля Землі (основного поля) та збурень, спричинених різними зовнішніми джерелами, тобто пов’язане з космічною погодою. Геомагнітними варіаціями, пов’язаними з космічною погодою, є: добові (24 год) варіації, обумовлені впливом сонячного електромагнітного випромінювання на іоносферу та течіями, спричиненими припливно-

тепловим припливом (переважно Е-область) та термосферними вітрами; і короткочасні варіації внаслідок порушень сонячного та магнітосферного походження, таких як геомагнітні бурі та суббурі, геомагнітні пульсації (наслідки збурень сонячного вітру та хвиль у магнітосфері), раптові імпульси (внаслідок взаємодії ударів у сонячний вітер і магнітосфера) та інші.

Регулярні вимірювання просторово-часового розподілу магнітного поля на поверхні Землі використовуються при розробці глобальної моделі Міжнародного геомагнітного опорного поля (IGRF) основного магнітного поля Землі, що модернізується кожні 5 років, що підтримує, наприклад , довгострокове виготовлення карт схилення або дослідження внутрішніх просторів Землі. З додаванням супутникових магнітних вимірювань модель IGRF використовується серед інших при розробці моделі магнітосферного магнітного поля та систем координат для оцінки супутникового середовища. Результати аеромагнітних досліджень щодо змін місцевого внутрішнього геомагнітного поля використовуються для гірничих робіт та інших геологічних застосувань.

Вимірювання короткочасних змін геомагнітного поля широко використовуються для підтримки служб космічної погоди для моніторингу та прогнозу геомагнітних збурень. Одним з найбільш традиційних видів використання є виготовлення декількох типів геомагнітних показників, таких як Kp, Dst, AE розроблений у з метою оцінки та прогнозування просторової та часової еволюції геомагнітних збурень у глобальному та локальному масштабах. Потреба у даних з високою роздільною здатністю в реальному часі та історичних даних зросла за останні кілька років для підтримки моніторингу в реальному часі чисельних моделювань та прогнозів, а також довгострокових оцінок впливу космічної погоди на критичні наземні інфраструктури, такі як ліній електропередач,

трубопроводів, а також для моніторингу вказівних помилок при спрямованому бурінні при розвідці нафти чи газу та інших послугах.

1.7 Кліматологічні моделі прогнозування

Деякі з найбільш ефективних моделей отримують шляхом простого усереднення репрезентативної кількості відповідних попередніх вимірювань активності системи. Приклади включають середню зміну сонячної плями протягом сонячного циклу, отриману в результаті вимірювань протягом кількох останніх циклів; або відповідне варіювання геомагнітного індексу Ap за той самий часовий період.

Такі моделі історично середньої активності називаються кліматологічними. Кліматологічна модель для x (t) має вигляд:

$$x(t) = a \cdot u(t) + b \tag{1.1}$$

де $u(t) = A\sin(\omega t)$ є періодичним зовнішнім фактором. Більш досконалі моделі додають динамічні терміни до базової лінії, яку надає кліматологія. Найпростішим динамічним терміном є термін стійкості, що представляє наслідки інерції системи. Додавання терміну стійкості до моделі (1.1) призводить до:

$$x_{t+\Delta t} = (1-\varepsilon)\, x_t + \varepsilon\left(a u_{t+\Delta t} + b\right) \tag{1.2}$$

де значення ε вибирається для збалансування стійкості моделі, і може бути пов’язаними з кроком у часі, або роздільною здатністю, Δt. Для малих Δt та ε = Δt рівняння (1.2) значення дискретного часу наближене до диференціального рівняння для x (t).

1.8 Моделі типу вхід-вихід.

Ключовим моментом прогнозування є те, що системи космічної погоди відкриті, тобто вони обмінюються масою, імпульсом та енергією з однією або кількома сусідніми плазмами. Тому плазмофізичні «коробкові

моделі» повинні включати вхідні дані, що залежать від часу, такі як електричні та магнітні поля, інжектовані пучки та / або припливні плазми, для того, щоб можна використовувати їх для прогнозування космічної погоди. На відміну від багатьох інших геофізичних систем, коливання рушійних змінних не є малими порівняно із їх середніми значеннями, але, залежно від космічного середовища, вони часто можуть значно їх перевершити. Реакція та стабільність космічного середовища на ці зміни вимагає зовсім іншого типу моделювання, ніж модель закритих систем або систем з постійним введенням.

Властивість відкритої системи означає, що модель космічної погоди повинна відстежувати історію часу своїх вхідних змінних. Основний напрямок передачі інформації - геліоцентричний, тобто від Сонця до краю геліосфери. Проте існує багато різних шляхів передачі цих величин, тому швидкість розповсюдження інформації сильно відрізняється в різних середовищах і навіть в одному середовищі. Наприклад, у міжпланетному середовищі існують порядки відмінностей у швидкості розповсюдження між електромагнітним випромінюванням, енергетичними частинками, хвилями та потоками.

Таким чином, комплексна модель може потреати відстеження дуже великої кількості параметрів за великий проміжок часу. Вимірювання, проведені поблизу сонячного або міжпланетного джерела космічних погодних подій, мають вищу значимість, ніж ті, що були зроблені ближче до Землі, оскільки перші дозволяють передбачити більший час виконання. З іншого боку, інформація про первинні сонячні або міжпланетні збурення стає спотвореною і з часом ненадійною, коли вони подорожують по різних середовищах. Ефективне спотворення можна виміряти за допомогою функції кореляції або структури.[3]

Модель NARMAX

Промислові програми представляють складні проблеми, з якими стикаються, коли динамічні моделі потрібні для управління нелінійними системами. У модельному керуванні вхід-вихід нелінійні моделі можуть бути розроблені на основі фізичних принципів або отримані в рамках процедури ідентифікації системи.

Перший підхід є найбільш адекватним, але на практиці він часто включає деякі основні проблеми:

• важко встановити правильні значення для фізичних параметрів, щоб отримати відповідну модель для конкретного застосування;

• ідентифікація фізичних параметрів за даними не є тривіальною через структуру нелінійних рівнянь;

• моделі, засновані на теоретичних засадах, можуть бути дуже складними, і їх використання для цілей контролю не є простим.

Альтернативним рішенням, як у лінійному випадку, є використання алгоритмів ідентифікації системи: вхід-вихід нелінійні моделі, які не обов'язково мають фізичний аналог, ідентифікуються з даних, щоб отримати модель управління. Для нелінійної ідентифікації системи доступно кілька класів нелінійних моделей.

Перша класифікація може бути зроблена з урахуванням попередніх знань: «моделі чорних ящиків» зазвичай визначаються як ті моделі, структура яких вибирається без фізичного розуміння системи. Ці моделі можна розглядати як нелінійні зіставлення зі спостережуваних даних у вихідний простір (пряме відображення або об'єднання відображень)[5].

Поліноміальне представлення моделі NARMAX - це набір нелінійних моделей чорного ящика, який можна застосувати до широкого класу нелінійних систем і який легко інтегрувати в просту процедуру оцінки параметрів та вибору структури моделей нелінійної системи

NARMAX на основі даних, заснованих на рекурсивній оцінці параметрів та виборі структури моделі.

Нелінійна авторегресивна модель ковзаючого середнього (NARMAX) модель може представляти широкий клас нелінійних систем (Leontaritis and Billings, 1985a, b). І визначаться як:

$$\begin{aligned} y(k) = F\,(&y(k-1), y(k-2), \ldots, y(k-n_y), \\ &u(k-\tau), u(k-\tau-1), \ldots, u(k-n_u), \\ &e(k-1), e(k-2), ..., e(k-n_e)) + e(k), \end{aligned} \tag{1.3}$$

де y (k), u (k), e (k) - це вихід системи, вхід і значення квадратичної похибки відповідно, і ny, nu, - це крок у часі за який визначаються дані. F деякі нелінійні функції.

Представлення NARMAX - це добре відомий інструмент для нелінійного моделювання, який включає кілька інших нелінійних подань, таких як блокструктуровані моделі та серії Вольтерри. Цей клас моделей має привабливу особливість бути лінійними за параметрами, так що можна застосовувати пряму реалізацію методів найменших квадратів. Розширюючи F (·) у (3) як поліном ступеня L (де L - ступінь нелінійності), вираз поліноміальної моделі NARMAX отримується наступним чином

$$y(t) = \sum_{i=1}^{n} \theta_i x_i(t) + e(t) \tag{1.4}$$

$$\begin{aligned} &n = \sum_{i=0}^{L} n_i, \;\; n_0 = 1 \\ &n_i = n_{i-1} \frac{(n_y + n_u + n_e + i - 1)}{i}, i = 1 \ldots L \\ &x_1(t) = 1 \\ &x_i(t) = \prod_{j=1}^{p} y(t - n_{yj}) \prod_{k=1}^{q} u(t - n_{uk}) \prod_{m=1}^{r} e(t - n_{em}) \end{aligned} \tag{1.5}$$

Вибір поліноміального виразу для регресора базується на можливості вивести нелінійні алгоритми управління для нелінійного полінома модель як пряме продовження класичної задачі управління лінійним розміщенням полюсів.

Нехай R = {1, 2, ... , m} позначають набір регрессорів із структури NARMAX. Розмірність R, тобто число комбінацій одночленних членів NARMAX, задається `-комбінацією з повтореннями, такими як

$$m = \left(\!\!\binom{n_\theta}{\ell}\!\!\right) = \binom{n_\theta + \ell - 1}{\ell}, \tag{1.6}$$

де $n_\theta = n_y + (n_u - \tau + 1) + n_e + 1$ і l - ступінь нелінійності у випадку поліноміальних моделей. Навіть для помірних значень ny, nu, ne і l` вибір структури за допомогою грубої сили неможливий. Наприклад, враховуючи ny = nu = 4, ne = 0 (отже, модель NARX), `= 3, тоді m = 165.

Отже, кількість усіх можливих моделей, які слід оцінити, становить $2^{165} = 4{,}67 * 10^{49}$. Це наочно демонструє необхідність ефективних методів вибору структури моделі.

Вибір структури моделі.

Коротше кажучи, проблема вибору структури моделі полягає у виборі з підмножини R регресорів, що утворюють остаточну модель. Для цього широко використовуваним критерієм є коефіцієнт зменшення помилок (ERR) (Billings et al., 1989), який вимірює зменшення дисперсії залишків, яке відається, коли в модель входить новий термін, нормований щодо до дисперсії на виході. Отже, коефіцієнт квдратичної похибки, внаслідок включення в модель і-го регресора може бути записаний як:

$$[\mathrm{ERR}_1]_i = \frac{\mathrm{MS1PE}(\mathcal{M}_{i-1}) - \mathrm{MS1PE}(\mathcal{M}_i)}{\langle \mathbf{y}, \mathbf{y} \rangle}, \tag{1.7}$$

для i = 1,2,. . . , m, де MS1PE (Mi) означає середньоквадратичну похибку прогнозування на крок вперед моделі регресорами; m - помилкова кількість випроаних кандидатів; і M - сімейство моделей із вбудованими структурами, таким чином, $M_{i-1} \dashrightarrow M_i$. Знаменник (5) - дисперсія даних.

Однією з переваг ERR є те, що її можна представити у компактній формі як (Billingset al., 1989):

$$[\mathrm{ERR}_1]_i = \frac{\hat{g}_i^2 \langle \mathbf{w}_i, \mathbf{w}_i \rangle}{\langle \mathbf{y}, \mathbf{y} \rangle}, \qquad i = 1, 2, \ldots, m, \tag{1.8}$$

де w_i - і-й ортогональний регресор, а g_i - відповідний оцінюваний параметр. Таким чином, на кожному кроці до моделі додається термін із найбільшим ERR1. ERR1 є простим і швидким у використанні, а також він є досить ефективним, тому він широко використовується. Розширення критерію ERR з використанням двокрокових прогнозів ERR2 запропоновано в (Alves et al., 2012) з метою виявлення небажаних термінів.

Також існує інший критерій оцінки помилки, такзваний коефіцієнт зменшення помилок моделювання (SRR), визначений як(Піродді та Спінеллі, 2003):

$$[\mathrm{SRR}]_i = \frac{\mathrm{MSSE}(\mathcal{M}_{i-1}) - \mathrm{MSSE}(\mathcal{M}_i)}{\langle \mathbf{y}, \mathbf{y} \rangle}, \tag{1.9}$$

для i = 1,2,…m, де MSSE (Mi) позначає середньоквадратичну помилку моделювання моделі з регрессором. У (1.8) використовується моделювання вільного запуску. SRR може бути ефективним в неідеальних умовах ідентифікації і часто дає більш компактні моделі. З іншого боку, такий критерій вимагає значно більшого обчислювального елемента і важко застосувати для моделей часових рядів (моделей без вводу). Коли йдеться про вибір структури з експериментальних даних, надзвичайно важливо усвідомити, що певні "справжні" або "стрибкоподібні" можуть бути більш корисними, ніж інші. Крім того, стверджувалося, що з огляду

на набір даних з обмеженою довжиною та точністю, різні зрізи моделей можуть стати нерозбірливими (Barbosa et al., 2015).

Висновки до розділу:

У розділі розглянуто основні природні процеси, що впливають на стан космічної погоди. Також було викладено основні сфери, на які впливає стан космічної погоди на сьогоднішній день.

Проаналізовано та описано набори параметрів, що використовуються при прогнозуванні. Описано основні методі прогнозування стану космічному середовища, що застосовуються на сьогоднішній день. Проаналізовано існуючі підходи до прогнозування космічної роботи.

Обґрунтовано необхідність розробки покращеного підходу, що дозволить оптимізувати процес прогнозування космічної погоди та збільшити точність прогнозу. Тобто, за допомогою цього підходу потрібно отримати математичну модель, яка забезпечує найменшу похибку при отриманні прогнозу. Зазначено та обґрунтовано ефективність D_{st} індексу, як показника, що ефективно описує стан процесів в магнітосфері та іоносфері.

Подано необхідність розробки NARMAX моделі, в якій складність поліному правою частини диференційного рівняння буде меншою за раніше існуючі підходи. Ключовим фактором моделювання космічної погоди є той факт, що ці середовища є відкритими системами, які обмінюються масою, імпульсом та енергією зі своїм середовищем. Для точного та довгострокового прогнозу стану системи, необхідне належне представлення всіх залежних та незалежних вхідних даних.

2. ЗАДАЧА ІДЕНТИФІКАЦІЇ NARMAX МОДЕЛІ

2.1 Задача вибору показників прогнозування

У цій роботі як результуючий показник вибрано D_{st} індекс для моделювання та прогнозування середовища космічної погоди (кільцевого струму). Він вибраний, оскільки його часові зміни можуть бути добре відображені простими динамічними моделями з чіткою фізичною інтерпретацією. Індекс призначений для представлення магнітних ефектів кільцевого струму і визначається як середньозважене компонента горизонтального збурення компонента Північ-Південь, виміряне в чотирьох місцях середньої широти. Кільцевий струм є домінуючим фактором магнітної активності середньої широти (хоча і не єдиним). Кілька струмів, відмінних від кільцевого, сприяють вимірюванню індексу Dst. Ефекти найважливішого - магнітопаузного струму, який реагує на високий міжпланетний тиск, усуваються з D_{st} на стадії попередньої обробки [Burton et al., 1975].

Повороти МВФ на південь супроводжуються посиленням кільцевого струму. Основними для динаміки D_{st} є варіації випрямленої складової міжпланетарного електричного поля

$$\left(\mathbf{V}_{SW} \times \mathbf{B}\right)_y = V_{SW} B_{South} \tag{2.1}$$

де V_{SW} - радіальна складова швидкості сонячного вітру, а B_{South} - випрямленої компоненти Bz МВФ (тобто воно дорівнює 0, коли Bz позитивне, і –Bz, коли Bz від'ємне). Використовуються координати GSM. Застосовуючи до магнітохвоста, цей компонент міжпланетного поля призводить до посиленої конвекції плазмового листа до землі у внутрішню магнітосферу і, таким чином, до збільшення кількості кільцевих потоків іонів. Як вихідну точку при моделюванні Dst можна розглянути лінійну регресію між двома змінними:

$$D_{st}(t) = c + b \cdot V_{SW}(t)\, B_{South}(t) \tag{2.2}$$

де час вимірюється в годинах, а отже, варіації сонячного вітрового генератора та результуюча конвекція плазмового листа можуть розглядатися як миттєві. Збільшення VSWB на південь призводить до посилення кільцевого струму і зменшується до Dst від його рівноважного значення приблизно –50 нТл. Регресія - це проста кліматологічна модель. Статична модель не враховує зміни часу Dst під час магнітних бур. Наприклад, коли VSWBSouth переходить до нуля, | Dst | величина зменшується, приблизно у вигляді повільної експоненції, що відображає втрати іонів у кільцевому струмі (через вихід магнітопаузи, взаємодії хвилі з частинками, обмін зарядом тощо). Значною мірою цю динаміку можна вловити, збалансувавши термін міжпланетного руху з терміном втрат та додавши термін стійкості. Результатом є диференціальне рівняння першого порядку [Burton et al., 1975]:

$$\frac{dD_{st}(t)}{dt} = bV_{SW}(t)B_{South}(t) - \frac{D_{st}(t)}{\tau} \qquad (2.3)$$

де термін $bV_{SW}(t)B_{South}(t)$ являє собою швидкість введення іонів плазмового листа в кільцевий струм. Час розпаду τ представляє ефекти згаданих процесів втрат. [Бертон та ін. 1975] модель - це проста, але ефективна модель варіацій часу D_{st} з фізично зрозумілими термінами. Порівняння моделі із спостережуваним D_{st} показано на. Часовий масштаб τ моделі коливається від кількох годин до частки доби [Burton et al., 1975]. Для того, щоб виміряти ступінь успіху у відтворенні спостережуваних змін, часовий ряд індексу, передбаченого моделлю D_{st}, порівнюється із спостереженнями, D_{st}. Тут використаємо коефіцієнт кореляції:

$$C^{(\hat{D}_{st},D_{st})} = \frac{1}{T}\frac{1}{\sigma_{\hat{D}_{st}}\sigma_{D_{st}}}\int_0^T \left(\hat{D}_{st}(t) - \bar{\hat{D}}_{st}\right)\left(D_{st}(t) - \bar{D}_{st}\right)dt \qquad (2.4)$$

де $\overline{X}$ та σ_X - середнє та стандартне відхилення змінної X відповідно. Кореляція (2.3) приймає значення в межах 60-70%. Відсоток дисперсії D_{st}, що пояснюється цим типом моделі, становить приблизно квадрат кореляції або в межах 35-50%.

Маючи попередній приклад як базову модель, тепер розглянемо додаткові складності.

Одним із способів узагальнення регресії (2.4), щоб як покращити точність прогнозу, так і врахувати більше представленої фізики, є включення довшої історії недавнього введення сонячного вітру, а не одного терміна. Запишемо поточний стан системи з точки зору історії введення сонячного вітру:

$$D_{st}(t) = \int_0^{\infty} H(\tau) V_{SW}(t-\tau) B_{South}(t-\tau) d\tau \tag{2.5}$$

Зв'язок між Dst і $V_{SW}B_{south}$ представляється функцією імпульсного відгуку H (τ), яка згортається з входом. Якщо H (τ) відомий, цю так звану модель кінцевої імпульсної характеристики можна використовувати для прогнозування лінійної динаміки Dst. Основною властивістю функції імпульсного відгуку є те, що якщо вхід сонячного вітру є імпульсом (подібним до дельта-функції), то розрахункова геомагнітна амплітуда становить:

$$D_{st}(t) = H(t). \tag{2.6}$$

Функцію імпульсного відгуку можна отримати або аналітично, застосовуючи перетворення Лапласа до відомої моделі, такої як (2.6), або чисельно, шляхом прямої інверсії (2.5) [Clauer, 1986]. Зазвичай динаміка невідома, і функцію відповіді потрібно вирішувати безпосередньо з

експериментальних даних часових рядів. Для часового ряду довжиною N, рів. (2.6) записано

$$D_{st}(t) = \sum_{i=0}^{T} H(i\Delta t) V_{SW}(t - i\Delta t) B_{South}(t - i\Delta t) \tag{2.7}$$

де відстежуємо історію сонячного вітру до часу T, і Is це часовий дозвіл. Час пам'яті T відповідає найдовшій шкалі часу у $V_{SW}B_{south}$, яка може визначати Dst. Інвертування (9) як багатолінійної регресії дає

$$H = \left[\mathbf{VB}_s \mathbf{VB}_s^T\right]^{-1} \left[\mathbf{D}_{st} \mathbf{VB}_s^T\right] \tag{2.8}$$

де Dst - N-вимірний вектор стовпця (часовий ряд Dst), а $V_{SW}B_{south}$ є матрицею NxT, отриманою з відповідних вимірювань $V_{SW}B_{south}$. Тут верхній індекс T позначає транспонування матриці. Функція імпульсного відгуку Dst має пік при τ = 1 година з наступним швидким зниженням до нуля.

2.2 Моделі з кількома входами та ранжирування вхідних даних

Як вже згадувалося, моделювання вводу-виводу необхідне для моделювання геопросторової погоди, і в міру того, як моделі стають більш реалістичними, кількість вхідних змінних зростає для. Визначення найбільш релевантних вхідних змінних базується на фізиці навколишнього середовища, інформаційно-теоретичних критеріях та практичних міркуваннях (таких як доступність або надійність входів). У магнітосферних та іоносферних додатках найбільш релевантні сонячні та міжпланетні входи включають компоненти МВФ (насамперед Bz) та абсолютна величина, радіальна швидкість та щільність сонячного вітру, УФ-випромінювання та потоки частинок сонячної енергії. Залежно від конкретного застосування, порядок цих вхідних змінних може бути різним; також можуть бути необхідні інші змінні Кожен вхід представляє різні

типи зв'язку між jw.org сонячні / міжпланетні джерела енергії та геопростір тонуть. Основними взаємодіями є магнітне повторне з’єднання, в’язка взаємодія, випромінювання випромінювання тощо.

Сонячна та міжпланетарна активність сильно корелює у процесі розвитку та поширення геоефективних структур. Нещодавнє дослідження показало високий ступінь кореляції в момент швидкісних потоків на сонячному вітрі [Siscoe and McPherron]. Тому кілька змінних можна вважати взаємозалежними. Компоненти магнітного поля, потоки плазми та потоки енергійних частинок слідують характерним варіаціям часу у формуванні та розповсюдженні геоефективних структур, таких як міжпланетні поштовхи, викиди корональної маси та потоки. Це ставить питання про те, які найважливіші вхідні дані для конкретного космічного середовища.

Ми можемо оцінити геоефективність сонячного / міжпланетного входу або даного типу структури за допомогою такої метрики, як коефіцієнт кореляції (2.7) Масштаби функції геоефективності з позицією або енергією частинок можуть бути використані для ідентифікації домінуючих фізичних процесів. Недавнє дослідження порівняло геоефективність сонячних, міжпланетних та магнітосферних змінних при визначенні високоенергетичного потоку електронів у радіаційних поясах [Vassiliadis et al., 2005].

Відомо, що потік визначається багатьма різними процесами, такими як в'язка взаємодія (і збудження НВЧ-хвиль), магнітне повторне з'єднання, іоносферні ефекти тощо. Показано порівняння геоефективності таких змінних як функції L оболонка. Тут геоефективність - це коефіцієнт кореляції між передбаченнями моделі та спостереженнями електронного потоку

$$C^{(j_e,j_e)} = \frac{1}{T}\frac{1}{\sigma_{j_e}\sigma_{j_e}}\int_0^T \left(\hat{j}_e(t) - \overline{\hat{j}}_e\right)\left(j_e(t) - \overline{j}_e\right)dt$$

(2.9)

подібно до рівняння. (7). На графіку змінні згруповані разом за подібністю профілів геоефективності та вказують на три типи зчеплення: a) гідродинамічні (включаючи в'язкі) взаємодії; b) магнітне повторне підключення та пов’язана з ним геомагнітна активність, та c) посилення втрат електронів (розширення іоносфери / плазмасфери) або пом’якшувальних процесів. Зверніть увагу, що профілі відрізняються залежно від L-оболонки. Таким чином, можемо вибрати найважливіші вхідні змінні для кожного L-діапазону, що цікавить радіаційні пояси. Крім того, детальна ідентифікація структури можлива за спостережуваними часовими рядами полів, потоків або потоків. Розроблено метод накладеної епохи, який може дати середній часовий профіль швидкісного потоку за швидкістю, полем та іншими змінними на інтерфейсі потоку. Потім часові профілі використовувались як шаблони для порівняння з потоками даних та автоматичної ідентифікації інтерфейсів потоку.

2.3 Зворотній зв'язок та нелінійність.

Ці дві властивості відрізняються, але тісно пов’язані між собою і на практиці вони часто зустрічаються разом. Хоча зворотний зв'язок, як правило, пов'язаний з нестабільністю та лінійним режимом, нелінійності пов'язані з насиченням нестабільностей.

Зворотній зв'язок.

Нестабільність плазми є яскравими прикладами механізмів позитивного зворотного зв'язку. Нестабільності призводять до швидкого (експоненціального для деяких випадків) збільшення енергії системи. У моделі (2.9) вже був представлений негативний зворотній зв'язок з точки зору терміну втрати кільцевого струму. Загальною формою (2.9), що включає кілька термінів зворотного зв'язку та зв'язку, є:

$$D_{st}(t) = \sum_{i=0}^{m} a_i D_{st,i}(t - i\Delta t) + b \cdot V_{SW}(t) B_{South}(t) \qquad (2.10)$$

У плазмовій системі нелінійність можна розглядати як порушення симетрії, а саме незмінності динаміки для різних рівнів активності. Зазвичай розрив відається при екстремальних (набагато вищих або нижчих за середній) рівнях активності. Існує багато способів узагальнення лінійної системи, наприклад (6), на нелінійну. Основні три способи представлено:

Коефіцієнти зв'язку - це функції вхідного сигналу. У тих випадках, коли система керується зовнішнім джерелом, прямим способом зробити динаміку моделі нелінійною є параметризація динаміки з точки зору вхідного рівня.

Наприклад, у нелінійній версії моделі Dst (2.6) коефіцієнт втрат τ моделювався як функція входу сонячного вітру $V_{SW}B_{South}$ [O'Brien and McPherron, 2000]. Коефіцієнти b і τ отримуються шляхом пристосування до історичних даних Dst та VSWBSouth. У сучасній та гнучкій методології наближається форма нелінійності за допомогою нейронної мережі або іншого ітераційного підходу. Проста мережа, така як багатошаровий персептрон, має ієрархічну структуру введення-виведення, де виходи рівня n є входами до рівня n + 1.

Вхідні дані першого шару - це входи в систему (у цьому випадку $V_{SW}B_{South}$ (t) та їх відставання), а вихідні дані останнього шару - це вихід системи (в даному випадку геомагнітна активність у момент часу t, Dst (t) [Лундштедт, 1996, 1997].

2.4 Моделювання підсистем.

Третім способом введення нелінійних зв'язків є визначення фізичних підсистем, моделювання кожної окремо, а потім синтез окремих прогнозів для оцінки активності всієї системи.

Індекс Dst, наприклад, визначається симетричними варіаціями часу кільцевого струму, але також такими, як асиметричний кільцевий струм, а також магнітопауза та хвостові струми. У роботі [Burton 1975] модель для Dst (2.6), модель магнітної сигнатури кільцевого струму побудована з окремих підходів до даних для активних $(V_{SW} B_{South} \neq 0)$. Геомагнітний ефект струму магнітопаузи моделювався окремо. У більш комплексній обробці геомагнітні ефекти п’яти струмів, що сприяють індексу, моделювались окремо на основі руху сонячного вітру та МВФ [Temerin and Li, 2001] з використанням власного часу зростання та занепаду з моделями, що нагадують рівняння (2.11).

У цій моделі оціночна активність Dst являла собою загальну суму геомагнітних ефектів усіх цих струмів, але можна передбачити подібні моделі з сильним зв’язком між підсистемами.

2.5 Вибір джерел вхідних даних та їх корекція

Для деяких програм необхідні спостережувані та скориговані значення сонячного потоку. Загальна форма повинна показуватися варіації сонячного циклу. У даних є великі стрибки, що вказує на неправильні, багаторазові або пропущені реальних значень, хоча частота зменшувалась у попередні роки. Також існують додаткові Інтернет-джерела, де можна знайти дані спостережень та прогнозів за минулі роки.

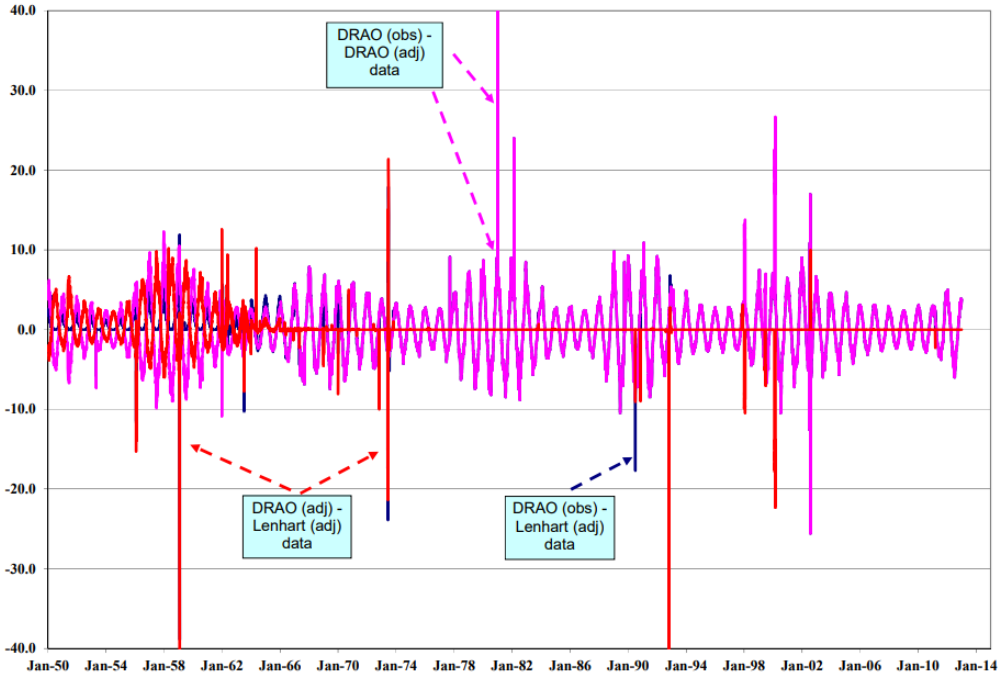

Рисунок 2.1 Порівняння спостережуваних та скоригованих сонячних потоків.

Циклічна природа стрибків обумовлена циклічністю сонячних процесів. Піки виникають, коли виправлення або значення є неправильними. У роботі розглянуто та вирішено відмінності по кожному пункту. Підхід до визначення різниці між кожним із 3 значень (з поправкою Ленхарта, з поправкою DRAO та DRAO спостереження). Помилки виявляються випадковими між усіма 3 базовими значеннями, але вдалося встановити методологію для виправлення невідповідностей, розглянувши відмінності. Необхідні дві корекції:

1. Якщо модуль (DRAO виправленого – Lenhart виправленого потоку) перевищує 0,001, то потік Lenhart змінюється, шляхом додавання його різниці та скоригованого DRAO-Lenhart.

2. Якщо модуль(DRAO змінений – DRAO спостереженого\ виправленого) > 0,5, то DRAO встновлюется на DRAO змінений + DRAO спостережений- DRAO спостереженого\виправленого.

Виправлені відмінності виглядають наступним чином.

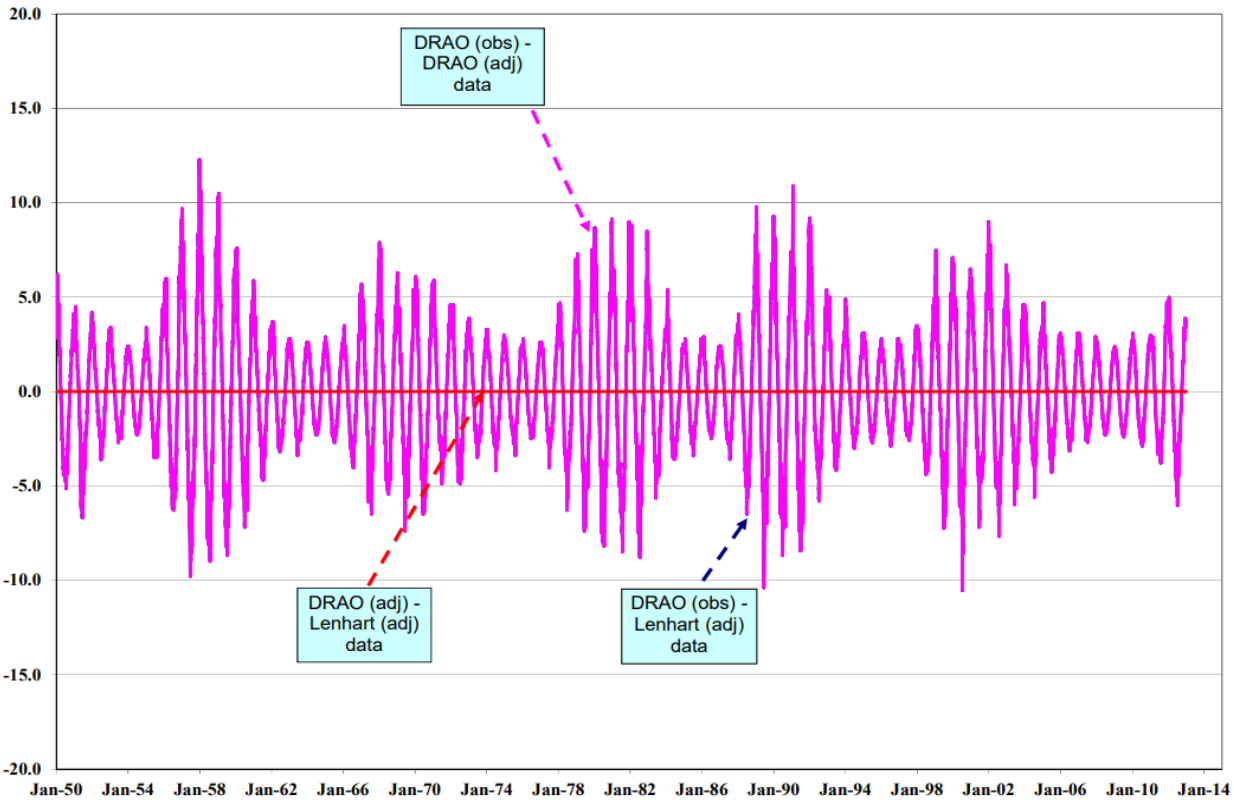

Рисунок 2.2 - Виправлений графік порівняння спостережуваних та скоригованих значень сонячних потоків.

З додаванням виправлень сторонніх стрибків бути не повинно. Спостережувані та скориговані дельти тепер однакові.

Асиміляція даних та фільтрація Калмана.

З середини 1990-х років дані про космічне середовище в реальному часі стали загальнодоступними через інтернет. Насправді потрібно близько 5-20 хвилин, щоб вимірювання магнітосфери та сонячного вітру на місці були телеметрізовані на Землю, оброблені та загальнодоступні у інтернеті. Це порівняно невелика затримка порівняно із шкалами часу грози (0,5-3 години) або шкалами часу шторму (десятки годин-днів).

Доступність даних дозволила розробити моделі, які могли б приймати та використовувати вимірювання для поліпшення якості прогнозу. В результаті стало можливим засвоєння даних. Асиміляція даних - це включення вимірювань у моделі з метою покращення якості прогнозу.

Реалізація асиміляції даних включає вибір методу локальної корекції прогнозу за допомогою фактичних вимірювань. Одним з основних методів інтеграції вимірювань в оцінку стану або контроль є фільтр Калмана. Нижче коротко обговоримо дискретний фільтр Калмана, який, як показано, є оптимальним оцінювачем стану для систем з лінійною динамікою. Його поширення на нелінійні системи або розширений фільтр Калмана [Welch and Bishop, 2001; Rigler, 2004].

2.6 Фільтрація Калмана.

Розглянемо космічне середовище, динаміка якого задається:

$$x_k = Ax_{k-1} + Bu_{k+1} + w_{k-1} \tag{2.14}$$

де A - динаміка системи, а B - зв'язок із зовнішнім джерелом u_k. За винятком терміну шуму wk-1, рівняння (2.14) є аналогом (2.6). Стан x_k неможливо спостерігати безпосередньо, але замість цього вимірювання y_k отримуються за допомогою функції вимірювання H.

$$y_k = Hx_k + v_k \tag{2.15}$$

Члени шуму для динамıки системи, wk та для спостереження, uk, зазвичай розподіляються з нульове середнє та коваріації Q та R відповідно. Метою є покращення нашої оцінки y_x стану космічного середовища за допомогою наявних вимірювань y_x більш конкретно, для зменшення невизначеності в y_x, вираженої як коваріація помилки:

$$P_k = E\left[e_k e_k^T \right] \tag{2.16}$$

Фільтр Кальмана є оптимальним вирішувачем цієї проблеми. Він складається з корекції оцінки

$$\hat{x}_k \leftarrow \hat{x}_k + K\left(y_k - H\hat{x}_k^- \right) \tag{2.17}$$

де оцінка коригується різницею між фактичним виміром y_k та прогнозом y_kH. Різниця модулюється коефіцієнтом підсилення Калмана, К. Мінімізуючи варіацію помилок P_k, отримуємо коефіцієнт посилення:

$$K_k = \frac{P_k H^T}{H P_k H^T + R} \tag{2.18}$$

Оціночний стан та його варіація можуть бути записані в такому вигляді:

$$\begin{aligned} \hat{x_k} &= A\hat{x}_{k-1} + Bu_k + K_k\left(y_k - H\hat{x}_k^-\right) \\ P_k &= \left(1 - K_k H\right)\left(AP_{k-t}A^T + Q\right) \end{aligned} \tag{2.19}$$

Іоносферна асиміляція даних

Значний прогрес вже досягнуто в іоносферному моделюванні з використанням методів асиміляції даних із кількома іоносферними програмами погоди. Однією з найтриваліших зусиль є модель широти для асимілятивного картографування іоносферної електродинаміки (AMIE) [Kamide et al., 1981; Річмонд і Каміде, 1988].

Алгоритм AMIE поглинає радіолокаційні, супутникові та іонозондні вимірювання електричних полів та щільностей, наземні магнітометри, вимірювання магнітних полів та інші потоки даних. Зовсім недавно ці зусилля переросли у модель реального часу.

Зусилля Глобальної асиміляції іоносферних вимірювань (GAIM) під керівництвом Університету Юти створило засновану на фізиці модель іоносферної плазмасфери, яка коригується в режимі реального часу за допомогою асиміляції даних. Модель асиміляції - це фільтр Калмана, який отримує різноманітний набір вимірювань: загальний вміст електронів (TEC), профілі електронної щільності іонозондів, дані окультації, дані ланцюга томографії, випромінювання УФ-випромінювання тощо. розподіл електронної густини від 90 км до 35000 км (геосинхронна орбіта), а також

кілька інших похідних електронних параметрів. У своєму режимі специфікації GAIM точно реконструює іоносферну електронну щільність.

Перевірка моделі

Після того, як модель космічної погоди пройшла стадію розробки, вона проходить випробування, щоб можна оцінити її точність та потенційний прогноз. Двома основними етапами тестування є валідація та верифікація: під час валідації вимірюється, наскільки добре модель робить те, що запрограмована, в тому числі, наскільки це чисельно точно; під час верифікації порівнюється модель із реальними вимірами. Решта цього розділу обговорює основи перевірки. Точність прогнозування моделі вимірюється за допомогою статистичних функцій або метрик, в абсолютному або відносному сенсі. Одним із стандартних таких абсолютних показників є коефіцієнт лінійної кореляції між фактичною (спостережуваною) активністю іх та передбаченням моделі іх кореляція моделі даних:

$$C_{\hat{x},x} = \frac{1}{N}\frac{1}{\sigma_{\hat{x}}\sigma_x}\sum_{i=1}^{N}\left(\hat{x}_i - \langle\hat{x}\rangle\right)\left(x_i - \langle x\rangle\right) \qquad (2.20)$$

де N - розмір прогнозованої вибірки, а σі - відповідні стандартні відхилення для спостережень та прогнозів. Більш детальною діагностикою, ніж коефіцієнт кореляції, є помилка прогнозування або різниця між прогнозом моделі та фактичною активністю системи. Зазвичай це середньоквадратична помилка

$$e_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{x}_i - x_i\right)^2} \qquad (2.21)$$

яка порівнюється із стандартним відхиленням вибірки спостережень, Третьою стандартною діагностикою є ефективність прогнозування, яка є

відсотковою дисперсією спостережень, що можна пояснити прогнозами моделі

$$PE = 1 - \frac{1}{N\sigma_x^2}\sum_{i=1}^{N}\left(\hat{x}_i - x_i\right)^2 = 1 - \frac{e_{rms}^2}{\sigma_x^2} \tag{2.22}$$

Коли розробляються різні моделі , стає все більш важливим виразити точність однієї моделі проти точності іншої. Навик моделі щодо еталонної моделі - це точність її прогнозів щодо точності прогнозів еталонної моделі. Ця відносна точність виражається метриками, які називаються оцінками навичок. Стандартна оцінка кваліфікації виражається через середньоквадратичну похибку моделі порівняно з аналогічною функцією похибки еталонної моделі.

$$SS = 1 - \frac{e_{rms}^2}{\left(e_{rms}^{(ref)}\right)^2} \tag{2.23}$$

Проста, але важлива еталонна модель - це модель стійкості періодична або періодична моделі та моделі кліматології. Якщо кліматологія є середньою активністю x, то оцінка кваліфікації стає ефективністю прогнозування (2.23). Додаткова інформація про верифікацію та її метрики доступна у [Doggett 1996].

Висновки до розділу:

Описано основні типи моделей прогнозування та їх особливості. Окреслено основні методи фільтрації вхідних даних. Розглянуто окремі методи нормалізації цих даних. Подані головні властивості моделей прогнозування.

Розглянута задача вибору показників прогнозування та запропоновано її вирішення. Запропоновано новий підхід до ідентифікації моделі прогнозування. Обґрунтовано використання вибраних параметрів стану комічної погоди. Як вхідний параметр запропоновано використовувати добуток південної компоненти геомагнітного поля та швидкості сонячного вітру. Як вихідний використовується D_{st} індекс.

Розглянуто гарантовану модель прогнозування D_{st} індексу. Запропоновано новий підхід до ідентифікації моделі прогнозування на основі експериментальних даних. Усі моделі, засновані на наборі даних дистанційних спостережень.

Подана проблема вибору джерел вхідних даних. Описані можливі рішення. Подані основні методи фільтрації та асиміляції вхідних даних.

3. ІМПЛЕМЕНТАЦІЯ ЗАПРОПОНОВАНОГО СПОСОБУ ІДЕНТИФІКАЦІЇ МОДЕЛЕЙ

3.1 Математичне моделювання

Під час дослідження використано поліном дискретної моделі індексу D_{st}, що дозволяє представляють вихідний сигнал через систему:

$$y(k)=F[y(k-1),..y(k-n_y),...u(k-1),...u(k-n_u),\xi(k),...\xi(k-n_\xi)], \quad 3.1)$$

Для вибору конкретного полінома F застосовано процедури структурно-параметричної ідентифікації моделі, суть якої полягає в додаванні таких нелінійних термінів для покращення передбачуваності та стабільності. Процес вибору нелінійних термінів зазвичай вважається закінченим, коли похибка прогнозування задовольняє певному тесту на точність моделі, що підтверджує неможливість подальшого зменшення помилок на даних спостережень.

Щоб спростити задачу прогнозування індексу Dst, розглядаємо магнітосферу як систему з одним входом і одним виходом. Опис нелінійних процесів також можна вдосконалити, ввівши в модель вільні параметри та шум. Як показали чисельні розрахунки, вибір параметра як вхідних даних дозволяє нам розробити модель, адекватну прогнозу.

Очевидно, що краще прогнозувати не результат, а на основі математичної моделі процесу визначає передбачення $\widetilde{y}_{k+n}$ наборів y_{k+n}

Розглянемо спочатку визначення прогнозної оцінки $\widetilde{y}_{k+n}$ для n = 1. Для цього потрібно оцінити $\widetilde{u}_{k-(m-l)+1}$ значення $u_{k-(m-l)+1}$. Розглянемо клас процесів, для яких існує апріорна оцінка швидкості зміни u_k у вигляді

$$\Delta u_k \in \delta \mathbf{u} = \left\{\Delta u_k : \left|\Delta u_k = u_{k-1} - u_k\right| \le \delta = const. \right. \quad (3.2)$$

Зауважимо, що в цьому випадку не приймається величина δ. З (3.2) випливає оцінка $\widetilde{u}_{k+1}$ значення u_{k+1} у вигляді:

$$\widetilde{u}_{k+1} \in \widetilde{\mathbf{u}}_{k+1} = \{u_{k+1} : u_k - \delta \le u_{k+1} \le u_k + \delta\}. \tag{3.3}$$

Перепишемо рівняння (3.2) для (k + 1) кроку:

$$y_{k+1} = f(y_k, y_{k-1}, \ldots y_{k-m+1}, u_{k-(m-l)+1}, u_{k-(m-l)}, \ldots u_{k-m+1}). \tag{3.4}$$

Додаванням (3.4) замість невідомої на k-му кроці його оцінки u_{k+1}, отримуємо:

$$y_{k+1} = f(y_k, y_{k-1}, \ldots y_{k-m+1}, u_{k-(m-l)} + \Delta u_k, \ldots u_{k-m+1}). \tag{3.5}$$

З (3.5) знаходимо інтервал передбачення [3.2], з величину $\widetilde{y}_{k+1}$, у вигляді:

$$\widetilde{y}_{k+1} \in \widetilde{\mathbf{y}}_{k+1} = \left\{ \widetilde{y}_{k+1} : \underline{y}_{k+1} \le \widetilde{y}_{k+1} \le \overline{y}_{k+1} \right\}, \tag{3.6}$$

Де:

$$\underline{y}_{k+1} = \min_{\Delta u_k \in \delta \mathbf{u}} f(y_k, y_{k-1}, \ldots y_{k-m+1}, u_{k-(m-l)} + \Delta u_k, \ldots u_{k-m+1}) \tag{3.7}$$

$$\overline{y}_{k+1} = \max_{\Delta u_k \in \delta \mathbf{u}} f(y_k, y_{k-1}, \ldots, y_{k-m+1}, x_{k-(m-l)} + \Delta u_k, \ldots u_{k-m+1}) \tag{3.8}$$

Якщо $f(\ldots, u_{k-(m-l)} + \Delta u_k, \ldots)$ є монотонною функцією , то його мінімум і максимум належать межам інтервалу $\delta \mathbf{u}$. Якщо, однак $f(\ldots, u_{k-(m-l)} + \Delta u_k, \ldots)$ - це багато-екстремумна функція, яка замінює інтервал $\{-\delta; \delta\}$ набором дискретних значень $\Delta u^{(i)}$, де i = l; K, так зменшуємо задачу до комбінаторної задачі, яка вирішується простим перебором.

Тепер знаходимо оцінку $\widetilde{y}_{k+2}$ величини y_{k+2}. Для цього перепишемо рівняння (3.7) для кроку (k +2)

$$y_{k+2} = f(y_{k+1}, y_k, \dots, y_{k-m+2}, u_{k-(m-l)+2}, \dots u_{k-m+2}) \quad (3.9)$$

Значення $u_{k-(m-l)+2}$ являється

$$u_{k-(m-l)+2} = u_{k-(m-l)} + \Delta u_{k+1} + \Delta u_k \quad (3.10)$$

Оскільки оцінка (3.5) величини Δu_k є дійсною для будь-якої k, отримуємо з (24) та (25) оцінку для $\tilde{u}_{k-(m-l)+2}$ кількості $u_{k-(m-l)+2}$:

$$\tilde{u}_{k-(m-l)+2} \in \tilde{\mathbf{u}}_{k-(m-l)+2} \quad (3.11)$$

де

$$\tilde{\mathbf{u}}_{k-(m-l)+2} = \left\{u : \underline{u}_{k-(m-l)+1} - \delta \le u_{k-(m-l)+1} \le \overline{u}_{k-(m-l)+1} + \delta\right\} \quad (3.12)$$

З (3.9), (3.8) і (3.6) випливає оцінка $\tilde{y}_{k+2}$, в формі

$$\tilde{y}_{k+2} \in \tilde{\mathbf{y}}_{k+2} \quad (3.13)$$

, де

$$\begin{gathered} \underline{y}_{k+2} \le \tilde{y}_{k+2} \le \overline{y}_{k+2}, \\ \tilde{y}_{k+1} \in \tilde{y}_{k+1} = \left\{\tilde{y}_{k+1} : \underline{y}_{k+1} \le \tilde{y}_{k+1} \le \overline{y}_{k+1}\right\} \end{gathered} \quad (3.14)$$

$$\begin{gathered} \underline{y}_{k+1} = \min_{\Delta u_k \in \delta u} F^L(y_k, y_{k-1}, \dots, y_{k-m+1}, u_{k-(m-l)} + \\ + \Delta u_k, \dots, u_{k-m+1}). \end{gathered} \quad (3.15)$$

З (3.14), (3.15) випливає, що інтервал, встановлений $\tilde{\mathbf{y}}_{k+2}$, залежить лише від трьох невизначених величин y_{k+2}, $u_{k-(m-l)+1}$, та Δu_k для яких наведені їх відповідні кратні оцінки. Тому значення інтервалу $\left\{\underline{y}_{k+2}; \overline{y}_{k+2}\right\}$ більше, ніж інтервалу $\left\{\underline{y}_{k+1}; \overline{y}_{k+1}\right\}$. Все сказане про розв'язання задач (3.7), (3.8), залишається справедливим для задач (3.14), (3.15), з тією лише різницею, що ці задачі оптимізації вирішуються у тривимірному просторі.

Визначення оцінок $\widetilde{y}_{k+n}$, значення y_{k+n},де n> 3, аналізується подібним чином, і ці оцінки стають більш грубими із збільшенням значення.

Для того, щоб зробити прогнозні оцінки індексу D_{st}, використовуючи описаний вище підхід, використано дві моделі для реконструкції з експериментальних даних: лінійну та нелінійну Використовуючи вираз (16) та генетичний алгоритм пошуку невідомих параметрів, Отримано модель:

$$\overline{y}_{k+1} = \max_{\Delta u_k \in \delta u} F^L(y_k, y_{k-1}, \ldots, y_{k-m+1}, u_{k-(m-l)} + \\ + \Delta u_k, \ldots, u_{k-m+1}), \tag{3.16}$$

де y - індекс D_{st}, u - значення B_z*V, k - дискретний час, а $\theta_1, \ldots, \theta_{11}$ - невідомі параметри.

3.2 Обґрунтування засобів розробки ПЗ

Під час вибору середовища розробки для створення програмного забезпечення були сформовані такі обов'язкові вимоги до нього:

- проста реалізація графічного інтерфейсу програми;
- можливість скомпонувати програмне забезпечення у вигляді бібліотеки для подальшого використання за основу у інших додатках, зокрема веб додатках;
- мультиплатформність, для забезпечення подальшого розвитку програмного забезпечення на різних платформах;
- достатній набір готових програмних бібліотек, для пришвидшення процесу розробки додатку.

Під визначений набір критеріїв в основному підходять лише дві платформи, такі як Python і його фреймворк для веб додатків Django, що мають велику бібліотеку додатків у сфері математичних обчислювань. Та Javascript, за його кроссплатформеність, що дозволяє розроблювати на

ньому додатки під будь яку платформу. А також Ruby за його простоту у використанні, та наявність необхідних бібліотек.

Згідно з останніх досліджень, Python є кращою мовою програмування для роботи з даними. Для роботи з даними потрібна проста у використанні мова з пристойною бібліотекою і широкою участю спільноти. Проекти з неактивними спільнотами зазвичай з меншою ймовірністю будуть підтримувати або оновлювати свої платформи.

При роботі з обробкою даних, дуже важливо не ускладнювати собі поставлену задачу, зав'язаючи в складних вимогах до програмування. Тому використовують мови програмування, такі як Python і Ruby, для безпроблемного виконання завдань.

Ruby відмінно підходить для виконання таких завдань, як очищення і зміна даних, а також для інших завдань попередньої обробки даних. Однак в ньому не так багато бібліотек машинного навчання, як в Python. Це дає йому перевага в області математичних обрахунків та машинного навчання.

Python також дозволяє розробникам розгортати програми та запускати прототипи, що значно прискорює процес розробки. Коли проект перетворюється в аналітичний інструмент або додаток, його можна при необхідності перенести на більш складні мови, такі як Java або C.

Python задовольняє цю потребу, будучи мовою програмування загального призначення. Це дозволяє створювати вихідні дані CSV для зручного читання даних в електронній таблиці. В якості альтернативи більш складні вихідні файли, які можуть бути отримані кластерами машинного навчання для обчислень.

Розглянемо наступний приклад:

Прогнози погоди ґрунтуються на минулих показах погодних записів за сторіччя. Машинне навчання може допомогти в створенні більш точних прогнозних моделей на основі минулих погодних явищ. Python може це зробити, тому що він легкий і ефективний при виконанні коду, але він також багатофункціональний. Крім того, ця мова може підтримувати об'єктно-орієнтовані, структуровані і функціональні стилі програмування, що означає, що він може знайти додаток де завгодно.

В даний час в індексі пакетів Python більше 70 000 бібліотек, і це число продовжує зростати. Як згадувалося раніше, він пропонує безліч бібліотек, орієнтованих на науку про дані. Простий пошук в Google показує близько 10 найкращих бібліотек для роботи з великими даними. Можливо, найпопулярнішою бібліотекою аналізу даних є бібліотека з відкритим кодом під назвою pandas. Це високопродуктивний набір додатків, які значно спрощують аналіз даних в Python.

Незалежно від того, що планується робити з Python, чи це передбачення причинно-наслідкових явищ або аналітика, він має набір інструментів для виконання безлічі потужних обчислень.

Після детального аналізу усіх платформ вибраний саме Python через його оптимізацію для розрахунків математичних моделей. Також за свою простоту та велику кількість готових бібліотек, що дозволяють значно спростити процес створення програмного забезпечення для автоматизації структурно-параметричної ідентифікації NARMAX моделей.

Python - одна з найпопулярніших мов програмування у світі, має ряд потужних пакетів, які допомагають вирішувати складні математичні проблеми простим та ефективним способом. Ці основні можливості для створення додатків у різних сферах, таких як машинне навчання, використовуючи знання в області обчислювальної математики.

Для спрощення виконання математичних обрахунків у програмному забезпеченні використовується бібліотека NumPy, що являє собою набір інструментів для зберігання

Для графічної складової додатку, використовується бібліотека Matplotlib . Вона призначена для побудови графіків для числового математичного розширення NumPy. Вона надає об'єктно-орієнтований API для вбудовування графіків у додатки з використанням наборів інструментів загального користувальницького інтерфейсу, таких як Tkinter, wxPython, Qt або GTK +. Існує також процедурний інтерфейс "pylab", заснований на OpenGL, розроблений ідентично до інтерфейсу MATLAB, хоча його використання не рекомендується.

3.2.1 Використання бібліотеки Matplotlib.

Matplotlib має розгалужену текстову підтримку, включаючи підтримку математичних виразів, підтримку справжнього типу для растрових та векторних виходів, текст, розділений новим рядком з довільними обертаннями, та підтримку Unicode. Оскільки він вбудовує шрифти безпосередньо у вихідні документи, наприклад, для постскриптуму або PDF, те, що відображається на екрані, це те, що буде отримано у друкованому вигляді. Підтримка FreeType дає дуже гарні, згладжені шрифти, які добре виглядають навіть при невеликих розмірах растра. Matplotlib включає власний matplotlib.font_manager (завдяки Полу Барретту), який реалізує крос-платформний алгоритм пошуку шрифтів, сумісний з W3C.

Користувач має повний контроль над властивостями тексту (розміром шрифту, вагою шрифту, розташуванням тексту та кольором тощо) із розумними значеннями за замовчуванням, встановленими у rc-файлі. І що особливо важливо, для тих, хто цікавиться математичними чи науковими цифрами, Matplotlib реалізує велику кількість математичних

символів і команд TeX, підтримуючи математичні вирази в будь-якому місці вашої фігури.

3.2.3 Робота з даними

Бази даних використовуються у проекті для зберігання отриманих результатів передбачень та отриманих зі сторонніх джерел реальних даних. У даному проекті прийнято рішення використовувати NoSql бази данних. Для зберігання великих обсягів даних, які можуть мати мало або взагалі не мати структури. Бази даних NoSQL не обмежують типи даних, які можна зберігати разом. Бази даних NoSQL також дозволяють додавати нові типи даних у міру зміни ваших потреб. За допомогою баз даних, орієнтованих на документи, є можливість зберігати дані в одному місці без необхідності заздалегідь визначати тип даних.

Також до уваги були прийняті наступні фактори:

Максимальне використання хмарних обчислень та зберігання. Для того щоб хмарне рішення масштабованим, дані повинні легко передаватися на декількох серверах.

Під час ітераційної розробки або регулярного оновлення структури даних, реляційна база даних уповільнює роботу. Однак, оскільки дані NoSQL не потрібно готувати заздалегідь, можна регулярно оновлювати структуру даних з мінімальним простоєм.

Теорема CAP (узгодженість, доступність і допуск на розділи) стверджує, що в будь-якій розподіленій системі одночасно можуть використовуватися лише дві з трьох властивостей CAP. Налаштування цих властивостей на користь високої толерантності розділів дозволяє підвищити горизонтальну масштабованість.

Алгоритм структурно-параметричної ідентифікації моделі

Процес структурно-параметричної ідентифікації моделі складається з декількох етапів:

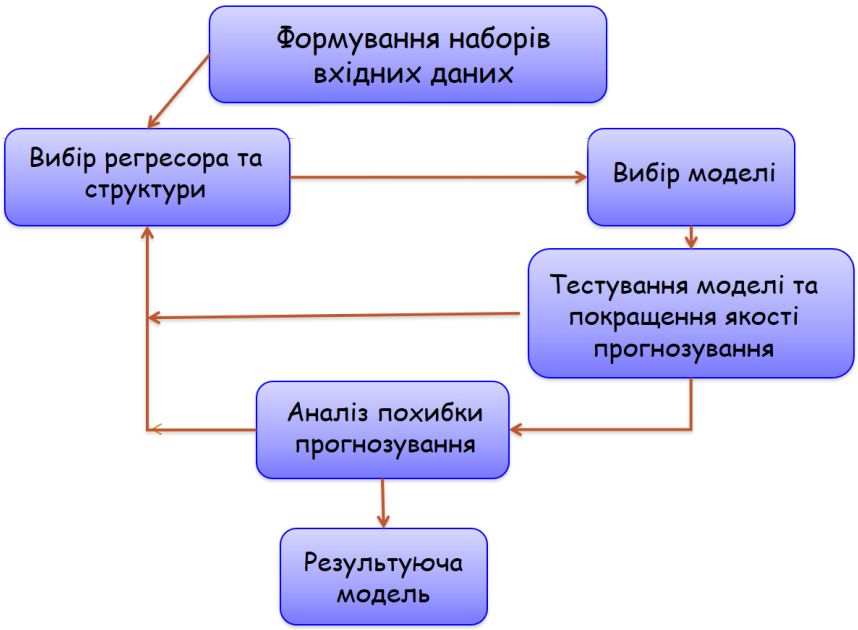

Рисунок 3.3 Структурна схема ПЗ

Спочатку формуються набори вхідних даних згідно методів описаних вище. У нашому випадку вхідними параметрами вибрана швидкість сонячного вітру та південна компонента магнітного поля. Для побудови конкретної моделі, запропоновано використовувати такі вхідні: параметри південна компонента земного магнітного поля і швидкість сонячного вітру. проведено оптимізацію вхідних параметрів шляхом покомпонентного множення. Тобто, як результат отримано лише один набір вхідних параметрів, що характеризує одночасно декілька наборів станів системи. При цьому складність самої моделі залишається лінійною. Як вихідний параметр вибрано D_{st} індекс. Також, розроблено алгоритм побудови прогнозу комічної погоди на основі запропонованого методу та алгоритм побудови програмного забезпечення, що буде виконувати прогнозування.

3.4 Функціональна структура програмного забезпечення

Розроблене програмне забезпечення розбито на декілька основних функціональних модулів, кожен з яких складеться декількох класів.

Модуль роботи з сервісами, що надають дані спостережень. Він складається з декількох класів: BaseAPIModule, що відповідає за базову логіку обробки запитів. Та класів що успадковують його та реалізують логіку роботи з конкретними джерелами данних.

```
class BaseAPIModule():
    def __init__(self, access_token, version="5.73"):
        self.access_token = access_token
        self.api_version = version
        self.device_id = uuid.uuid4().hex[:16]
        self.api_host = "api.weather.com"

    def fetch(self, method, data=None):
        if data is None:
            data = {}

        return method, data

    def get_hash(self, additional: list, user_id):
        ids = "".join(map(str, additional)) + "3aUFMZGRCJ"
        ids_hash = hashlib.md5(ids.encode()).hexdigest()

        user = str(int(user_id) ^ 202520)
        user_hash = hashlib.md5(user.encode()).hexdigest()

        device = str(self.device_id) + "0MgLscD6R3"
        device_hash = hashlib.md5(device.encode()).hexdigest()

        return "{}#{}#{}".format(ids_hash, user_hash, device_hash)

    def send_answer(self, *, coins_answer: bool, , value, date, type):
        secure_hash = self.get_hash([value, date], type)

        data = {
            "date": date,
            "value": value,
            "type": self.type,
            "hash": secure_hash,
        }

        if coins_answer:
            data["coins_answer"] = True

        return self.fetch("streamQuiz.sendAnswer", data)
```

Рис. 3.4 BaseAPIModule

Для прикладу:

```
class JDEWeatherAPIModule(BaseAPIModule):
    def __init__(self, access_token, version="5.73"):
        super().__init__(access_token, version=version)

        self.session = requests.Session()
        self.session.headers.update({
            "User-Agent": "/2.3.3 (Redmi Note 5; "
            "Android 28; SDK 1.6.8; com.vk.quiz)".encode(
                "utf-8")
        })

    def fetch(self, method, data=None):
        if data is None:
            data = {}

        data.update({
            "access_token": self.access_token,
            "v": self.api_version,
            "lang": "ru",
            "https": 1
        })

        url = f"https://{self.api_host}/method/{method}"

        content = self.session.post(url, data=data).json()
        error = content.get("error")

        if error is not None:
            raise ApiResponseError(json.dumps(content))

        return content["response"]
```

Рис. 3.5 JDEWeatherAPIModule

Модуль зберігання накопичених даних прогнозів. Для його реалізації використовується локальна база даних на основі NoSql.

```
class Database(object):
    _conf = {
        'map_size': 1024 * 1024 * 1024 * 2,
        'subdir': True,
        'metasync': False,
        'sync': True,
        'lock': True,
        'max_dbs': 64,
        'writemap': True,
        'map_async': True
    }

    def __init__(self, name, conf=None, binlog=True, size=None, master=False):
        conf = dict(self._conf, **conf.get('env', {})) if conf else self._conf
        if size: conf['map_size'] = size
        self._tables = {}
        self._semaphore = False
        self._name = name
        self._env = lmdb.Environment(name, **conf)
        self._db = self._env.open_db()
        self._binlog = None
        self._binidx = None
        self._meta = self.table('__metadata__')
        doc = self._meta.get(b'__node__')
        self._node = doc.get('value') if doc else 0
```

```
        try:
            self.set_binlog(enable=False)
            if binlog:
                self.set_binlog(enable=True)
                # try:
                #     self._semaphore = Semaphore(semaphore_path(name))
                # except ExistentialError:
                #     pass
                self._binlog = self._env.open_db(b'__binlog__', create=binlog)
                self._binidx = self._env.open_db(b'__binidx__', create=binlog)
                with self._env.begin(write=True) as txn:
                    if not txn.stat(db=self._binlog)['entries']:
                        dat = dumps({'txn': []}).encode()
                        txn.put(pack('>Q', 1), dat, db=self._binlog, append=False)

        except lmdb.NotFoundError:
            pass
        except:
            raise

    def set_binlog(self, enable=True):
        if enable:
            if not self._binlog:
                self._binlog = self.env.open_db(b'__binlog__')
                self._binidx = self.env.open_db(b'__binidx__')
        else:
            if self._binlog:
                with self.env.begin(write=True) as txn:
                    txn.drop(self._binlog, True)
                    txn.drop(self._binidx, True)

                self._binlog = None
                self._binidx = None

    def begin(self, *args, **kwargs):
        return Transaction(self, *args, write=True, **kwargs)

    def close(self):
        if hasattr(self, '_env') and self._env:
            self._env.close()
            self._env = None

    def _return_tables(self, all, txn=None):
        def tables():
            result = []
            with lmdb.Cursor(self._db, txn) as cursor:
                if cursor.first():
                    while True:
                        name = cursor.key().decode()
                        if all or name[0] not in ['_', '~']:
                            result.append(name)
                        if not cursor.next():
                            break
            return result

        if not txn:
            with self.env.begin() as txn:
                return tables()
        else:
            return tables()
```

Рис. 3.6 Database

Модуль зберігання накопичених даних реальних спостережень.

Він працює на основі тих основних модулів, що і модуль зберігання даних прогнозів. В основному ці модулі використовують схожі функції, проте мають дещо різні інтерфейси взаємодій через те, що отримують дані у різних вхідних форматах. Основною їх функцією є перетворення даних до уніфікованого формату, що буде використовуватися при створені нових моделей.

Модуль нормалізації вхідних даних.

Цей модуль виконує математичні перетворення даних згідно алгоритмів описаних у другому розділі. Він складається з значної кількості різних методів отримання параметрів

```python
class Estimators:
    """Oridanry Least squares for linear parameter estimation"""

    def __init__(
        self,
        aux_lag=1,
        lam=0.98,
        delta=0.01,
        offset_covariance=0.2,
        mu=0.01,
        eps=np.finfo(np.float).eps,
        gama=0.2,
        weight=0.02,
    ):

        self._eps = eps
        self._mu = mu
        self._offset_covariance = offset_covariance
        self._aux_lag = aux_lag
        self._lam = lam
        self._delta = delta
        self._gama = gama
        self._weight = weight  # <0  e <1
        self._validate_params()

    def _validate_params(self):
        """Validate input params."""
        attributes = {
            "aux_lag": self._aux_lag,
            "lam": self._lam,
            "delta": self._delta,
            "offset_covariance": self._offset_covariance,
            "mu": self._mu,
            "eps": self._eps,
            "gama": self._gama,
            "weight": self._weight,
        }

        for attribute, value in attributes.items():
            if not isinstance(value, (np.integer, int, float)):
                raise ValueError(
                    (
                        f"{attribute} must be int or float (positive)."
                        f"Got {type(attribute)}"
                    )
                )

            if attribute in ["lam", "weight", "offset_covariance"]:
                if value > 1 or value < 0:
                    raise ValueError(
                        f"{attribute} must lies on [0 1] range. Got {value}"
                    )

            if value < 0:
                raise ValueError(
                    (
                        f"{attribute} must be positive. Got {value}"
                        f"Check the documentation for allowed values"
                    )
                )
```

Рис. 3.6 Estimators

Оцінка параметрів моделі за допомогою методу найменших квадратів.

```
    def least_squares(self, psi, y):
        self._check_linear_dependence_rows(psi)

        y = y[self._aux_lag :, 0].reshape(-1, 1)
        theta = (np.linalg.pinv(psi.T @ psi)) @ psi.T @ y
        return theta
```

Рис. 3.7 Estimators, least_squares method

Оцінка параметрів, що використовуючи рекурсивний метод загальних найменших квадратів.

```
    def recursive_least_squares(self, psi, y):
        y, n_theta, n, theta, self.xi = self._initial_values(y, psi)

        p = np.eye(n_theta) / self._delta

        for i in range(2, n):
            psi_tmp = psi[i, :].reshape(-1, 1)
            k_numerator = self._lam ** (-1) * p.dot(psi_tmp)
            k_denominator = 1 + self._lam ** (-1) * psi_tmp.T.dot(p).dot(psi_tmp)
            k = np.divide(k_numerator, k_denominator)
            self.xi[i, 0] = y[i, 0] - psi_tmp.T @ theta[:, i - 1]
            theta[:, i] = list(theta[:, i - 1].reshape(-1, 1) + k.dot(self.xi[i, 0]))

            p1 = p.dot(psi[i, :].reshape(-1, 1)).dot(psi[i, :].reshape(-1, 1).T).dot(p)
            p2 = (
                psi[i, :].reshape(-1, 1).T.dot(p).dot(psi[i, :].reshape(-1, 1))
                + self._lam
            )

            p_numerator = p - np.divide(p1, p2)
            p = np.divide(p_numerator, self._lam)

        self.theta_evolution = theta.copy()
        return theta[:, -1].reshape(-1, 1)
```

Рис. 3.8 Estimators, recurcive_east_squares method

Оцінка параметрів, що використовуючи метод загальних найменших квадратів.

```
    def total_least_squares(self, psi, y):
        y = y[self._aux_lag :, 0].reshape(-1, 1)
        full = np.hstack((psi, y))
        n = psi.shape[1]
        u, s, v = np.linalg.svd(full, full_matrices=True)
        theta = -v.T[:n, n:] / v.T[n:, n:]
        return theta.reshape(-1, 1)
```

Рис. 3.9 Estimators, total_east_squares method

Модуль, графічного інтерфейсу. Специфіка виконаної роботи потребувала написання додаткових класів для реалізації необхідного функціоналу графічних нод:

```
class ImageViewNode(Node):
    """Node that displays image data in an ImageView widget"""
    nodeName = 'ImageView'

    def __init__(self, name):
        self.view = None
        ## Initialize node with only a single input terminal
        Node.__init__(self, name, terminals={'data': {'io':'in'}})

    def setView(self, view):  ## setView must be called by the program
        self.view = view

    def process(self, data, display=True):
        ## if process is called with display=False, then the flowchart is being operated
        ## in batch processing mode, so we should skip displaying to improve performance.

        if display and self.view is not None:
            ## the 'data' argument is the value given to the 'data' terminal
            if data is None:
                self.view.setImage(np.zeros((1,1))) # give a blank array to clear the view
            else:
                self.view.setImage(data)

class UnsharpMaskNode(CtrlNode):
    """Return the input data passed through an unsharp mask."""
    nodeName = "UnsharpMask"
    uiTemplate = [
        ('sigma',  'spin', {'value': 1.0, 'step': 1.0, 'bounds': [0.0, None]}),
        ('strength', 'spin', {'value': 1.0, 'dec': True, 'step': 0.5, 'minStep': 0.01, 'bounds': [0.0, None
    ]
    def __init__(self, name):
        ## Define the input / output terminals available on this node
        terminals = {
            'dataIn': dict(io='in'),    # each terminal needs at least a name and
            'dataOut': dict(io='out'),  # to specify whether it is input or output
        }                               # other more advanced options are available
                                        # as well..

        CtrlNode.__init__(self, name, terminals=terminals)

    def process(self, dataIn, display=True):
        # CtrlNode has created self.ctrls, which is a dict containing {ctrlName: widget}
        sigma = self.ctrls['sigma'].value()
        strength = self.ctrls['strength'].value()
        output = dataIn - (strength * pg.gaussianFilter(dataIn, (sigma,sigma)))
        return {'dataOut': output}
```

Рис. 3.10 Реалізація графічних методів

Клас MainOutputWindow відповідає за головне вікно розробленого ПЗ. Він ініціалізує роботу програми та контролює основні функціональні можливості інших модулів.

```
class MainOutputWindow():
    def init() :
        app = QtGui.QApplication([])

        ## Create main window with a grid layout inside
        win = QtGui.QMainWindow()
        win.setWindowTitle('pyqtgraph example: FlowchartCustomNode')
        cw = QtGui.QWidget()
        win.setCentralWidget(cw)
        layout = QtGui.QGridLayout()
        cw.setLayout(layout)

        ## Create an empty flowchart with a single input and output
        fc = Flowchart(terminals={
            'dataIn': {'io': 'in'},
            'dataOut': {'io': 'out'}
        })
        w = fc.widget()

        layout.addWidget(fc.widget(), 0, 0, 2, 1)

        ## Create two ImageView widgets to display the raw and processed data with contr
        ## and color control.
        v1 = pg.ImageView()
        v2 = pg.ImageView()
        layout.addWidget(v1, 0, 1)
        layout.addWidget(v2, 1, 1)
        win.show()

        ## Set the raw data as the input value to the flowchart
        fc.setInput(dataIn=data)

        library = fclib.LIBRARY.copy() # start with the default node set
        library.addNodeType(ImageViewNode, [('Display',)])
        # Add the unsharp mask node to two locations in the menu to demonstrate
        # that we can create arbitrary menu structures
        library.addNodeType(UnsharpMaskNode, [('Image',),
                                              ('Submenu_test','submenu2','submenu3')])
        fc.setLibrary(library)
        ## Now we will programmatically add nodes to define the function of the flowchar
        ## Normally, the user will do this manually or by loading a pre-generated
        ## flowchart file.

        v1Node = fc.createNode('ImageView', pos=(0, -150))
        v1Node.setView(v1)

        v2Node = fc.createNode('ImageView', pos=(150, -150))
        v2Node.setView(v2)

        fNode = fc.createNode('UnsharpMask', pos=(0, 0))
        fc.connectTerminals(fc['dataIn'], fNode['dataIn'])
        fc.connectTerminals(fc['dataIn'], v1Node['data'])
        fc.connectTerminals(fNode['dataOut'], v2Node['data'])
        fc.connectTerminals(fNode['dataOut'], fc['dataOut'])
```

Рис. 3.11 MainOutputWindow

Модуль, що відповідає за зберігання у пам’яті та обробку конкретних моделей. Основним класом цього модулю є PolynomialNarmax, що являє собою готову модель

```
class PolynomialNarmax(
    GenerateRegressors, HouseHolder, InformationMatrix, ResiduesAnalysis, Estimators):

    def __init__(
        self,
        non_degree=2,
        ylag=2,
        xlag=2,
        order_selection=False,
        info_criteria="aic",
        n_terms=None,
        n_inputs=1,
        n_info_values=10,
        estimator="least_squares",
        extended_least_squares=True,
        aux_lag=1,
        lam=0.98,
        delta=0.01,
        offset_covariance=0.2,
        mu=0.01,
        eps=np.finfo(np.float).eps,
        gama=0.2,
        weight=0.02,
    ):

        self.non_degree = non_degree
        self._order_selection = order_selection
        self._n_inputs = n_inputs
        self.ylag = ylag
        self.xlag = xlag
        [self.regressor_code, self.max_lag] = GenerateRegressors().regressor_space(
            non_degree, xlag, ylag, n_inputs
        )

        self.info_criteria = info_criteria
        self.n_info_values = n_info_values
        self.n_terms = n_terms
        self.estimator = estimator
        self._extended_least_squares = extended_least_squares
        self._eps = eps
        self._mu = mu
        self._offset_covariance = offset_covariance
        self._aux_lag = aux_lag
        self._lam = lam
        self._delta = delta
        self._gama = gama
        self._weight = weight  # <0  e <1
        self._validate_params()
```

Рис. 3.12 Клас Polynomial Narmax

Основними його методами є:

Метод error_reduction_ratio – виконує алгоритм вменшення похибки

```
def error_reduction_ratio(self, psi, y, process_term_number):

    squared_y = y[self.max_lag:].T @ y[self.max_lag:]
    tmp_psi = np.array(psi)
    y = np.array([y[self.max_lag:, 0]]).T
    tmp_y = np.copy(y)
    [n, dimension] = tmp_psi.shape
    piv = np.arange(dimension)
    tmp_err = np.zeros(dimension)
    err = np.zeros(dimension)

    for i in np.arange(0, dimension):
        for j in np.arange(i, dimension):
            num = np.array(tmp_psi[i:n, j].T @ tmp_y[i:n])
            num = np.power(num, 2)
            den = np.array(
                (tmp_psi[i:n, j].T @ tmp_psi[i:n, j]) * squared_y)
            tmp_err[j] = num / den

        if i == process_term_number:
            break

        tmp_err = list(tmp_err)
        piv_index = tmp_err.index(max(tmp_err[i:]))
        err[i] = tmp_err[piv_index]
        tmp_psi[:, [piv_index, i]] = tmp_psi[:, [i, piv_index]]
        piv[[piv_index, i]] = piv[[i, piv_index]]
        x = tmp_psi[i:n, i]

        v = HouseHolder()._house(x)

        aux_1 = tmp_psi[i:n, i:dimension]

        row_result = HouseHolder()._rowhouse(aux_1, v)

        tmp_y[i:n] = HouseHolder()._rowhouse(tmp_y[i:n], v)

        tmp_psi[i:n, i:dimension] = np.copy(row_result)

    tmp_piv = piv[0:process_term_number]
    tmp_psi = np.array(psi)
    psi_orthogonal = np.copy(tmp_psi[:, tmp_piv])
    tmp_psi = np.array(psi)
    regressor_code_buffer = self.regressor_code
    model_code = np.copy(regressor_code_buffer[tmp_piv, :])
    return model_code, err, piv, psi_orthogonal
```

Рис. 3.12 Клас Polynomial Narmax, метод error_reduction_ratio

Метод train – метод що приймає у себе два набори даних. Вхідні та вихідні значення параметрів моделі

```
def train(self, X, y):
    if y is None:
        raise ValueError("y cannot be None")

    check_X_y(X, y)

    reg_Matrix = InformationMatrix().build_information_matrix(
        X, y, self.xlag, self.ylag, self.non_degree
    )

    if self._order_selection is True:
        self.info_values = self.information_criterion(X, y)

    if self.n_terms is None and self._order_selection is True:
        model_length = np.where(
            self.info_values == np.amin(self.info_values))
        model_length = int(model_length[0] + 1)
        self.n_terms = model_length
    elif self.n_terms is None and self._order_selection is not True:
        raise ValueError(
            "If order_selection is False, you must define n_terms value."
        )
    else:
        model_length = self.n_terms

    (self.final_model, self.err, self.pivv, psi) = self.error_reduction_ratio(
        reg_Matrix, y, model_length
    )

    parameter_estimation = Estimators(
        aux_lag=self.max_lag,
        lam=self._lam,
        delta=self._delta,
        offset_covariance=self._offset_covariance,
        mu=self._mu,
        eps=self._eps,
        gama=self._gama,
        weight=self._weight,
    )
    self.theta = getattr(parameter_estimation, self.estimator)(psi, y)

    if self._extended_least_squares is True:
        self.theta = self._unbiased_estimator(
            psi, X, y, self.theta, self.max_lag, parameter_estimation
        )
    return self
```

Рис. 3.12 Клас Polynomial Narmax, метод train

Модуль тестування ПЗ.

Також у рамках роботи була створена система модульних тестів для програмного забезпечення, щоб забезпечити можливість подальшого вдосконалення та розширення ПЗ

```python
def create_test_data(n=1000):
    # np.random.seed(42)
    # x = np.random.uniform(-1, 1, n).T
    # y = np.zeros((n, 1))
    theta = np.array([[0.6], [-0.5], [0.7], [-0.7], [0.2]])
    data = np.loadtxt("examples/datasets/data_for_testing.txt")
    x = data[:, 0].reshape(-1, 1)
    y = data[:, 1].reshape(-1, 1)
    return x, y, theta


def test_error_reduction_ration():
    piv = np.array([4, 2, 7, 11, 5])
    model_code = np.array(
        [[2002, 0], [1002, 0], [2001, 1001], [2002, 1002], [1001, 1001]]
    )
    x, y, theta = create_test_data()
    model = PolynomialNarmax(
        non_degree=2,
        n_terms=5,
        order_selection=True,
        n_info_values=5,
        info_criteria="aic",
        extended_least_squares=False,
        ylag=[1, 2],
        xlag=2,
        estimator="least_squares",
    )
    model.fit(x, y)
    # assert_array_equal(model.pivv, piv)
    assert_array_equal(model.final_model, model_code)


def test_fit_with_information_criteria():
    x, y, theta = create_test_data()
    model = PolynomialNarmax(
        non_degree=2,
        n_terms=15,
        order_selection=True,
        extended_least_squares=False,
    )
    model.fit(x, y)
    assert "info_values" in dir(model)


def test_fit_without_information_criteria():
    x, y, theta = create_test_data()
    model = PolynomialNarmax(
        non_degree=2,
        n_terms=15,
        extended_least_squares=False,
    )
    model.fit(x, y)
    assert "info_values" not in dir(model)
```

Рис. 3.13 вибірка методів з модулю тестування ПЗ

Робота програмного забезпечення організована за рахунок об’єктних та міжмодульних взаємодій. Також у програмі представлені такі класи:

ModelIdentifers – класс, що імплементуе вище вказаний інтерфейс і без посередньо займається рекурсивним пошуком оптимальної моделі. На кожному кроці модель ідентифікується з певним ступенем правої частини диференційного рівняння. Потім проводиться розрахунок середньої квадратичної похибки. За допомогою класу ErrorEstimator,

Цей клас для кожної ітерації ідентифікації моделі проводить її оцінку шляхом визначення квадратичної похибки прогнозу. Що визначаються через порівняння експериментальних результатів з даними отриманими на поточному кроці ідентифікації.

SmoothedFiniteDifference - виконує диференціацію, згладжуючи вхідні дані, а потім застосовуючи кінцевий метод різниці.

```
class SmoothedFiniteDifference(FiniteDifference):
    def __init__(self, smoother=savgol_filter, smoother_kws={}, **kwargs):
        super(SmoothedFiniteDifference, self).__init__(**kwargs)
        self.smoother = smoother
        self.smoother_kws = smoother_kws

        if smoother is savgol_filter:
            if "window_length" not in smoother_kws:
                self.smoother_kws["window_length"] = 11
            if "polyorder" not in smoother_kws:
                self.smoother_kws["polyorder"] = 3
            self.smoother_kws["axis"] = 0

    def _differentiate(self, x, t):
        """Apply finite difference method after smoothing."""
        x = self.smoother(x, **self.smoother_kws)
        return super(SmoothedFiniteDifference, self)._differentiate(x, t)
```

Рис. 3.14 SmoothedFiniteDifference

SINDerivative - клас обгортки для диференціації моделей. Цей клас призначений забезпечити ту саму функціональність, що і derivative.differentiation.dxdt з урахуванням специфіки ідентифікації

моделі. Цей клас також має метод differentiate, що використовується PySINDy.

```
class SINDyDerivative(BaseEstimator):
    def __init__(self, **kwargs):
        self.kwargs = kwargs

    def set_params(self, **params):
        if not params:
            # Simple optimization to gain speed (inspect is slow)
            return self
        else:
            self.kwargs.update(params)

        return self

    def get_params(self, deep=True):
        """Get parameters."""
        params = super().get_params(deep)

        if isinstance(self.kwargs, dict):
            params.update(self.kwargs)

        return params


    def _differentiate(self, x, t=1):
        if isinstance(t, (int, float)):
            if t < 0:
                raise ValueError("t must be a positive constant or an array")
            t = arange(x.shape[0]) * t

        return dxdt(x, t, axis=0, **self.kwargs)

    def __call__(self, x, t=1):
        x = validate_input(x, t=t)
        return self._differentiate(x, t)
```

Рис. 3.15 SINDerivative

Цей клас працює на основі класу Estimator, що відповідає за базову обробку моделі. Існує безліч моделей валідацій, таких як кореляційні тести валідності, перехресна перевірка, тестування крок-відповідь THOMSON (1996). У цій роботі перехресна перевірка проводилася з набором даних, відмінним від використовуваного оцінка параметрів та оцінюється шляхом обчислення коефіцієнта детермінації (R-квадрат). Слід зазначити, що R-квадрат у випадку нелінійної ідентифікації може статися від 0 до 1. Це трапляється, коли ідентифікація дуже погана і взагалі не представляє

систему. Перехресна перевірка проводится шляхом порівняння нового набору даних із прогнозованим вихід, що генерується в кожному горизонті прогнозування.

Перевірку можна також розділити на три типи щодо тривалості траєкторії прогнозування.

- Перевірка на один крок вперед складається із значень прогнозування на один крок вперед, обчислених з використанням минулих вихідних даних, тобто кожна траєкторія передбачення має довжину в одну точку.

- Перевірка на крок вперед складається з траєкторії прогнозування, обчисленої з використанням кількох точок минулих вихідних даних таким чином, що передбачувані значення обчислюються з передбаченими даними, але після певної кількості точок (горизонт прогнозування) вона використовує оригінальні дані знову, щоб перезапустити траєкторію передбачення, тому кожна траєкторія прогнозування має довжину k точок.

- Перевірка нескінченного горизонту складається з траєкторії прогнозування, обчисленої з використанням лише передбачуваних значень, за винятком вихідних точок, які використовують минулі вихідні дані, тобто. е., це рекурсивний розрахунок із лише початковою точкою залежно від вихідних даних. У цьому випадку траєкторія передбачення має нескінченну довжину. Обраним методом перевірки перехресна перевірка з k кроками вперед, оскільки запропонована методологія ідентифікації спрямована на цілі оптимізації та контролю, яким потрібна ідентифікована модель для представлення процесу в межах певного горизонту прогнозування.

BaseFeatureLibrary – клас інтерфейс, що описує базові можливості бібліотеки для зовнішнього використання. Змушує підкласи реалізовувати fit, transform, та метод get_feature_names.

```python
class BaseFeatureLibrary(TransformerMixin):
    def __init__(self, **kwargs):
        pass

    # Force subclasses to implement this
    @abc.abstractmethod
    def fit(self, X, y=None):

        raise NotImplementedError

    # Force subclasses to implement this
    @abc.abstractmethod
    def transform(self, X):
        raise NotImplementedError


    # Force subclasses to implement this
    @abc.abstractmethod
    def get_feature_names(self, input_features=None):
        raise NotImplementedError

    def __add__(self, other):
        return ConcatLibrary([self, other])

    @property
    def size(self):
        check_is_fitted(self)
        return self.n_output_features_
```

Рис. 3.16 BaseFeatureLibrary

ConcatLibrary – об’єднує кілька інтерфейсів в одну бібліотеку.. Приймає у себе набір класів, що потрібно обьеднати у один інтерфейс. Кількість вихідних функцій це сума чисел вихідних ознак для кожної з об'єднаних бібліотек.

```python
class ConcatLibrary(BaseFeatureLibrary):
    def __init__(self, libraries: list):
        super(ConcatLibrary, self).__init__()
        self.libraries_ = libraries

    def fit(self, X, y=None):
        _, n_features = check_array(X).shape
        self.n_input_features_ = n_features

        # First fit all libs provided below
        fitted_libs = [lib.fit(X, y) for lib in self.libraries_]

        # Calculate the sum of output features
        self.n_output_features_ = sum([lib.n_output_features_ for lib in fitted_libs])

        # Save fitted libs
        self.libraries_ = fitted_libs

        return self

    def transform(self, X):
        for lib in self.libraries_:
            check_is_fitted(lib)
        n_samples = X.shape[0]

        # preallocate matrix
        XP = np.zeros((n_samples, self.n_output_features_))

        current_feat = 0
        for lib in self.libraries_:

            # retrieve num features from lib
            lib_n_output_features = lib.n_output_features_

            start_feature_index = current_feat
            end_feature_index = start_feature_index + lib_n_output_features

            XP[:, start_feature_index:end_feature_index] = lib.transform(X)

            current_feat += lib_n_output_features

        return XP

    def get_feature_names(self, input_features=None):
        feature_names = list()
        for lib in self.libraries_:
            lib_feat_names = lib.get_feature_names(input_features)
            feature_names += lib_feat_names
        return feature_names
```

Рис. 3.17 ConcatLibrary

FiniteDifference – клас, що займається кінцевими різницевими похідними. На даний момент застосовуються лише методи скінченних різниць першого та другого порядку реалізовано.

Параметри:

order: int, 1 або 2, необов’язковий (за замовчуванням 2). Порядок використання методу скінченних різниць. Якщо 1, буде використана різниця вперед для першого замовлення. Якщо 2, буде використана різниця у центрі другого порядку

drop_endpoints: логічне значення, необов’язкове (за замовчуванням False). Незалежно від того, чи похідні обчислюються для кінцевих точок. Якщо значення False, для кінцевих точок буде встановлено значення np.nan.

```
class FiniteDifference(BaseDifferentiation):
    def __init__(self, order=2, drop_endpoints=False):
        if order <= 0 or not isinstance(order, int):
            raise ValueError("order must be a positive int")
        elif order > 2:
            raise NotImplementedError

        self.order = order
        self.drop_endpoints = drop_endpoints

    def _differentiate(self, x, t):
        if self.order == 1:
            return self._forward_difference(x, t)
        else:
            return self._centered_difference(x, t)

    def _forward_difference(self, x, t=1):
        x_dot = np.full_like(x, fill_value=np.nan)

        # Uniform timestep (assume t contains dt)
        if np.isscalar(t):
            x_dot[:-1, :] = (x[1:, :] - x[:-1, :]) / t
```

```
def _forward_difference(self, x, t=1):
    x_dot = np.full_like(x, fill_value=np.nan)

    # Uniform timestep (assume t contains dt)
    if np.isscalar(t):
        x_dot[:-1, :] = (x[1:, :] - x[:-1, :]) / t
        if not self.drop_endpoints:
            x_dot[-1, :] = (3 * x[-1, :] / 2 - 2 * x[-2, :] + x[-3, :] / 2) / t

    # Variable timestep
    else:
        t_diff = t[1:] - t[:-1]
        x_dot[:-1, :] = (x[1:, :] - x[:-1, :]) / t_diff[:, None]
        if not self.drop_endpoints:
            x_dot[-1, :] = (
                3 * x[-1, :] / 2 - 2 * x[-2, :] + x[-3, :] / 2
            ) / t_diff[-1]

    return x_dot

def _centered_difference(self, x, t=1):
    x_dot = np.full_like(x, fill_value=np.nan)

    # Uniform timestep (assume t contains dt)
    if np.isscalar(t):
```

```
    # Uniform timestep (assume t contains dt)
    if np.isscalar(t):
        x_dot[1:-1, :] = (x[2:, :] - x[:-2, :]) / (2 * t)
        if not self.drop_endpoints:
            x_dot[0, :] = (
                -11 / 6 * x[0, :] + 3 * x[1, :] - 3 / 2 * x[2, :] + x[3, :] / 3
            ) / t
            x_dot[-1, :] = (
                11 / 6 * x[-1, :] - 3 * x[-2, :] + 3 / 2 * x[-3, :] - x[-4, :] / 3
            ) / t

    # Variable timestep
    else:
        t_diff = t[2:] - t[:-2]
        x_dot[1:-1, :] = (x[2:, :] - x[:-2, :]) / t_diff[:, None]
        if not self.drop_endpoints:
            x_dot[0, :] = (
                -11 / 6 * x[0, :] + 3 * x[1, :] - 3 / 2 * x[2, :] + x[3, :] / 3
            ) / (t_diff[0] / 2)
            x_dot[-1, :] = (
                11 / 6 * x[-1, :] - 3 * x[-2, :] + 3 / 2 * x[-3, :] - x[-4, :] / 3
            ) / (t_diff[-1] / 2)
```

Рис. 3.18 FiniteDifference

SINDyOptimizer - клас обгортки для оптимізаторів / розріджених методів регресії, переданих в об'єкт SINDy.

Дозволяє визначати одноцільові регресори (тобто ті, чиї передбачення є одновимірними) виконати багатоцільову регресію (тобто передбачення є двовимірними).

Параметри:

optimizer: об'єкт оцінювача, регресор, який потрібно обернути, реалізуючи fit та predict.

unbias: логічне значення, необов'язкове (за замовчуванням True).

Воно визначає чи потрібно виконувати додатковий крок нерегульованої лінійної регресії до неупередженості коефіцієнтиа для ідентифікованої підтримки. Наприклад, якщо використовується оптимізатор = STLSQ (альфа = 0,1) ", тоді вивчені коефіцієнти будуть зміщені до 0 через регуляризацію L2. Встановлення unbias = True " ініціює додатковий крок, коли ненульове значення коефіцієнти, вивчені об'єктом оптимізатора, будуть оновлені за допомогою підходять нерегульовані мінімальні квадрати.

```
class SINDyOptimizer(BaseEstimator):
    def __init__(self, optimizer, unbias=True):
        if not hasattr(optimizer, "fit") or not callable(getattr(optimizer, "fit")):
            raise AttributeError("optimizer does not have a callable fit method")
        if not hasattr(optimizer, "predict") or not callable(
            getattr(optimizer, "predict")
        ):
            raise AttributeError("optimizer does not have a callable predict method")

        self.optimizer = optimizer
        self.unbias = unbias

    def fit(self, x, y):
        if len(y.shape) > 1 and y.shape[1] > 1:
            if not supports_multiple_targets(self.optimizer):
                self.optimizer = _MultiTargetLinearRegressor(self.optimizer)
        self.optimizer.fit(x, y)
        if not hasattr(self.optimizer, "coef_"):
            raise AttributeError("optimizer has no attribute coef_")
        self.ind_ = np.abs(self.coef_) > COEF_THRESHOLD

        if self.unbias:
            self._unbias(x, y)

        return self

    def _unbias(self, x, y):
        coef = np.zeros((y.shape[1], x.shape[1]))
        if hasattr(self.optimizer, "fit_intercept"):
            fit_intercept = self.optimizer.fit_intercept
        else:
            fit_intercept = False
```

```
        if hasattr(self.optimizer, "normalize"):
            normalize = self.optimizer.normalize
        else:
            normalize = False
        for i in range(self.ind_.shape[0]):
            if np.any(self.ind_[i]):
                coef[i, self.ind_[i]] = (
                    LinearRegression(fit_intercept=fit_intercept, normalize=normalize)
                    .fit(x[:, self.ind_[i]], y[:, i])
                    .coef_
                )
        if self.optimizer.coef_.ndim == 1:
            self.optimizer.coef_ = coef[0]
        else:
            self.optimizer.coef_ = coef

    def predict(self, x):
        prediction = self.optimizer.predict(x)
        if prediction.ndim == 1:
            return prediction[:, np.newaxis]
        else:
            return prediction

    @property
    def coef_(self):
        if self.optimizer.coef_.ndim == 1:
            return self.optimizer.coef_[np.newaxis, :]
        else:
            return self.optimizer.coef_
```

```
        for i in range(self.ind_.shape[0]):
            if np.any(self.ind_[i]):
                coef[i, self.ind_[i]] = (
                    LinearRegression(fit_intercept=fit_intercept, normalize=normalize)
                    .fit(x[:, self.ind_[i]], y[:, i])
                    .coef_
                )
        if self.optimizer.coef_.ndim == 1:
            self.optimizer.coef_ = coef[0]
        else:
            self.optimizer.coef_ = coef

    def predict(self, x):
        prediction = self.optimizer.predict(x)
        if prediction.ndim == 1:
            return prediction[:, np.newaxis]
        else:
            return prediction

    @property
    def coef_(self):
        if self.optimizer.coef_.ndim == 1:
            return self.optimizer.coef_[np.newaxis, :]
```

Рис. 3.19 SINDyOptimizer

Клас MatplotlibWidget для роботи з графікою, успадковує PyQt4.QtGui.QWidget та matplotlib. backend_bases.FigureCanvasBase

Параметри:

parent : батьківський віджет;

title : заголовок малюнка;

xlabel : Мітка осі X;

ylabel : Мітка осі Y;

xlim : обмеження по осі X ([хв., макс.]);

ylim : Обмеження по осі Y ([хв., макс.]);

xscale ('лінійний'): шкала осі X;

yscale ('лінійний'): шкала осі Y;

width: ширина в дюймах;

висота : висота в дюймах;

dpi : роздільна здатність у dpi;

hold (False): якщо False, цифра буде очищатися кожного разу, коли буде викликаний графік

```
class MatplotlibWidget(Canvas):
    def __init__(self, parent=None, title='', xlabel='', ylabel='',
                 xlim=None, ylim=None, xscale='linear', yscale='linear',
                 width=4, height=3, dpi=100, hold=False):
        self.figure = Figure(figsize=(width, height), dpi=dpi)
        self.axes = self.figure.add_subplot(111)
        self.axes.set_title(title)
        self.axes.set_xlabel(xlabel)
        self.axes.set_ylabel(ylabel)
        if xscale is not None:
            self.axes.set_xscale(xscale)
        if yscale is not None:
            self.axes.set_yscale(yscale)
        if xlim is not None:
            self.axes.set_xlim(*xlim)
        if ylim is not None:
            self.axes.set_ylim(*ylim)
        self.axes.hold(hold)

        Canvas.__init__(self, self.figure)
        self.setParent(parent)

        Canvas.setSizePolicy(self, QSizePolicy.Expanding, QSizePolicy.Expanding)
        Canvas.updateGeometry(self)

    def sizeHint(self):
        w, h = self.get_width_height()
        return QSize(w, h)

    def minimumSizeHint(self):
        return QSize(10, 10)
```

Рис. 3.20 MatplotlibWidget

Висновки до розділу:

Поданий алгоритм математичного моделювання за представленим методом структурно параметричної ідентифікації NARMAX моделі.

Представлена порівняльна характеристика стеків технологій розробки. Обґрунтовано вибір стеку технологій розробки. Та представлений огляд вибраної мови програмування та бібліотек. Були обрані мова програмування Python разом з її бібліотеками Mathplotlib та PyQt4. Для зберігання даних використовуються NoSql бази даних.

Запропонований програмний алгоритм структурно параметричної ідентифікації моделі за алгоритмами поданими у другому розділі. Подана функціональна структура програмного забезпечення за описаними алгоритмами. Також описана структура основних класів та реалізацій програмного забезпечення.

За структурою створено програмне забезпечення, що містить у собі повну реалізацію алгоритму структурно-параметричну ідентифікацію NARMAX моделей за поданими алгоритмом. Тобто виконує асиміляцію та фільтрацію вхідних даних, за алгоритмом знаходить оптимальний набір регрессорів, для отриманого набору даних.

Створено API для взаємодії з іншими додатками, що хочуть використовувати імплементований функціонал. Розроблено графічний інтерфейс для відображення отриманих моделей та прогнозу, отриманого з їх використанням.

4. АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У ході програмного моделювання за допомогою викладених алгоритмів були отримані декілька моделей, за рахунок використання різних часових проміжків для тренування моделі. Кожна з моделей була протестована на експериментальних значеннях. Проведені тести під час яких визначався прогноз для різних розмір горизонту прогнозу: 1, 2,4 та 6 годин. Розглянемо одну з отриманих моделей.

$$y_i = 1.36\, y_{i-1} - 4.99\, u_{i-1} - 0.18\, y_{i-2}\, u_{i-1} - 0.57\, y_{i-4} -$$

$$-1.43\, u_{i-4}\, u_{i-6} - 0.75\, y_{i-2} + 0.53\, y_{i-3} + 0.1\, y_{i-3}\, u_{i-1} +$$

$$+0.36\, y_{i-5} + 0.92\, u_{i-6}\, u_{i-7} + 2.71\, u_{i-2} + 0.08\, y_{i-3}\, u_{i-2} +$$

$$+0.78\, u_{i-2}\, u_{i-5} - 0.91\, u_{i-5} + 0.25\, u_{i-7}\, u_{i-12}$$

Рис. 4.1 Модель

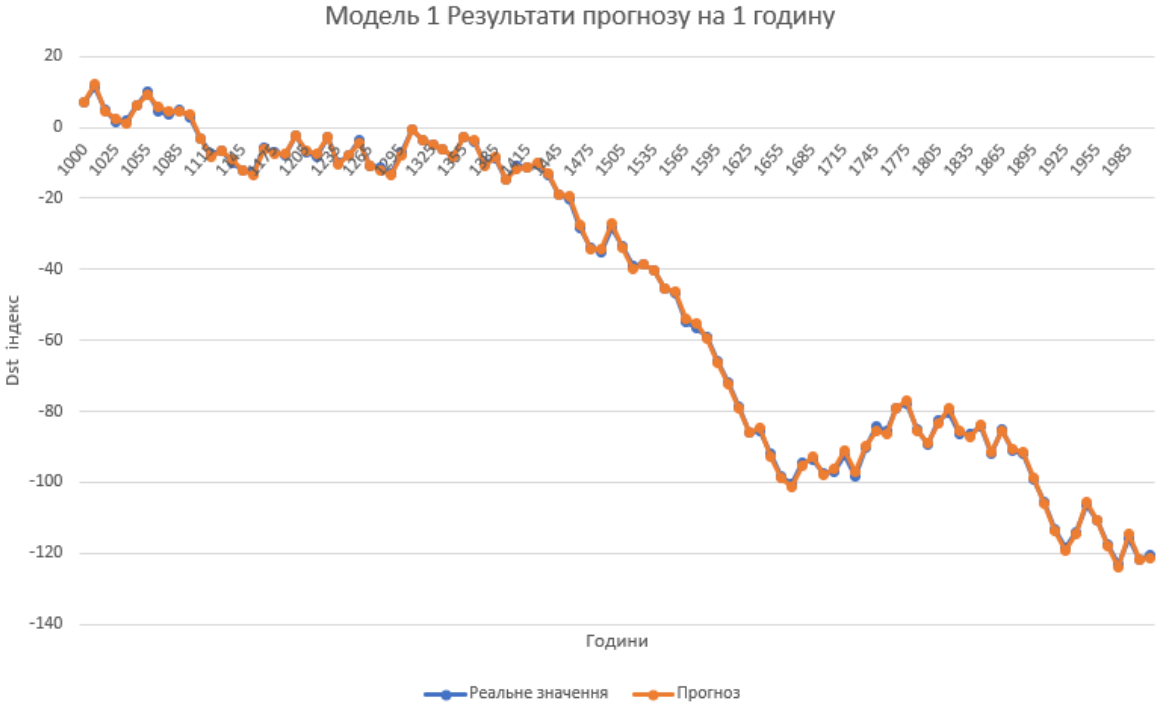

Рис. 4.2 Результат прогнозу на 1 годину

Рис. 4.3 Результат прогнозу на 1 годину

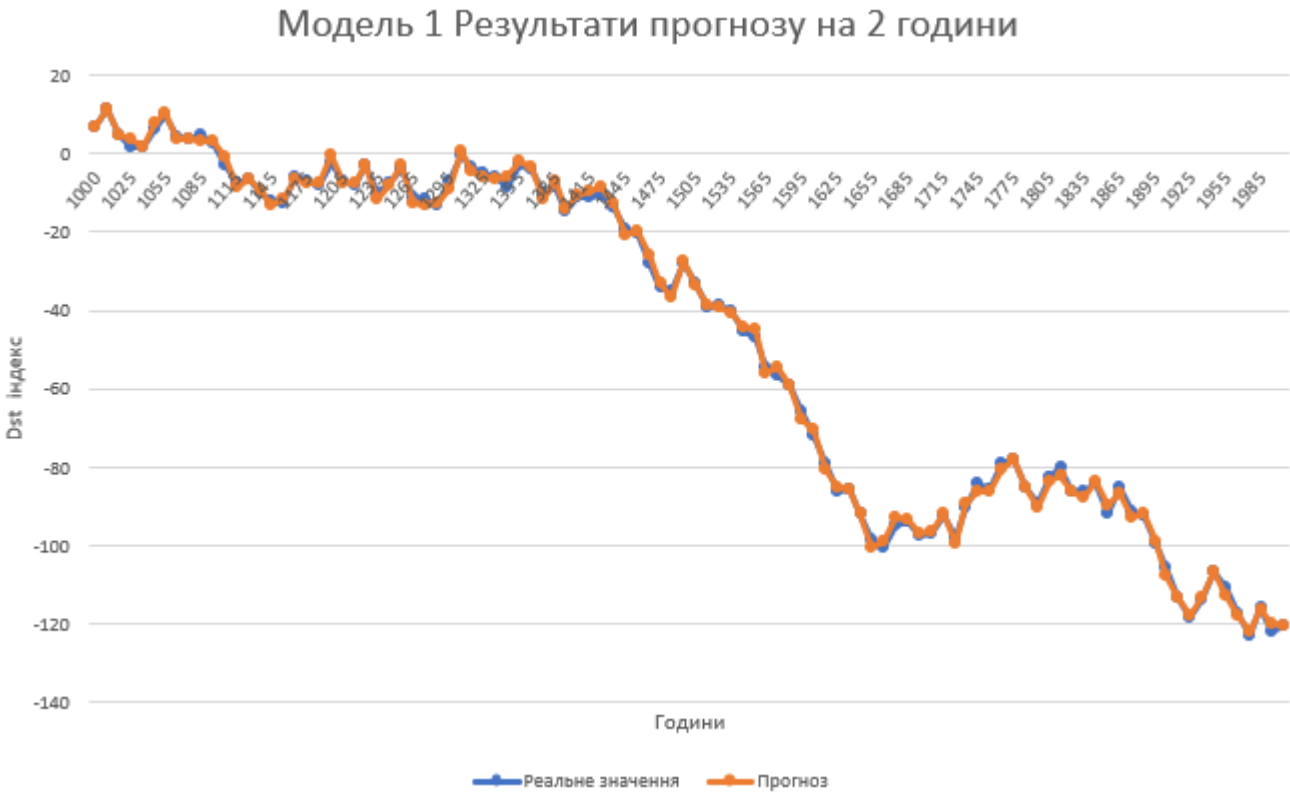

Рис. 4.4 Результат прогнозу на 2 години

Рис. 4.5 Результат прогнозу на 2 години

Як видно з графіків похибка поступово зростає, зміщуючи лінію прогнозу відносно експериментальних даних.

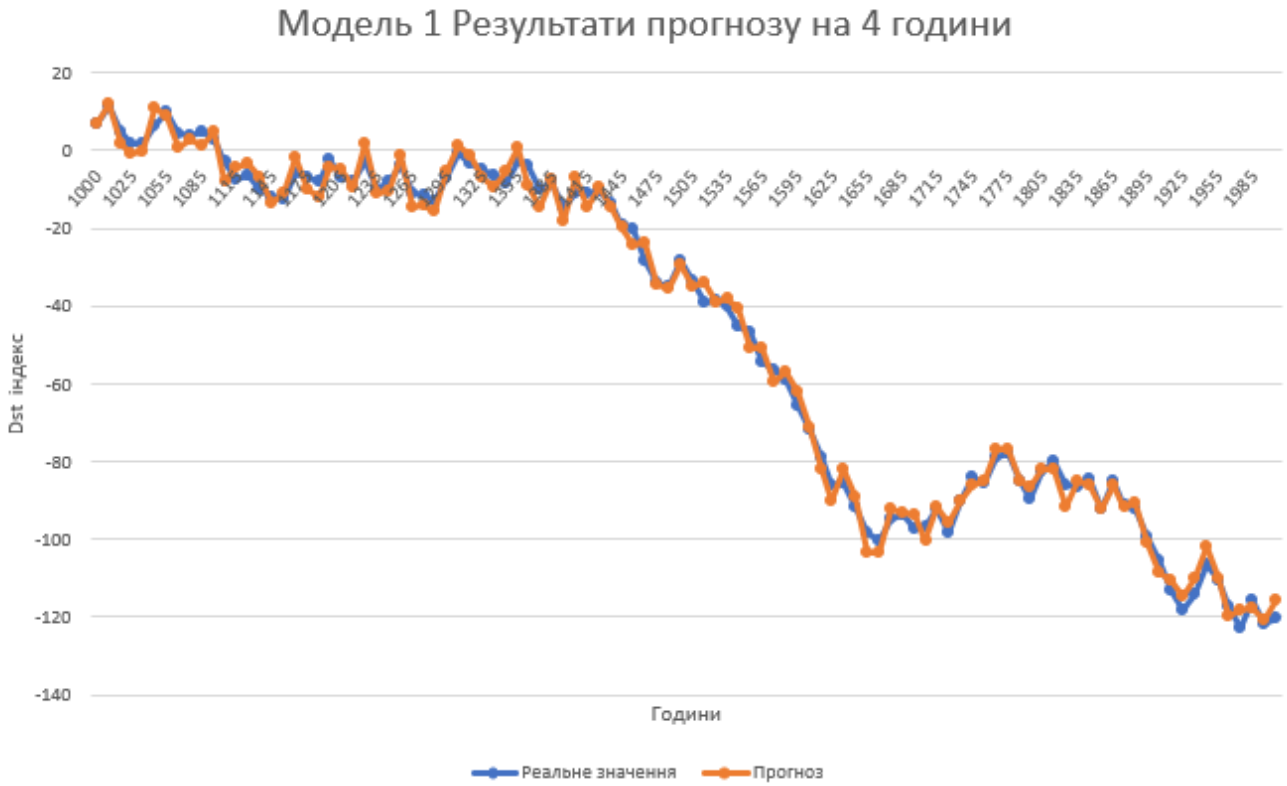

Рис. 4.5 Результат прогнозу на 4 години

Рис. 4.6 Результат прогнозу на 4 години

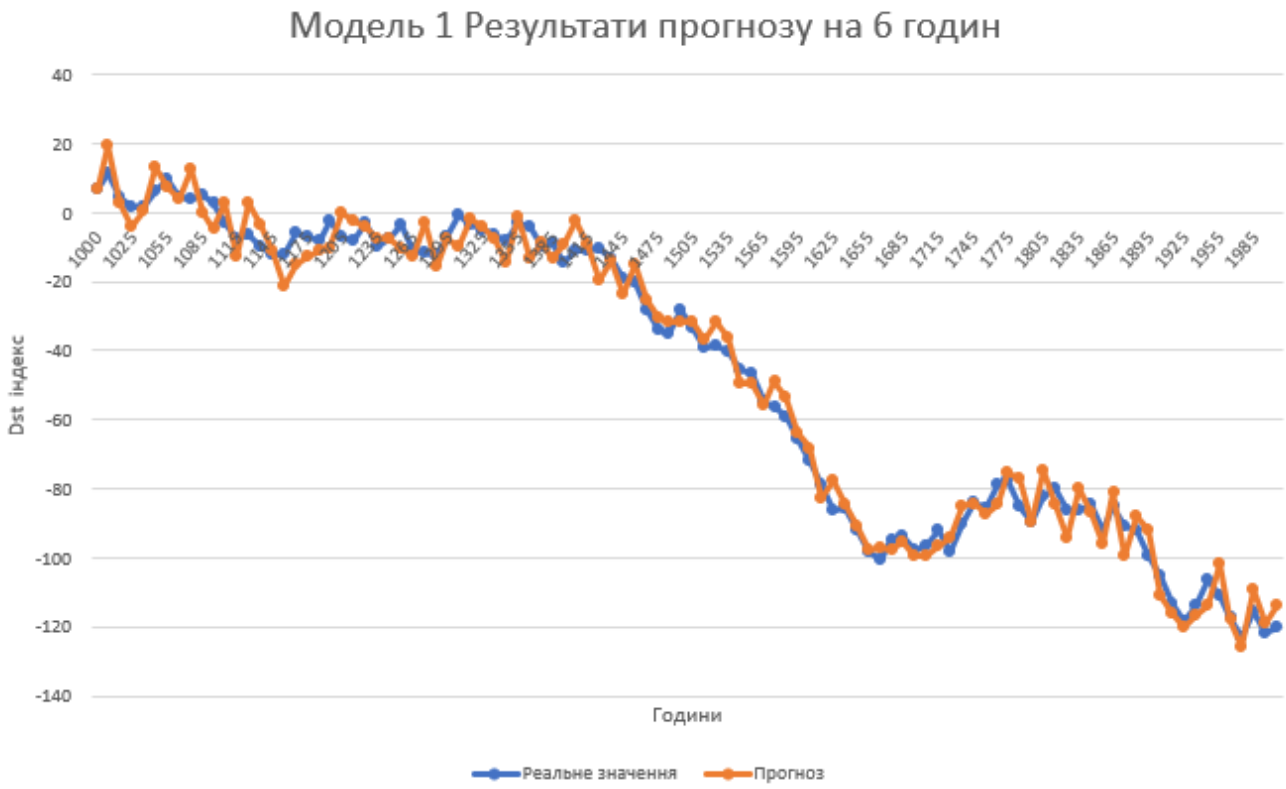

Рис. 4.7 Результат прогнозу на 6 годин

Рис. 4.8 Результат прогнозу на 6 годин

Рис. 4.9 Графік зміни похибки прогнозу від зміни горизонту прогнозу

На інших часових даних була ідентифікована так модель Рис.4.9. Для неї були також отримані результати по прогнозуванню.

$$y(k) = a_1 y(k-1) + a_2 y(k-2) + a_3 y(k-3) + a_4 y(k-4) + a_5 y(k-5) + a_6 u(k-1) +$$
$$+ a_7 u(k-2) + a_8 u(k-3) + a_9 u(k-4) + a_{10} u(k-5) + a_{11} u(k-6) +$$
$$+ a_{12} u(k-4) u(k-5) + a_{13} y(k-1) u(k-4) + a_{14} u(k-5) u(k-3),$$

Рис. 4.10 Модель 2

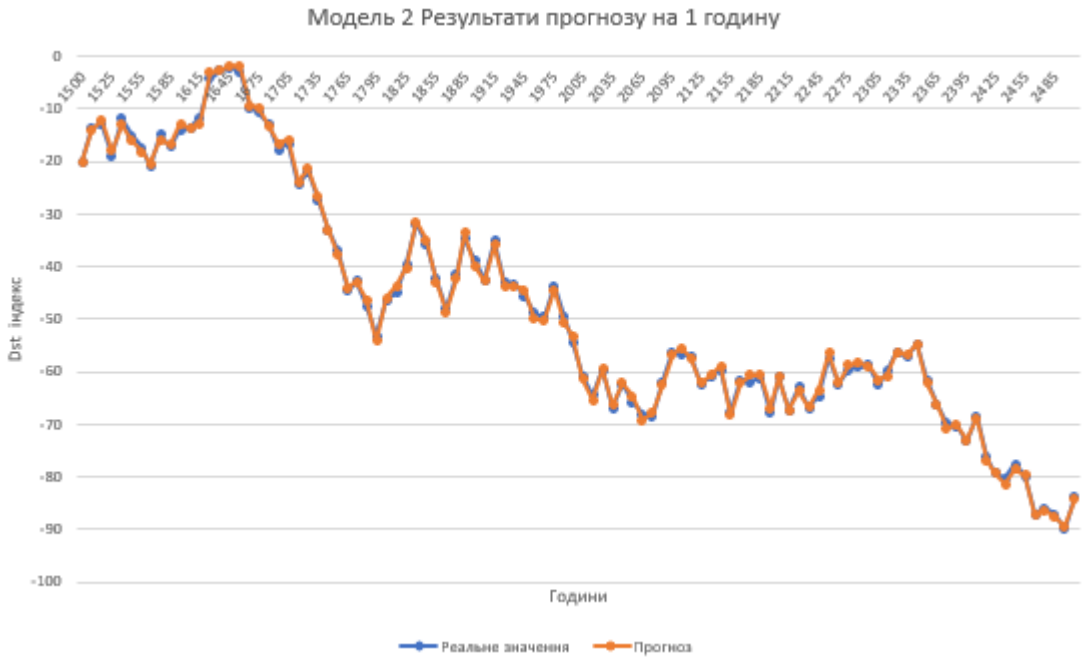

Рис. 4.11 Прогноз на 1 годину

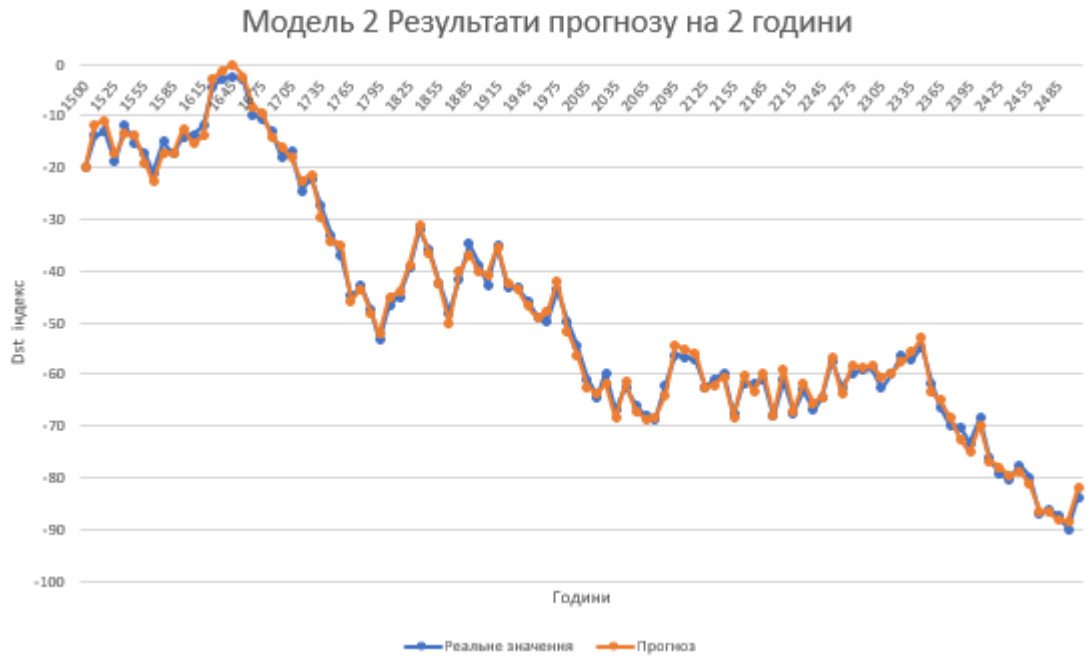

Рис. 4.12 Прогноз на 2 години

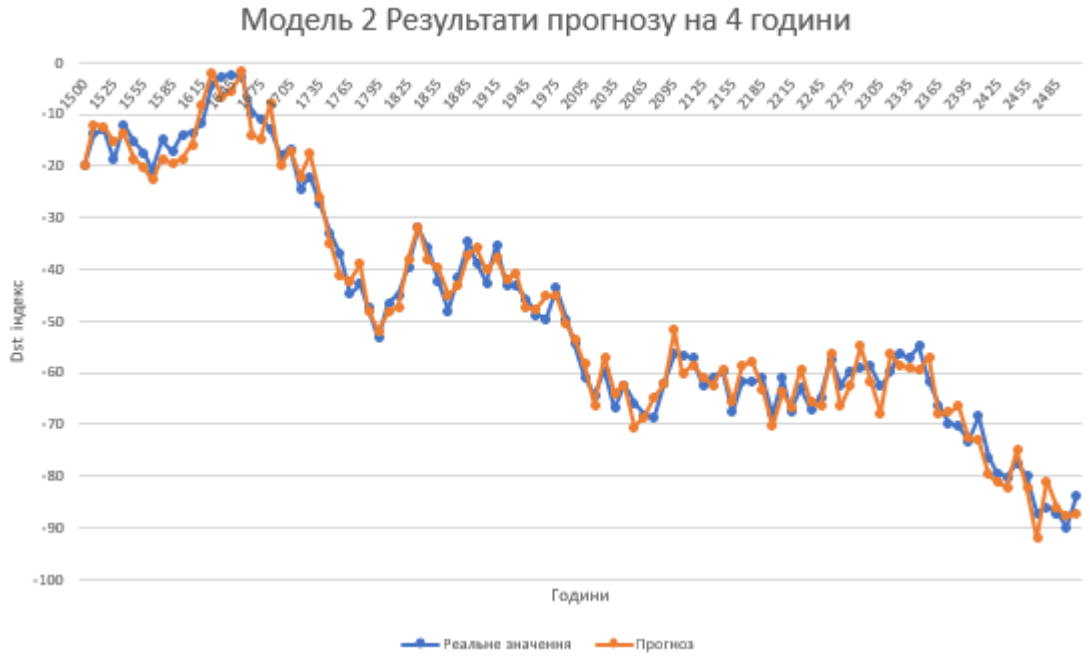

Рис. 4.13 Прогноз на 4 години

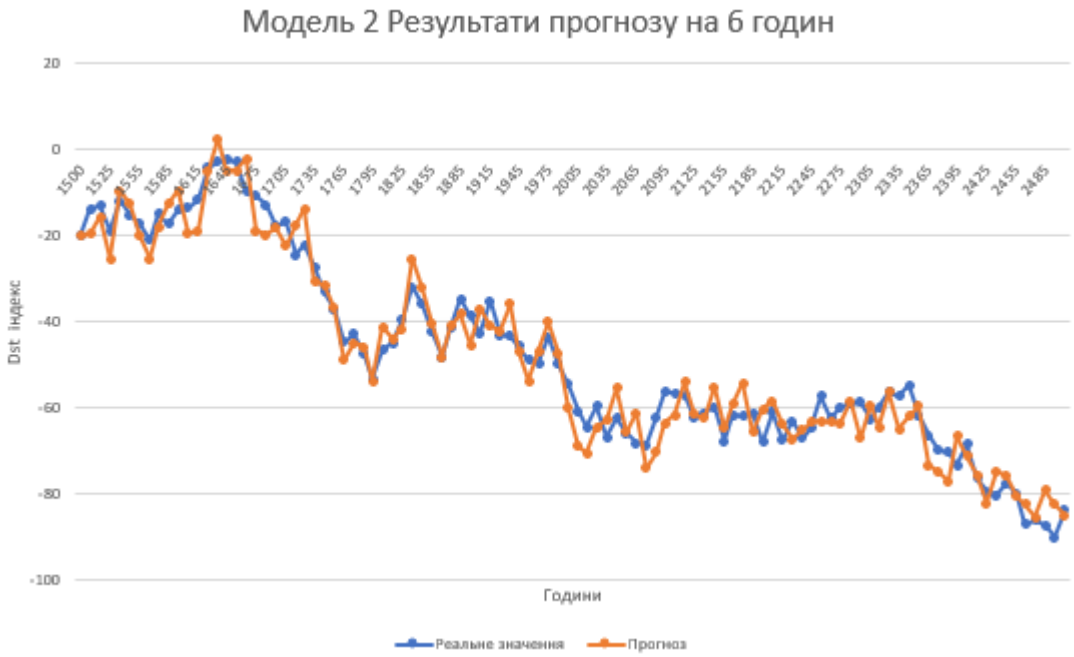

Рис. 4.14 Прогноз на 6 годин

Отже, очевидно, що зі збільшенням горизонту прогнозу зростає абсолютна похибка прогнозу, причому цей ріст має експоненціальний характер. У ході програмного моделювання на різних часових проміжках, ідентифіковано дані моделі прогнозування: Перед ідентифікацією моделі по набору даних за 1995 - 2010 роки застосовано процедуру відновлення пропущених спостережень. Описано результати прогнозування Dst-індексу з використанням білінійної динамічної системи експериментальних даних спостережень про швидкість сонячного вітру та даними про стан південної компоненти магнітного поля.

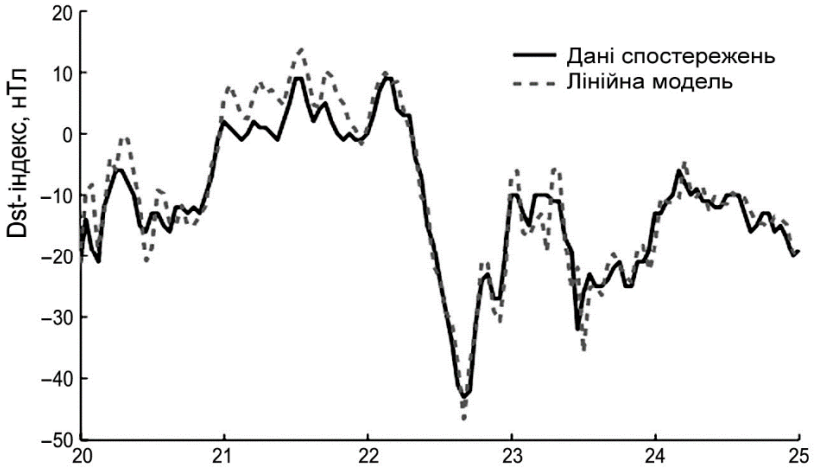

Рисунок 4.10 Лінійне прогнозування Dst-індексу. (NARX-модель)

Для порівняння з альтернативними моделями прогнозування розглянуто нелінійну модель авторегресії з екзогенними (рівноправними) входами (NARX-модель), яка є окремим випадком більш загальних структур, що містять компоненти змінного середнього і відомих як NARMAX-моделі, які характеризуються більшою гнучкістю і потенційно більш високою точністю.

Рисунок 4.15 Білінійне прогнозування Dst-індексу на годину наперед.

Теоретично на основі навіть на основі багатошарового персептрона, який містить зворотні зв’язки, може бути побудована прогнозуюча NARMAX-модель, однак навчання цієї системи характеризується низькою

швидкістю збіжності, неможливістю роботи з нестандартними сигналами сигналами необхідністю використання налаштувань по часовим інтервалам. На рисунку 4.16 приведено результати порівняння лінійної та нелінійної моделей. Із рисунку видно, що білінійна модель досить точно прогнозує поведінку D_{st}-індексу у порівнянні з лінійними моделями.

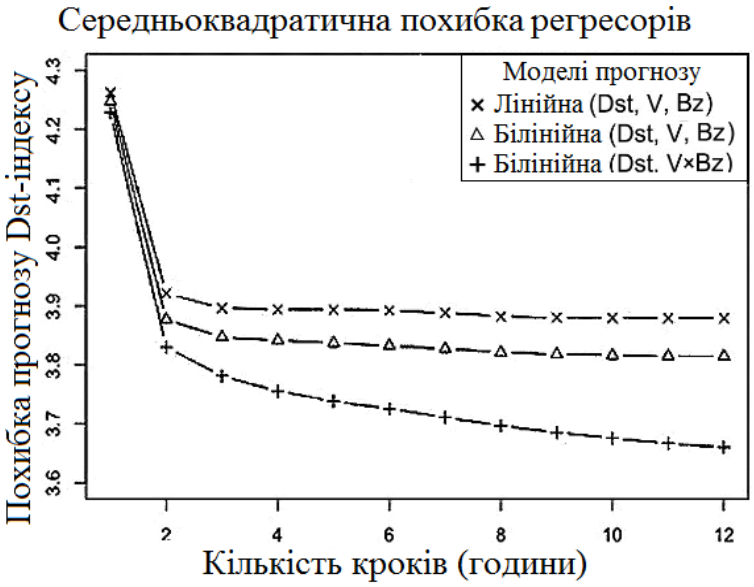

Рис. 4.16 Результат порівняння похибки зміни квадратичної похибки для різного числа регрессорів

На рис. 4.2 приведено чисельні результати щодо білінійного прогнозування D_{st}- індексу на одну годину наперед. Із рисунку видно, що білінійна модель досить точно прогнозує поведінку D_{st}-індексу. На рис. 4.16 приведено результати порівняння лінійного прогнозування D_{st}-індексу на одну годину наперед з експериментальними даними. Із рисунку видно, що лінійна модель прогнозує поведінку D_{st}-індексу з більш великою похибкою у порівнянні з білінійною.

Отримані моделі забезпечую більш точне порівняно, як з лінійними, так і з білінійними(з двома вхідними параметрами), точне прогнозування D_{st}- індексу. Приведено чисельні результати гарантованого прогнозування D_{st}-індексу з використанням супутникових даних спостережень параметрів сонячного вітру.

Висновки до розділу:

Подано отримані за допомогою програмного забезпечення NARMAX моделі. Показані результати прогнозу у порівнянні с реальними даними за прогнозовані проміжки часу. Проведено прогнозування за допомогою цих моделей. Подані результати прогнозів з різним горизонтом прогнозів, тобто показані результати, що відображають роботу моделі при прогнозуванні на різні відрізки часу. А саме на 1, 2, 4 та 6 годин. Наведена порівняльна характеристика цих прогнозів. Проаналізована поведінка значень середньою абсолютної похибки, на різних часових проміжках.

Також представлені результати прогнозування білінійної моделі, без використання нового підходу до структурно-параметричної ідентифікації та результати прогнозування лінійної моделі за той самий проміжок часу, що і прогнози, що базуються на моделях, отриманих за допомогою нового методу структурно-параметричної ідентифікації.

Наведена порівняльна характеристика якості прогнозу на основі кількості включених регрессорів. Тобто, відношення якості прогнозу, вираженого середньою квадратичною похибкою, до кількості ітерацій на яких ідентифікувалася модель. Як показано на графіках, квадратична похибка зменшується зі збільшенням числа ітерацій, проте розроблена за допомогою нового методу білінійна модель має найменше з представлених значення похибки. Це дозволяє стверджувати, про ефективність вибраного методу ідентифікації.

Висновки:

Запропоновано новий метод структурно-параметричної ідентифікації NARMAX моделей.

Розроблена структура NARMAX модель типу «вхід-вихід», що базується на описаному методі та включає у себе прогнозування D_{st} індексу з використанням даних про швидкість сонячного вітру та південну компоненту магнітного поля.

Запропоновано та обґрунтована структура програмного забезпечення для прогнозування космічної погоди на основі викладених методів. За цією структурою розроблено програмне забезпечення на основі Python, що дозволяє прогнозувати D_{st} індексу у реальному часі з мінімальною квадратичною похибкою.

За описаною структурою, розроблено програмне забезпечення, для автоматизації процесу структурно-параметричної ідентифікації NARMAX моделей, зберігання та обробки вхідних даних та проведення прогнозів з використанням готових моделей. За допомогою алгоритму отримані результуючі моделі. Також було проведено тестування моделей. Експериментальні дані підтверджують, що розроблене програмне забезпечення дозволяє отримувати моделі, що дозволяють проводити прогнозування з меншою похибкою на більшому проміжку часу.

Побудована чисельна модель поведінки D_{st}-індексу в часі. Встановлено, що ця модель може дати обґрунтований прогноз поведінки D_{st}-індексу на 5 - 6 годин наперед, і тому може бути використана для передбачення геомагнітних бур.

У подальшому програмне забезпечення може бути використано для створення веб-додатку для передбачення космічної погоди у реальному часі.

Список використаних джерел

1. Prediction of Geospace Radiation Environment and solar wind parameters https://cordis.europa.eu/article/id/245776-improved-space-weather-forecasting-protects-vital-infrastructure/fr

2. *Семенив О. В., Яценко В. О., Сидоренко В. И., Черемных О. К. и др*. Оптимизационный подход к прогнозированию космической погоды // Проблеми управленния и информатики. — 2008. — № 4. — С. 115—130.

3. Forecasting space weather by Dimitris Vassiliadis, Volker Bothmer, Ioannis A. Daglis

4. Retes, P.F.L. and Aguirre, L.A. (2019) 'NARMAX model identification using a randomised approach', Int. J. Modelling, Identification and Control, Vol. 31, No. 3, pp.205–216. DOI: 10.1504/IJMIC.2019.098779

5. A Methodology For Identification Of Narmax Models Applied To Diesel Engines 1 Gianluca Zito, Ioan Dor´e Landau ∗

6. Balikhin M., Bates I., Walker S. N. Identification of linear and nonlinear processes in space plasma turbulence // Adv. Space. – 2001. – 28. – P. 787–800.

7. Семенив О. В., Яценко В. О., Сидоренко В. И., Черемных О. К. и др. Оптимизационный подход к прогнозированию космической погоды // Проблеми управленния и информатики. — 2008. — № 4. — С. 115—130.

8. Baker D. N., Klimas A. J., McPherron R. L., Buchner J. The evolution from weak to strong geomagnetic activity: an interpretation in terms of deterministic chaos // J. Geophys. — 1990. — 17 (1). — P. 41—44.

9. V. Yatsenko. The influence of the free space environment on the superlight-weight thermal protection system: conception, methods, and risk analysis.- Automatic Welding, Paton Publishing House.-2016.-P.9 (in print).

10. V. Yatsenko, O. Cheremnykh. Kp index forecasting with robust dynamic models. - Journal “Automation and Information”, Kiev, Ukraine, 2016.-12 P. (in print)