МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова
Навчально науковий інститут автоматики та електротехніки

Кафедра комп'ютеризованих систем управління

«Допущений до захисту»
Завідувач кафедри
_____**Черно О.О.**
«12» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка інформаційно-обчислювального комплексу для моделювання руху триланкового робота-маніпулятора

Виконав: студент 4341 групи
_____**Мельникова О.Є.**
(підпис)

Керівник роботи:
доцент кафедри КСУ, к.т.н., доцент
(посада, науковий ступінь, вчене звання)
_____ **Герасін О.С.**
(підпис)

Миколаїв – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально науковий інститут автоматики та електротехніки

Кафедра комп'ютеризованих систем управління
Рівень вищої освіти **перший (бакалаврський)**
Спеціальність **151 – «Автоматизація та комп'ютерно-інтегровані технології»**
ООП «Комп'ютеризовані системи управління та автоматика»

ЗАТВЕРДЖУЮ
Гарант освітньої програми
_____ **Вінниченко І.Л.**
«29» квітня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту *Мельниковій Олені Євгенівні*
(прізвище, ім'я, по батькові)

1. Тема роботи: *Розробка інформаційно-обчислювального комплексу для моделювання руху триланкового робота-маніпулятора*

Керівник роботи *Герасін Олександр Сергійович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу № 286-уч від «22» квітня 2026 року

2. Термін подання роботи: *12.06.2026 р.*

3. Вихідні дані по роботі: *розробити інформаційно-обчислювальний комплекс для моделювання руху триланкового робота-маніпулятора на базі Matlab з функціями розв'язання прямої та зворотної задачі кінематики і графічним інтерфейсом для візуалізації результатів.*

4. Перелік питань, що належать до розробки (найменування розділів)
Аналіз методів математичного моделювання роботів-маніпуляторів.
Математична модель руху триланкового маніпулятора.
Інформаційно-обчислювальний комплекс для моделювання руху робота-маніпулятора.
Дослідження та аналіз результатів моделювання маніпулятора.
Охорона праці.

5. Перелік презентаційних матеріалів (з точним зазначенням обов'язкових креслень)
Автоматизація виробництва як рушійна сила сучасної промисловості.
Актуальність, мета та задачі роботи.
Огляд конструктивних типів роботів-маніпуляторів.
Кінематичний аналіз триланкового робота-маніпулятора.
Розв'язання прямої задачі кінематики.
Методи розв'язання зворотної задачі кінематики.
Гібридна стратегія розв'язання ЗЗК: градієнтний спуск та геометричний метод.
Модифікація методу градієнтного спуску: підбір ітераційного коефіцієнта.
Блок-схема роботи підфункції методу градієнтного спуску.
Блок-схема роботи підфункції геометричного (аналітичного) методу.
Блок-схема гібридного алгоритму розв'язання ЗЗК.

Сингулярні конфігурації триланкового маніпулятора.
Планування траєкторії на основі поліноміальної інтерполяції.
Вибір програмного середовища для моделювання.
Архітектура інформаційно-обчислювального комплексу.
Загальна блок-схема функціонування інформаційно-обчислювального комплексу.
Графічний інтерфейс та демонстрація роботи інформаційно-обчислювального комплексу.
Аналіз точності розв'язання ЗЗК.
Вплив ітераційного коефіцієнта на збіжність алгоритму.
Просторова структура збіжності алгоритму в робочій області.
Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*Герасін О.С., доцент каф. КСУ*	*16.02.2026*	*01.06.2026*

7. Дата видачі завдання: «16» лютого 2026 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів проектування	Термін виконання етапів роботи	Примітка
1.	*Аналіз методів математичного моделювання роботів-маніпуляторів.*	*20.03.2026*	*Виконано*
2.	*Математична модель руху триланкового маніпулятора.*	*13.04.2026*	*Виконано*
3.	*Інформаційно-обчислювальний комплекс для моделювання руху робота-маніпулятора.*	*01.05.2026*	*Виконано*
4.	*Дослідження та аналіз результатів моделювання маніпулятора.*	*26.05.2026*	*Виконано*
5.	*Охорона праці.*	*01.05.2026*	*Виконано*
6.	*Оформлення пояснювальної записки кваліфікаційної роботи*	*05.06.2026*	*Виконано*

Студент _____ Мельникова О.Є.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Герасін О.С.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розробці інформаційно-обчислювального комплексу для моделювання руху триланкового робота-маніпулятора.

Розроблено загальну математичну модель маніпулятора на основі параметрів Денавіта-Хартенберга, розв'язано пряму та зворотну задачі кінематики, проведено аналіз сингулярних конфігурацій та визначено робочий простір. Для розв'язання зворотної задачі кінематики реалізовано два методи: ітераційний алгоритм градієнтного спуску з адаптивним підбором ітераційного коефіцієнта на основі сумарної довжини ланок та геометричний (аналітичний) метод як резервний. Розроблено алгоритм планування траєкторії на основі полінома 5-го ступеня (квінтичного полінома), що забезпечує безперервність положення, швидкості та прискорення. Створено програмний застосунок з графічним інтерфейсом у середовищі MATLAB Robotics Toolbox з можливістю зміни конфігураційних характеристик маніпулятора (довжин ланок та допустимих діапазонів руху сполучень), візуалізацією кінематики, анімацією руху та побудовою графіків кінематичних характеристик.

Проведена валідація алгоритму на наборі з 48139 точок робочого простору для діапазону сумарних довжин ланок 0,3-20,1 м підтвердила його ефективність: при оптимальному значенні ітераційного коефіцієнта частка недосяжних точок не перевищує 12 %, тоді як при неоптимальному – зростає до 43-49 %. Встановлено залежність ефективності алгоритму від форми робочого простору та виявлено зони уповільненої збіжності поблизу сингулярних конфігурацій.

Ключові слова: кінематика маніпулятора, інформаційно-обчислювальний комплекс, пряма і зворотна задачі кінематики, траєкторія руху, метод градієнтного спуску.

ABSTRACT

This qualification thesis is devoted to the development of an information and computing complex for simulating the motion of a three-link robot manipulator.

A general mathematical model of the manipulator based on the Denavit-Hartenberg parameters has been developed. Forward and inverse kinematics problems have been solved, singular configurations have been analyzed, and the workspace has been determined. Two approaches to solving the inverse kinematics problem have been implemented: an iterative gradient descent algorithm with adaptive selection of the iteration gain based on the total link length, and a geometric (analytical) method used as a backup solution. A trajectory planning algorithm based on a fifth-order (quintic) polynomial has been developed to ensure continuity of position, velocity, and acceleration. A software application with a graphical user interface has been created in the MATLAB Robotics Toolbox environment, providing the ability to modify manipulator configuration parameters (link lengths and allowable joint motion ranges), visualize kinematics, animate motion, and generate plots of kinematic characteristics.

Validation of the proposed algorithm on a dataset of 48,139 workspace points for a total link length range of 0.3-20.1 m confirmed its effectiveness. At the optimal value of the iteration gain, the proportion of unreachable points does not exceed 12 %, whereas with a non-optimal gain it increases to 43-49 %. The dependence of the algorithm efficiency on the workspace shape has been established, and regions of slow convergence near singular configurations have been identified.

Keywords: manipulator kinematics, information and computing complex, forward and inverse kinematics, manipulator trajectory, gradient descent method.

ЗМІСТ

ВСТУП

Сучасні технологічні процеси в промисловості та науці характеризуються зростаючою потребою в автоматизації виробництва. Серед різноманітних засобів автоматизації особливе місце займають роботи-маніпулятори – багатоланкові механічні системи, здатні виконувати широкий спектр задач з високою точністю, швидкістю та надійністю. Упродовж останніх десятиліть спостерігається стійка тенденція їх застосування не лише у промисловому виробництві, а й у медицині, будівництві, агропромисловості та інших галузях.

Незважаючи на появу нових типів маніпуляторів – колаборативних роботів, дельта-роботів, SCARA-систем, – саме послідовні антропоморфні роботи-маніпулятори продовжують займати провідне місце серед промислових робототехнічних систем. Завдяки універсальній кінематичній структурі, значній робочій області та можливості реалізації складних просторових рухів вони залишаються основним інструментом автоматизації у зварюванні, фарбуванні, складанні та переміщенні деталей.

Ключовим етапом при проектуванні таких систем є кінематичний аналіз, який дозволяє оцінити коректність конструкції до її фізичної реалізації. Водночас кінематичні розрахунки для багатоланкових маніпуляторів ускладнюються через нелінійність і громіздкість обчислень, що обмежує можливість дослідження поведінки робота в реальному часі. Це зумовлює потребу в розробці спеціалізованих програмних засобів для автоматизації розрахунків і моделювання.

Метою роботи є розробка інформаційно-обчислювального комплексу для автоматизованого розрахунку кінематики триланкового робота-маніпулятора за допомогою пакету MATLAB Robotics Toolbox, що включає моделювання прямої та зворотної задач кінематики, планування траєкторії руху робочого органу та перевірку коректності роботи реалізованих алгоритмів.

З метою досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати існуючі методи кінематичного аналізу маніпуляторів та методи планування траєкторій руху робочого органу;
- розробити математичну модель триланкового маніпулятора на основі параметрів Денавіта-Хартенберга та вирішити пряму і зворотну задачі кінематики;
- виконати аналіз сингулярних конфігурацій маніпулятора та дослідити їх вплив на збіжність ітераційного алгоритму;
- здійснити планування траєкторії руху робочого органу маніпулятора для досягнення заданого положення на основі квінтичного полінома;
- розробити інформаційно-обчислювальний комплекс з графічним інтерфейсом користувача для автоматизації розрахунків;
- провести модельні дослідження точності розв’язання зворотної задачі кінематики та дослідити стійкість алгоритму за різних геометрій робочого простору.

Об’єктом дослідження є кінематичні процеси переміщення триланкового робота-маніпулятора з послідовною структурою ланок.

Предметом дослідження є моделі, методи та алгоритми розв’язання прямої і зворотної задач кінематики, планування траєкторії руху робочого органу та їх програмна реалізація у вигляді інформаційно-обчислювального комплексу.

Практична цінність отриманих результатів полягає у створенні програмного застосунку, який автоматизує повний цикл кінематичного аналізу маніпулятора – від введення геометричних параметрів до анімації руху та побудови графіків кінематичних характеристик і може бути використаний на ранніх етапах проєктування робототехнічних систем для тестування конструктивних рішень, аналізу робочої області та оцінювання кінематичних властивостей маніпуляторів.

РОЗДІЛ 1

АНАЛІЗ МЕТОДІВ МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ РОБОТІВ-МАНІПУЛЯТОРІВ

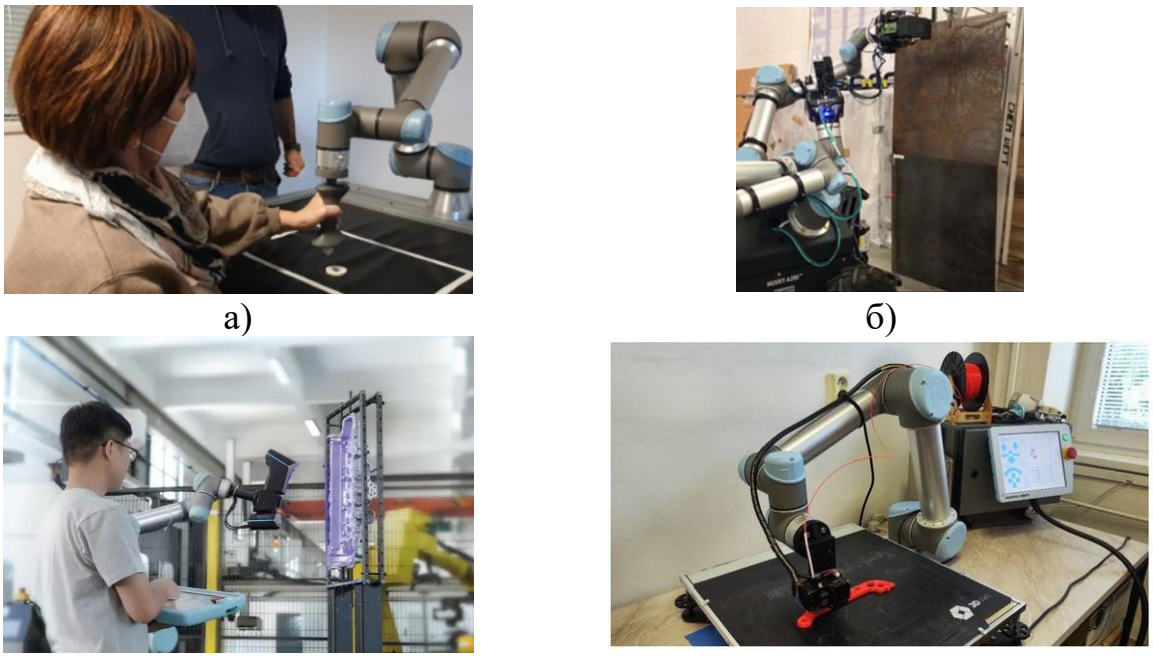
1.1. Сучасні тенденції розвитку робототехнічних систем і застосування маніпуляторів

Робототехнічні системи охоплюють широкий спектр технічних об’єктів – від найпростіших роботизованих пристроїв до складних автоматизованих виробничих комплексів і механізмів [1]. Одним із найбільш поширених класів таких систем є роботи-маніпулятори, які широко застосовуються для автоматизації технологічних процесів, що потребують високої точності позиціонування, повторюваності рухів і можливості безперервної роботи.

Конструктивною особливістю роботів цього типу є багатоланкова механічна система [2], призначена для переміщення робочого органу у просторі та виконання різноманітних технологічних операцій. Маніпуляційна структура робота значною мірою визначає техніко-експлуатаційні можливості системи та забезпечує низку важливих переваг, зокрема:

- універсальність – можливість виконання різних типів повторюваних, небезпечних або фізично напружених операцій, які традиційно виконуються людиною;
- адаптивність – здатність до переналаштування та коригування траєкторії руху на основі даних сенсорних систем;
- автономність – можливість тривалого функціонування без безпосереднього втручання оператора при збереженні стабільності виконання виробничих процесів;
- інтелектуальність – інтеграція алгоритмів машинного навчання та штучного інтелекту для розпізнавання об’єктів, прийняття рішень у реальному часі та оптимізації енергоспоживання.

Зазначені властивості маніпуляційних систем пояснюють їх широке поширення не лише у промисловому виробництві, а й в інших сферах людської діяльності. Упродовж останніх десятиліть спостерігається стійка тенденція їх застосування у медицині [3], сферах послуг [4], будівництві [5], археології [6], агропромисловості [7] та інших галузях [8, 9, 10]. Переконливим прикладом, що відображає універсальність таких систем є колаборативні роботи-маніпулятори, зокрема робот UR5 компанії Universal Robots, який використовується у різних напрямках людської діяльності, таких як допомога пацієнтам у реабілітації [11] (Рис.1.1, а), обстеження і обслуговування морських вітрових електростанцій [12, 13, 14] (Рис.1.1, б), автоматизоване 2D- та 3D- сканування об’єктів для перевірки стану продукції [15] (Рис.1.1, в) та адитивному виробництві [16] (Рис.1.1, г).



а) б) в) г)

Рис. 1.1. Застосування робота-маніпулятора Universal Robots UR5 у різних сферах життєдіяльності: а) виконання реабілітаційних вправ для людей, що пережили інсульт, б) перевірка на наявність корозії за допомогою інспекційного інструменту FMCW, встановленого на маніпулятор, в) здійснення 3D-сканування та дефектоскопії виробів, г) реалізація технології FDM-друку з використанням маніпулятора UR5

Наведений вище приклад демонструє, що універсальність сучасних роботів-маніпуляторів значною мірою зумовлена еволюцією їх конструктивної та кінематичної структури. Прагнення розширити робочу область, підвищити точність позиціонування та адаптувати маніпулятори до різноманітних технологічних операцій зумовило появу широкого спектру кінематичних конфігурацій.

Залежно від взаємного розташування ланок і типу кінематичних пар формуються різні структурні схеми роботів, кожна з яких має власні особливості робочої області, динамічних характеристик і технологічних можливостей. У результаті цього в сучасній робототехніці сформувався цілий ряд конструктивних типів маніпуляторів, серед яких виділяють:

- декартові;
- циліндричні;
- паралельні (дельта-роботи);
- послідовні (антропоморфні роботи);
- SCARA робот-маніпулятор;
- колаборативні роботи.

Кожна з наведених архітектур виникла як відповідь на потреби конкретних виробничих задач і, відповідно, має свої переваги та недоліки. Розглянемо кожен тип більш детально.

Декартові роботи-маніпулятори також відомі як лінійні роботи або портативні, використовують три лінійні осі (X, Y, Z) для точного переміщення робочого органу, що робить їх незамінними для операцій збірки та інспекції (Рис.1.2) [16].

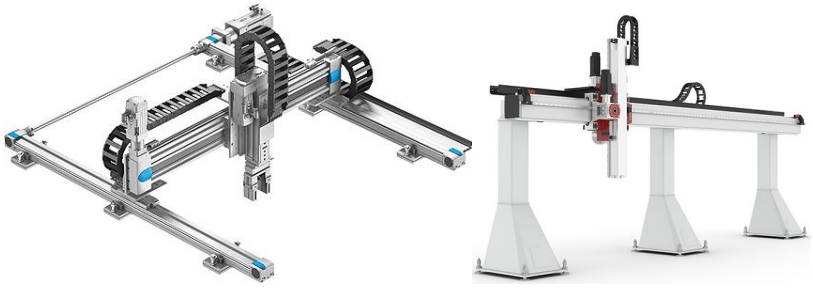


Рис. 1.2. Зовнішній вигляд декартових роботів-маніпуляторів

Переваги: висока точність позиціонування до 0,001 мм; легка масштабованість під необхідні розміри робочого простору.

Недоліки: обмежена ефективність при складних криволінійних траєкторіях; підвищене енергоспоживання через значні масо-габаритні характеристики.

Приклад застосування: складання друкованих плат в електронній промисловості.

Циліндричні роботи-маніпулятори реалізують переміщення робочого органу через поєднання вертикальної (Z), радіальної (R) та обертальної (θ) осей, формуючи циліндричний робочий простір [16]. Конструктивно подібні роботи складаються з вертикальної стійки, закріпленої на поворотній основі, та маніпуляторної ланки, яка може висуватися у радіальному напрямку і обертатися навколо колони (Рис.1.3).

Переваги: ефективний обхід перешкод у циліндричній робочій зоні; компактність встановлення порівняно з іншими типами.

Недоліки: робоча область обмежена циліндричною геометрією; ризики накопичення похибки позиціонування через поєднання лінійних та обертальних рухів.

Приклад застосування: складання та пакування медичних виробів, де важливі компактність і маневреність.

Рис. 1.3. Зовнішній вигляд циліндричних роботів-маніпуляторів

Паралельні роботи-маніпулятори (дельта-роботи), також відомі як дельта-роботи, відзначаються дуже високою швидкістю і точністю виконання операцій. Зазвичай вони встановлюються над робочою зоною та здійснюють рух у тривимірному просторі за допомогою кількох синхронно керованих ланок, що забезпечує надвисоку швидкість і точність операцій (Рис.1.4) [17].

Переваги: здатність виконувати сотні циклів захоплення за хвилину; висока точність і повторюваність; компактність – не займають виробничої площі.

Недоліки: обмежена вантажопідйомність; відносно невелика робоча зона; високі початкові інвестиції та вимоги до кваліфікації обслуговуючого персоналу.

Рис. 1.4. Зовнішній вигляд паралельних роботів-маніпуляторів

Приклад застосування: сортування, пакування та швидкісне переміщення продукції у харчовій, електронній та фармацевтичній промисловості.

Послідовні (антропоморфні) роботи-маніпулятори є одним із найпоширеніших типів промислових систем. Їх конструкція подібна до людської руки: обертова основа та послідовно з'єднані шарнірами ланки, що забезпечує високу маневреність у складному просторовому середовищі (Рис.1.5) [18].

Рис. 1.5. Зовнішній вигляд послідовних (антропоморфних) роботів-маніпуляторів

Переваги: широкий діапазон рухів і великий радіус досягнення об'єктів; різноманіття конструктивних варіантів за вантажопідйомністю та розмірами.

Недоліки: потребують значного робочого простору; менша швидкість порівняно з Delta- та SCARA-роботами; нижча точність позиціонування порівняно з паралельними структурами, що вимагає складних алгоритмів керування.

Приклад застосування: зварювання, фарбування та переміщення деталей на автомобільних виробничих лініях.

SCARA робот-маніпулятори є промисловими роботами, призначеними для виконання швидких і точних операцій у автоматизованому виробництві. Їх

конструкція зазвичай включає чотири осі: дві обертальні у площині XY, вертикальну лінійну (Z) та вісь обертання робочого органу, що забезпечує високу жорсткість у вертикальному напрямку та гнучкість у горизонтальній площині (Рис.1.6) [19].

Рис. 1.6. Зовнішній вигляд SCARA роботів-маніпуляторів

Переваги: висока швидкість циклів «pick-and-place»; точність у площині XY; компактність для інтеграції в обмежений виробничий простір.

Недоліки: обмежене вертикальне переміщення та малий робочий простір порівняно з 6-осьовими роботами; невисока вантажопідйомність.

Приклад застосування: автоматизоване складання електронних компонентів, паяння та переміщення дрібних деталей у електронній промисловості.

Колаборативні роботи-маніпулятори, також відомі як коботи, спроектовані для безпечної прямої взаємодії з людиною в спільному робочому просторі без захисних огорож (Рис.1.7) [20]. Зазвичай мають 6-осьову антропоморфну структуру з інтегрованими датчиками моменту в кожному сполученні та системами обмеження сили, що дозволяє миттєво зупинятися при контакті з людиною відповідно до стандартів ISO 10218 та ISO/TS 15066.

Переваги: безпека спільної роботи з людиною без захисних кліток; легкість програмування через функцію «навчання показом»; компактність і мобільність між ділянками виробництва – що робить їх ідеальними для дрібносерійного виробництва (HMLV – High Mix, Low Volume) [21].

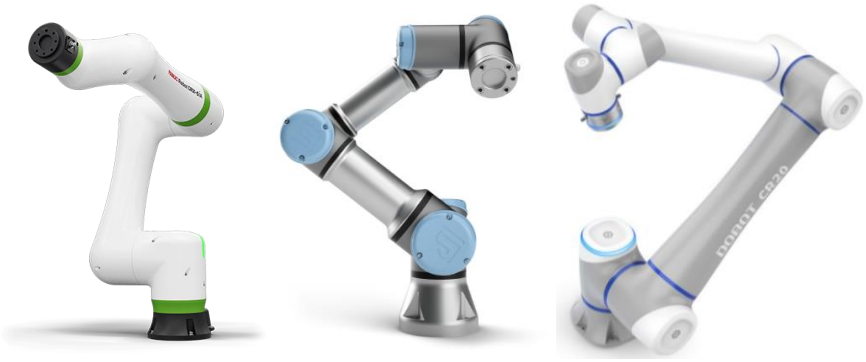


Рис. 1.7. Зовнішній вигляд колаборативних роботів-маніпуляторів

Недоліки: обмежена швидкість і вантажопідйомність через вимоги безпеки; менша жорсткість і точність порівняно з традиційними промисловими роботами; вища вартість на одиницю вантажопідйомності через спеціалізовані датчики.

Приклад застосування: використовуються в автомобільній та пакувальній промисловості для «подачі деталей», закручування гвинтів, нанесення клею та контролю якості поруч з оператором.

Попри стрімкий розвиток колаборативних та інших спеціалізованих кінематичних структур, вони здебільшого орієнтовані на вузькоспеціалізовані задачі, де пріоритетами є безпека взаємодії з людиною або висока швидкість окремих операцій. Натомість вимоги масового та серійного виробництва передбачають високу універсальність, вантажопідйомність, точність та автономність робототехнічних систем. У цьому контексті послідовні антропоморфні роботи-маніпулятори продовжують займати провідне місце

серед промислових систем: завдяки універсальній кінематичній структурі, значній робочій області та здатності реалізовувати складні просторові рухи вони залишаються основним інструментом автоматизації у багатьох галузях промисловості [22-24]. Сучасні тенденції спрямовані не на їх витіснення, а на подальше вдосконалення через підвищення точності, інтеграцію інтелектуальних систем керування та розширення функціональних можливостей [25-29].

Таким чином, незважаючи на появу нових типів маніпуляторів, саме антропоморфні роботи залишаються найзастосованішими для побудови високоефективних автоматизованих виробничих систем і продовжують активно розвиватися відповідно до вимог сучасної промисловості.

1.2. Методи кінематичного аналізу маніпуляторів

Кінематичний аналіз – це початковий етап проектування робототехнічних систем, який полягає у аналітичному описі геометрії руху маніпулятора відносно заданої фіксованої системи координат (базової системи) без урахування сил і моментів, які породжують цей рух [30, 31, 32]. Основною метою кінематичного аналізу є встановлення математичного зв’язку між узагальненими координатами зчленувань (сполучень) робота та положенням його робочого органа (захвата) у тривимірному просторі, де під положенням розуміється сукупність позиції та орієнтації захвата.

Кінематичний аналіз традиційно поділяється на дві взаємопов’язані підзадачі:

- перша – вирішення прямої задачі кінематики, що передбачає побудову кінематичної схеми маніпулятора, застосування методу Денавіта-Хартенберга для формалізованого опису отриманих кінематичних

ланцюгів, задання початкового стану сполучень та обчислення координат положення і орієнтації захвата у базовій системі відліку;

- друга – вирішення зворотної задачі кінематики, де на основі побудованої кінематичної схеми та визначених вимог до точності й швидкодії обирається відповідний клас алгоритмів для знаходження значень узагальнених координат сполучень, що відповідають заданому положенню захвата у декартовому просторі.

Детальний розгляд кожної із зазначених підзадач наведено у п.п. 1.2.1 та 1.2.2.

1.2.1. Пряма задача кінематики

Пряма задача кінематики (ПЗК) полягає у визначенні положення та орієнтації робочого органа (захвата) маніпулятора у тривимірному просторі відносно базової (нерухомої) системи координат [32, 33]. Дана задача є однозначною – для заданого набору узагальнених координат існує єдиний розв’язок, тому вона може бути розрахована для довільної кількості ланок, що утворюють кінематичний ланцюг маніпулятора.

Повний опис положення та орієнтації захвата у тривимірному просторі вимагає задання шести незалежних параметрів: трьох для визначення координат позиції та трьох для опису орієнтації. Проте безпосередня робота з таким набором параметрів супроводжується значним обчислювальним навантаженням. З метою спрощення цього процесу Ж. Денавітом і Р. С. Хартенбергом було запропоновано компактний матричний підхід, відомий як представлення Денавіта-Хартенберга (ДХ-представлення) [34], який дозволяє описувати перетворення між суміжними системами координат за допомогою чотирьох уніфікованих геометричних параметрів.

Згідно з цим підходом, кожній *i*-й ланці маніпулятора ставляться у відповідність параметри $d_i, a_i, \alpha_i, \theta_i$, які однозначно визначають взаємне розташування суміжних ланок:

- d_i – зміщення ланки, що є відстанню від початку координат *(i−1)*-ї системи координат до точки, де спільний перпендикуляр перетинається з віссю z_{i-1};
- a_i – довжина ланки, що визначається як відстань між осями z_{i-1} та z_i вздовж осі x_i, спільна нормаль двох послідовних ланок;
- α_i – кут повороту ланки, що є кутом повороту навколо осі x_i від z_{i-1} до z_i;
- θ_i – кут між поточною та наступною ланками, що є кутом між x_{i-1} та x_i на осі z_{i-1}.

Значення цих параметрів безпосередньо визначаються з кінематичної схеми маніпулятора: для обертальних сполучень змінною є θ_i, тоді як d_i залишається сталою величиною; для поступальних – навпаки. Геометричні параметри a_i та α_i є сталими значеннями відповідно до особливостей конструкції і не змінюються під час руху.

Перехід між двома послідовними системами координат (*i*-ї системи координат ланки у *(i−1)*-шу) описується за допомогою однорідної матриці перетворення розміром 4×4 [30]:

$$A_i^{i-1} = \begin{bmatrix} R_{3\times3} & p_{3\times1} \\ f_{1\times3} & 1\times1 \end{bmatrix} = \\ = \begin{bmatrix} \text{Поворот} & \text{Зсув} \\ \text{Перетворення перспективи} & \text{Масштабування} \end{bmatrix} = \\ = \begin{bmatrix} cos(\theta_i) & -sin(\theta_i)cos(\alpha_i) & sin(\theta_i)sin(\alpha_i) & a_i cos(\theta_i) \\ sin(\theta_i) & cos(\theta_i)cos(\alpha_i) & -cos(\theta_i)sin(\alpha_i) & a_i sin(\theta_i) \\ 0 & sin(\alpha_i) & cos(\alpha_i) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1.1)$$

де підматриця $R_{3\times3}$ є матрицею повороту; підматриця $p_{3\times1}$– вектор положення початку координат повернутої системи відліку відносно абсолютної; підматриця $f_{1\times3}$, що задає перетворення перспективи.

Для маніпулятора з n ступенями вільності положення та орієнтація захвата відносно базової системи координат визначається послідовним множенням матриць усіх ланок кінематичної моделі:

$$T_n^0 = A_1^0 A_2^1 \dots A_n^{n-1} = \prod_{i=1}^{n} A_i^{i-1}. \quad (1.2)$$

Отримана матриця перетворення також має розмір 4×4 і містить повну інформацію про положення та орієнтацію захвата. Її структурно можна представити у наступному вигляді:

$$T_n^0 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} n & s & a & p \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (1.3)$$

де вектор $p = \left(p_x, p_y, p_z\right)^{\mathrm{T}}$ визначає позицію захвата у базовій системі координат, а вектори n, s та a описують його орієнтацію як напрямки відповідних осей. Таким чином, розв'язання прямої задачі кінематики зводиться до обчислення елементів підсумкової матриці перетворення на основі геометричних параметрів ланок та поточних значень узагальнених координат у сполученнях.

1.2.2. Зворотна задача кінематики

Зворотна задача кінематики (ЗЗК) полягає у знаходженні узагальнених координат сполучень маніпулятора $\theta = (\theta_1, \theta_2, \dots, \theta_n)^T$ за заздалегідь відомим положенням та орієнтацією захвата, заданими у базовій системі координат [32]. На відміну від прямої задачі кінематики, яка має єдиний розв'язок, ЗЗК є неоднозначною і залежить від бажаного положення та орієнтації: для досягнення однієї й тієї самої точки у робочому просторі розв'язок може не існувати взагалі або існувати скінченна/нескінченна множина розв'язків. Так, для плоского дволанкового маніпулятора існують дві можливі конфігурації для досягнення

заданої точки – «лікоть вгору» та «лікоть вниз». Для надлишкових маніпуляторів, кількість ступенів свободи яких перевищує шість, простір розв’язків є нескінченновимірним, що надає додаткову гнучкість при плануванні траєкторії, але ускладнює вибір оптимального розв’язку.

Для вирішення зворотної задачі кінематики існують різні методи, які умовно можна розділити на чотири великі групи [35, 36]:

- аналітичний (точний) метод;
- ітераційні (чисельні\наближені) методи;
- за допомогою машинного навчання;
- гібридні.

Аналітичний метод забезпечує отримання розв’язку оберненої задачі кінематики з абсолютною точністю, тоді як ітераційні підходи дозволяють знаходити розв’язок із наперед заданою похибкою [35]. Завдяки цьому точний метод має певну перевагу, оскільки, як правило, характеризуються вищою швидкодією та не потребує ітераційних процедур. З цієї причини його широко застосовують для розв’язання ЗЗК у випадках, коли це можливо.

Водночас, як зазначено у [36], вирази аналітичного рішення суттєво ускладнюються зі зростанням складності кінематичних ланцюгів маніпулятора, що обмежує його практичне застосування. Зокрема, для маніпуляторів із надлишковим числом ступенів вільності множина розв’язків є нескінченною, що унеможливлює однозначне визначення результату за допомогою точних методів.

У таких умовах доцільним є застосування чисельних методів, які забезпечують універсальність і можуть бути застосовані до широкого класу робототехнічних систем, хоча й потребують ітераційних обчислень та залежать від початкового наближення.

Окрему групу становлять підходи, засновані на методах машинного навчання, що надають рішення для швидкого оброблення і уникнення проблем

надлишковості та сингулярності, завдяки апроксимації даних зворотної кінематики з декартового простору у спільний [37]. Основними недоліками таких підходів є потреба у значній кількості навчальних даних, високі обчислювальні витрати та, в деяких випадках, неможливість забезпечення необхідної точності [38].

Гібридні методи застосовуються для маніпуляторів, що функціонують у складних середовищах, зокрема в комічній роботехніці, де критично важлива висока стійкість та адаптивність до сингулярних положень, а також можливість управління в реальному часі. Перевагою таких підходів є можливість одночасного використання можливостей декількох вище зазначених методів, наприклад поєднання обчислювальної ефективності аналітичних підходів та гнучкості чисельних методів з подальшим використанням нейронної мережі для аналізу даних або навчання [39, 40].

Незважаючи на різноманіття підходів, найбільш поширеними методами розв’язання ЗЗК залишаються аналітичні та ітераційні методи, тому саме вони розглядаються більш детально у подальшому.

Аналітичний метод. Також відомий як геометричний, оскільки в процесі розв’язання ЗЗК здійснюється декомпозиція просторової геометрії маніпулятора на сукупність планарних 2D-підзадач. Для кожної з отриманих площинних підзадач сполученні кути визначаються з тригонометричних співвідношень (теореми косинусів та синусів) між ланками маніпулятора, після чого отримані вирази розв’язуються алгебраїчно для знаходження значень узагальнених координат сполучень.

Для ілюстрації особливостей даного підходу розглянемо його застосування на прикладі дволанкового планарного маніпулятора, кінематична схема якого наведена на Рис.1.8. У цій моделі a_1, a_2 – довжини ланок, а шуканими величинами є значення кутів сполучень θ_1, θ_2, що забезпечують досягнення заданої позиції робочого органа.

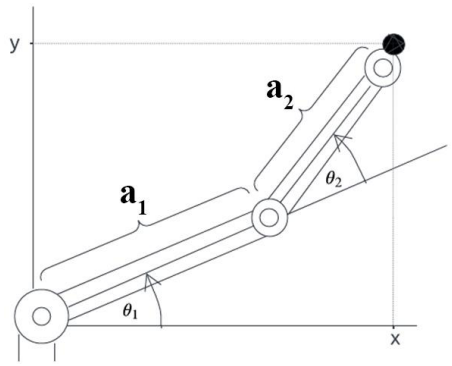
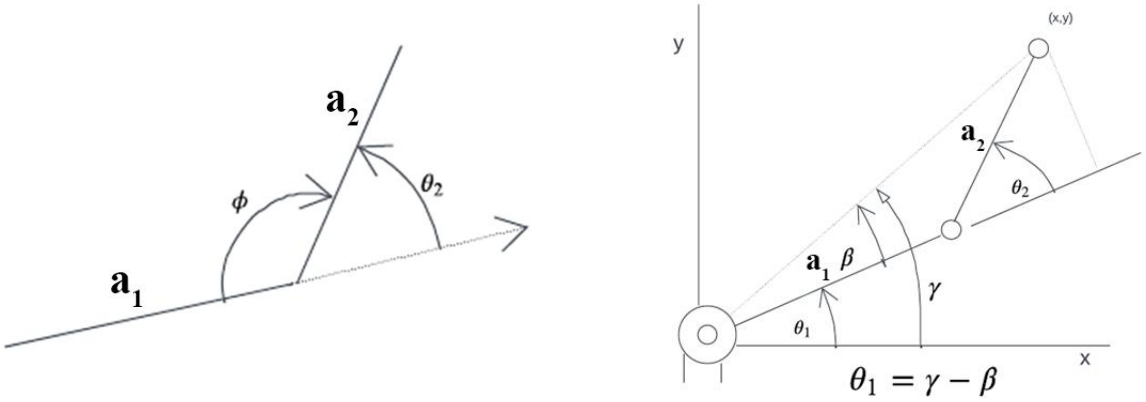


Рис. 1.8. Кінематична схема дволанкового планарного маніпулятора a_1 та a_2

Спочатку за теоремою косинусів знаходимо кут ϕ, утворений ланками a_1 та a_2 (Рис. 1.8, Рис. 1.9, а):

$$cos(\phi) = \frac{(a_1)^2 + (a_2)^2 - x^2 - y^2}{2a_1a_2}. \quad (1.4)$$



а) б)

Рис. 1.9. Декомпозиція просторової геометрії маніпулятора: а) знаходження θ_2, б) знаходження θ_1

Оскільки $\theta_2 = \pi - \phi$, то використовуючи тригонометричну тотожність $cos(\pi - \phi) = -cos(\phi)$, отримуємо:

$$\theta_2 = arccos \left(\frac{x^2 + y^2 - (a_1)^2 - (a_2)^2}{2a_1a_2}\right). \quad (1.5)$$

Оскільки кут γ – напрямок вектора до схвата від основи, а кут β – відхилення a_2 від a_1, то кут θ_1 знаходимо наступним чином:

$$\theta_1 = \gamma - \beta = arctan\left(\frac{y}{x}\right) - arctan\left(\frac{a_2 sin(\theta_2)}{a_1 + a_2 cos(\theta_2)}\right). \quad (1.6)$$

Ітераційні (чисельні) методи. Не потребують замкненого аналітичного розв'язку, а покроково наближаються до бажаного положення схвата шляхом послідовного уточнення вектора узагальнених координат кутів сполучень маніпуялтора $\theta = (\theta_1, \theta_2, \ldots, \theta_n)^T$. Найбільш поширеними представниками цієї групи розглянуті нижче.

Метод псевдооберненої матриці Якобі здійснює обчислення приросту кутів сполучень на кожній ітерації за допомогою матриці Мура-Пенроуза:

$$(J^T J)^{-1} J^T, \quad (1.7)$$

де формула матриці Якобі має наступний вигляд:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \cdots & \frac{\partial x}{\partial \theta_n} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \cdots & \frac{\partial y}{\partial \theta_n} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \cdots & \frac{\partial z}{\partial \theta_n} \end{bmatrix}. \quad (1.8)$$

Використання матриці Мура-Пенроуза в цьому методі забезпечує рух до цільової точки з мінімальною нормою вектора узагальнених координат [41]. Проте метод потребує або явного обернення матриці, або розкладання за сингулярними значеннями (SVD) на кожній ітерації, що визначає кубічну обчислювальну складність $O(n^3)$ [41].

Метод Левенберга–Марквардта (метод демпфованих найменших квадратів) базується на модифікованому підході Гауса-Ньютона та забезпечує стійкість алгоритму поблизу сингулярних конфігурацій робочої області маніпулятора шляхом введення параметра демпфування λ у вираз для оберненої матриці [42]:

$$(J^T J + \lambda^2 I)^{-1} J^T. \quad (1.9)$$

Метод характеризується швидкою збіжністю, однак також потребує матричної інверсії Якобі на кожній ітерації, а оптимальний вибір параметра λ вимагає додаткового динамічного налаштування в процесі роботи алгоритму.

Метод градієнтного спуску вирізняється своєю універсальністю і придатністю для розрахунку складних кінематичних структур, оскільки використовує транспоновану матрицю Якобі J^T замість псевдооберненої. Основна ідея даного методу полягає у мінімізації цільової функції похибки, яка відображає просторове відхилення поточного положення робочого органу від заданого [43, 58]:

$$J(\theta) = \frac{1}{2}((x_{des} - x(\theta))^2 + (y_{des} - y(\theta))^2 + (z_{des} - z(\theta))^2) \to min, \quad (1.10)$$

де $\theta = (\theta_1, \theta_2, \ldots, \theta_n)^T$ – вектор приєднаних координат; $x_{des}, y_{des}, z_{des}$ – координати цільової точки; $x(\theta), y(\theta), z(\theta)$ – поточні координати, обчислені за допомогою рівнянь прямої кінематики на кожній ітерації.

Недоліком даного підходу є повільніша збіжність порівняно з псевдообернем методом поблизу сингуляних положень.

В даній роботі реалізовано комбінацію геометричного та ітераційного методів, що забезпечує їх ефективне застосування в умовах обмежених апаратних ресурсів мікропроцесора. Геометричний алгоритм відіграє роль дублювання рішення двома різними положеннями (лікоть вгору та лікоть вниз), забезпечуючи гарантоване отримання точного розв’язку в ситуаціях, коли ітераційний метод втрачає збіжність.

1.3. Методи планування траєкторій маніпуляторів

Планування траєкторії – формування руху робочого органу маніпулятора у просторі сполучень (Join space) або декартовому просторі з метою узгодження усіх змінних, що описують рух сполучень. Це забезпечує переміщення захвату до заданих точок простору з необхідною орієнтацією з урахуванням обмежень приводів і конструктивних можливостей робота в цілому.

Планування траєкторії є підзадачею загальної задачі планування руху захвату маніпулятора, яка охоплює такі рівні:

- планування задачі (визначення високорівневих цілей),
- планування шляху (генерація послідовності вузлових точок),
- планування траєкторії (формування часових проміжків проходження шляху з урахуванням обмежень на кути, положення, швидкість та прискорення)
- відпрацювання траєкторії системою керування.

Одним із ключових рішень при синтезі системи керування є вибір простору, в якому здійснюється інтерполяція заданої траєкторії. Порівняльну характеристику обох підходів наведено у Табл. 1.1.

Таблиця 1.1 – Порівняння планування у просторі сполучень та декартовому просторі

Параметр	Простір сполучень	Декартовий простір
Тип траєкторії схвата	Крива лінія	Пряма лінія (найкоротший шлях до цілі)
Використання ЗЗК	Низьке (лише для вузлових точок)	Високе (на кожному кроці)
Обчислювальні витрати	Низькі	Високі

Швидкість виконання	Вища (ЗЗК розв’язується лише у вузлових точках)	Нижча (ЗЗК розв’язується на кожному часовому кроці)
Передбачуваність траєкторії схвата	Низька	Висока (інтерполяція безпосередньо у декартовому просторі)
Обробка перешкод та зіткнень	Ускладнена (проміжні точки не гарантовано задовольняють обмеження)	Краща (траєкторія явно задана у просторі задачі)
Плавність руху приводів	Висока, легко верифікується	Не гарантована, складніше верифікувати
Проходження через проміжні точки	Не гарантує дотримання обмежень сполучень та відсутності зіткнень	Точне проходження через задані точки у просторі задачі
Залежність від кінематики маніпулятора	Так	Ні

Планування у просторі сполучень є найпростішим у реалізації підходом, оскільки інтерполяція здійснюється безпосередньо між значеннями узагальнених координат сполучень, а зворотна задача кінематики розв’язується лише для початкової та кінцевої (та, за необхідності, проміжних вузлових) точок. Перевагами є низька обчислювальна складність та гарантоване проходження через задані конфігурації сполучень. Недоліком є те, що захват рухається по криволінійній траєкторії у декартовому просторі, яка залежить від конкретної кінематики маніпулятора і може бути непередбачуваною з точки зору зіткнень з перешкодами.

Планування у декартовому просторі забезпечує природній прямолінійний рух схвата у робочому середовищі, що є найкоротшим шляхом між двома точками та не залежить від кінематики конкретного маніпулятора. Проте суттєвим недоліком є висока обчислювальна вартість: зворотна задача кінематики має розв’язуватися на кожному кроці оновлення траєкторії (зазвичай із частотою 60-2000 Гц), що накладає жорсткі вимоги на продуктивність

обчислювальної системи [44]. Крім того, у декартовому просторі можуть виникати специфічні геометричні проблеми: деякі проміжні точки прямолінійного шляху можуть виходити за межі досяжного робочого простору навіть при досяжності початкової та кінцевої точок, а поблизу сингулярних конфігурацій швидкості сполучень можуть необмежено зростати.

Незалежно від обраного простору планування, для інтерполяції між вузловими точками використовуються різні математичні функції. Найпопулярнішою є поліноміальна інтерполяція, що розглянута нижче.

Поліноміальні траєкторії дозволяють забезпечити задані граничні умови на положення, швидкість та прискорення у вузлових точках шляхом підбору коефіцієнтів полінома. Найпоширенішими є поліноми третього та п’ятого порядків.

Кубічний поліном (3-го порядку) задовольняє чотири граничні умови – положення та швидкість на початку та в кінці сегмента [33]:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3. \quad (1.11)$$

Продиференціювавши вираз (1.11) знаходимо рівняння швидкості та прискорення:

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2, \quad (1.12)$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t. \quad (1.13)$$

З отриманої системи лінійних рівнянь (1.12)-(1.23) для початкових та кінцевих значень θ_0 та θ_f отримуємо наступні значення коефіцієнтів кубічного поліному [45]:

$$a_0 = \theta_0, a_1 = 0, a_2 = \frac{3}{t_f^2}(\theta_f - \theta_0), a_3 = -\frac{2}{t_f^3}(\theta_f - \theta_0). \quad (1.14)$$

Квінтичний поліном (поліном 5-го порядку) додатково надає можливості для задання прискорення, а тому вимагає шести граничних умов [46]:

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5. \quad (1.15)$$

Перша та друга похідні є гладкими функціями:

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2 + 4a_4 t^3 + 5a_5 t^4, \tag{1.16}$$

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t + 12a_4 t^2 + 20a_5 t^3. \tag{1.17}$$

Через це розрахунки значно ускладнюються, тому шукані коефіцієнти поліному a_0, $a_1 \dots a_5$ не виражаються компактно через граничні умови безпосередньо, а визначаються розв'язанням системи шести лінійних рівнянь для t_0 та t_{end}, докладно розглянутої у п.п. 2.5.

1.4. Висновки до розділу 1

У розділі проведено огляд підходів до математичного моделювання роботів-маніпуляторів, зокрема конструктивних конфігурацій, методів кінематичного аналізу, оцінювання маніпулятивності та планування траєкторій.

Встановлено, що сучасні маніпулятори представлені декартовими, циліндричними, паралельними, антропоморфними, SCARA та колаборативними структурами, кожна з яких має власні переваги та обмеження. Обґрунтовано доцільність використання антропоморфних маніпуляторів для задач, що потребують великої робочої зони та складних просторових переміщень.

Розглянуто методи кінематичного аналізу маніпуляторів. Показано, що пряма задача кінематики однозначно розв'язується за допомогою апарату Денавіта-Хартенберга, тоді як зворотна задача є неоднозначною та потребує застосування аналітичних або чисельних методів. Визначено переваги та обмеження методів псевдооберненої матриці Якобі, Левенберга-Марквардта і градієнтного спуску.

Отримані результати є теоретичною основою для вибору методів моделювання та синтезу алгоритмів керування у розробленому інформаційно-обчислювальному комплексі.

РОЗДІЛ 2

МАТЕМАТИЧНА МОДЕЛЬ РУХУ ТРИЛАНКОВОГО МАНІПУЛЯТОРА

2.1. Кінематична схема триланкового маніпулятора

Як було зазначено у п. 1.1, найбільш поширеним типом роботів-маніпуляторів у промисловості є послідовні антропоморфні механізми. У зв’язку з цим для дослідження методів кінематичного аналізу, планування траєкторій і динамічного моделювання доцільно використовувати спрощені, але репрезентативні моделі таких систем. Однією з найбільш простих і водночас поширених моделей є триланковий маніпулятор із трьома обертальними сполученнями – конфігурація RRR (Revolute-Revolute-Revolute).

При цьому важливо зазначити, що розроблені для триланкового маніпулятора математичні моделі, алгоритми керування та методи планування траєкторій не є самоціллю: вони формують універсальну методологічну основу, яка може бути послідовно розширена до систем із більшою кількістю ступенів вільності. Зокрема, як показано у [47], підходи, розроблені та верифіковані на 3-DOF маніпуляторі, успішно масштабуються до 7-DOF робототехнічних систем, що виконують складні промислові задачі переміщення і точного позиціонування матеріалів. Таким чином, триланковий маніпулятор виступає не лише самостійним об’єктом дослідження, а й базовою платформою для відпрацювання підходів, що надалі можуть бути інтегровані у повнофункціональні промислові комплекси.

Структура послідовного антропоморфного 3-DOF маніпулятора представляється як триланковий кінематичний ланцюг з послідовними обертальними кінематичними парами, де один кінець ланцюга закріплено на обертальній основі, що пов’язана з базовою системою координат, а вільний кінець ланцюга з’єднано із робочим органом. Таким чином досліджуваний маніпулятор має три ступені вільності [48, 46].

Після дослідження конструкції та основних характеристик маніпулятора застосовуються правила Денавіта-Хартенберга (ДХ-представлення), що дають змогу описати взаємне положення та орієнтацію ланок шляхом введення базової та послідовності ортонормованих декартових систем координат [49, 46]. Для розглянутого маніпулятора задається набір систем координат: (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3), де базовою є перша. Орієнтація осей у кожній локальній системі координат визначається відповідно до стандартних правил:

- вісь z_{i-1} суміщається з віссю обертання відповідного сполучення;
- вісь x_i є ортогональною до z_{i-1} і спрямована вздовж ланки;
- вісь y_i визначається таким чином, щоб утворювати праву ортонормовану систему координат разом з осями x_i, z_i.

На Рис. 2.1 наведено кінематичну схему маніпулятора з відповідним розташуванням локальних систем координат, де $\theta_1, \theta_2, \theta_3$ – кути повороту відповідних сполучень, L_1, L_2, L_3 – довжини ланок маніпулятора.

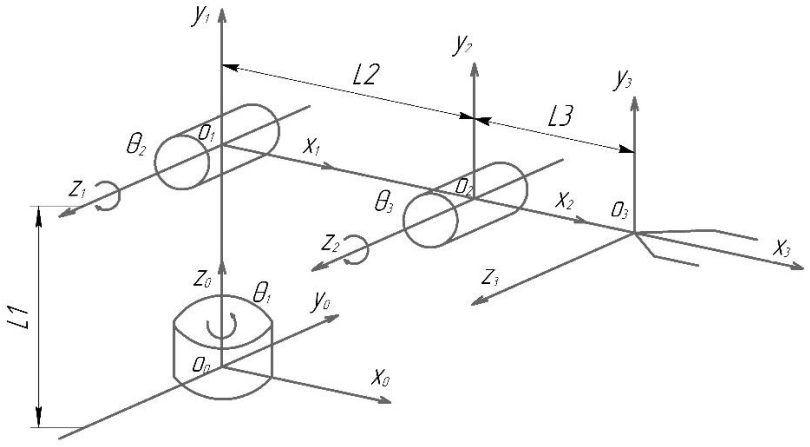


Рис. 2.1. Кінематична схема досліджуваного триланкового маніпулятора

2.2. Розв’язання прямої задачі кінематики

Згідно з побудованою кінематичною схемою (Рис. 2.1) та правилами ДХ-представлення, наведеними у п. 1.2.1, ДХ-параметри для досліджуваного триланкового маніпулятора зведено до Табл. 2.1 [46].

Таблиця 2.1 – Параметри ДХ-представлення триланкового маніпулятора

№ кінематичної пари (i)	Тип пари	Параметри маніпулятора			
		θ_i	d_i	a_i	α_i
1	Revolute	θ_1	L_1	0	$\frac{\pi}{2}$
2	Revolute	θ_2	0	L_2	0
3	Revolute	θ_3	0	L_3	0

Наступним кроком послідовно обчислено матриць однорідного перетворення A_i^{i-1} для кожної ланки за формулою (1.1) з підстановкою відповідних ДХ-параметрів з Табл. 2.1 [46]:

$$A_1^0 = \begin{bmatrix} cos(\theta_1) & 0 & sin(\theta_1) & 0 \\ sin(\theta_1) & 0 & -cos(\theta_1) & 0 \\ 0 & 1 & 0 & L_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.1)$$

$$A_2^1 = \begin{bmatrix} cos(\theta_2) & -sin(\theta_2) & 0 & L_2 cos(\theta_2) \\ sin(\theta_2) & cos(\theta_2) & 0 & L_2 sin(\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.2)$$

$$A_3^2 = \begin{bmatrix} cos(\theta_3) & -sin(\theta_3) & 0 & L_3 cos(\theta_3) \\ sin(\theta_3) & cos(\theta_3) & 0 & L_3 sin(\theta_3) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.3)$$

Результуюча матриця T_3^0 визначена як добуток (2.1)-(2.2) згідно з виразом (1.2) [46]:

$$T_3^0 = A_1^0 \cdot A_2^1 \cdot A_3^2 = \begin{bmatrix} n_x & s_x & a_x & p_x \\ n_y & s_y & a_y & p_y \\ n_z & s_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad (2.4)$$

де:

$$n_x = cos\,\theta_1\,cos\,\theta_2\,cos\,\theta_3 - cos\,\theta_1\,sin\,\theta_2\,sin\,\theta_3, \quad (2.5)$$
$$n_y = sin\,\theta_1\,cos\,\theta_2\,cos\,\theta_3 - sin\,\theta_1\,sin\,\theta_2\,sin\,\theta_3, \quad (2.6)$$
$$n_z = sin\,\theta_2\,cos\,\theta_3 + cos\,\theta_2\,sin\,\theta_3, \quad (2.7)$$
$$s_x = -cos\,\theta_1\,cos\,\theta_2\,sin\,\theta_3 - cos\,\theta_1\,sin\,\theta_2\,cos\,\theta_3, \quad (2.8)$$
$$s_y = -sin\,\theta_1\,cos\,\theta_2\,sin\,\theta_3 - sin\,\theta_1\,sin\,\theta_2\,cos\,\theta_3, \quad (2.9)$$
$$s_z = -sin\,\theta_2\,sin\,\theta_3 + cos\,\theta_2\,cos\,\theta_3, \quad (2.10)$$
$$a_x = sin\,\theta_1, \quad (2.11)$$
$$a_y = -cos\,\theta_1, \quad (2.12)$$
$$a_z = 0, \quad (2.13)$$
$$p_x = L_2\,cos\,\theta_1\,cos\,\theta_2 + L_3\,cos\,\theta_1\,cos\,\theta_2\,cos\,\theta_3 - L_3\,cos\,\theta_1\,sin\,\theta_2\,sin\,\theta_3, \quad (2.14)$$
$$p_y = L_2\,sin\,\theta_1\,cos\,\theta_2 + L_3\,sin\,\theta_1\,cos\,\theta_2\,cos\,\theta_3 - L_3\,sin\,\theta_1\,sin\,\theta_2\,sin\,\theta_3, \quad (2.15)$$
$$p_z = L_1 + L_2\,sin\,\theta_2 + L_3\,sin\,\theta_2\,cos\,\theta_3 + L_3\,cos\,\theta_2\,sin\,\theta_3. \quad (2.16)$$

Елементи (2.14)-(2.16) є координатами вектору положення захвата маніпулятора.

2.3. Вирішення зворотної задачі кінематики

Метод градієнтного спуску. Як зазначено у п.1.2.2 вирішення ЗЗК у методі градієнтного спуску зводиться до мінімізації цільової функції [43, 58]:

$$J(\theta) = \frac{1}{2}((x_{des} - x(\theta))^2 + (y_{des} - y(\theta))^2 + (z_{des} - z(\theta))^2) \to min, \quad (2.17)$$

де $\theta = (\theta_1, \theta_2, \theta_3)^T$ – вектор приєднаних координат; $x_{des}, y_{des}, z_{des}$ – бажані координати захвата; $x(\theta), y(\theta), z(\theta)$ – координати захвата, що отримані в циклі розвязання прямої задачі кінематики за виразами (2.14)-(2.16).

Коригування кутів зчленувань маніпулятора шляхом коригування веткора θ приєднаних координат у (2.17) здійснюється в циклі формулою [43]:

$$\theta_i^{k+1} = \theta_i^k - h \cdot \frac{\partial J}{\partial \theta_i^k}, \quad (2.18)$$

де h – ітреаційний коефіцієнт, $\frac{\partial J}{\partial \theta_i^k}$ – елементи матриці Якобі.

Тут і надалі верхній індекс k позначає номер ітерації, $i = 1, 2, 3$ – номер кінематичної пари маніпулятора.

На основі (2.14)-(2.16) та (2.17), (2.18) було отримано наступні вирази для оновлення значень кутів зчленувань $\theta_1, \theta_2, \theta_3$ [43]:

$$\theta_1^{k+1} = \theta_1^k + h \cdot \left((x_{des} - x^k) \frac{\partial x}{\partial \theta_1^k} + (y_{des} - y^k) \frac{\partial y}{\partial \theta_1^k} + (z_{des} - \right. \\ \left. - z^k) \frac{\partial z}{\partial \theta_1^k} \right), \tag{2.19}$$

$$\theta_2^{k+1} = \theta_2^k + h \cdot \\ \cdot \left((x_{des} - x^k) \frac{\partial x}{\partial \theta_2^k} + (y_{des} - y^k) \frac{\partial y}{\partial \theta_2^k} \right. \\ \left. + (z_{des} - z^k) \frac{\partial z}{\partial \theta_2^k} \right), \tag{2.20}$$

$$\theta_3^{k+1} = \theta_3^k + h \cdot \\ \cdot \left((x_{des} - x^k) \frac{\partial x}{\partial \theta_3^k} + (y_{des} - y^k) \frac{\partial y}{\partial \theta_3^k} \right. \\ \left. + (z_{des} - z^k) \frac{\partial z}{\partial \theta_3^k} \right), \tag{2.21}$$

де x^k, y^k, z^k – поточні значення координат захвату, що розраховуються на кожній ітерації k з формул (2.14)-(2.16); $\frac{\partial x}{\partial \theta_i^k}$ – елементи матриці Якобі, відомі як часттинні похідні, що визначаються шляхом диференціювання формул (2.14)-(2.16) за відповідним кутом θ_i на кожній ітерації k в циклі.

Продиференціювавши відповідно формули (2.14)-(2.16) отримуємо наступні вирази частинних похідних елементів матитриці Якобі [43, 46]:

$$\frac{\partial x}{\partial \theta_1} = -\sin\theta_1 \, (L_3 \cos(\theta_2 + \theta_3) + L_2 \cos\theta_2), \tag{2.22}$$

$$\frac{\partial x}{\partial \theta_2} = -\cos\theta_1 \, (L_3 \sin(\theta_2 + \theta_3) + L_2 \sin\theta_2), \tag{2.23}$$

$$\frac{\partial x}{\partial \theta_3} = -L_3 \sin(\theta_2 + \theta_3) \cos\theta_1, \tag{2.24}$$

$$\frac{\partial y}{\partial \theta_1} = \cos\theta_1 \, (L_3 \cos(\theta_2 + \theta_3) + L_2 \cos\theta_2), \tag{2.25}$$

$$\frac{\partial y}{\partial \theta_2} = -\sin\theta_1\,(L_3\sin(\theta_2+\theta_3)+L_2\sin\theta_2), \tag{2.26}$$

$$\frac{\partial y}{\partial \theta_3} = -L_3\sin(\theta_2+\theta_3)\sin\theta_1, \tag{2.27}$$

$$\frac{\partial z}{\partial \theta_1} = 0, \tag{2.28}$$

$$\frac{\partial z}{\partial \theta_2} = L_3\cos(\theta_2+\theta_3)+L_2\cos\theta_2, \tag{2.29}$$

$$\frac{\partial z}{\partial \theta_3} = L_3\cos(\theta_2+\theta_3). \tag{2.30}$$

Достатньою точністю позиціонування захвата маніпулятора для таких типів алгоритмів вважається значення, починаючи з 10^{-4} і менше. Задля досягнення поставленого порогу та запобігання появи зацикленості алгоритму доцільним є попереднє обмеження кількості ітерацій до 10^4 ітерацій. Встановлене граничне значення дозволяє запобігти розходженню алгоритму на ранній стадії та є достатнім для забезпечення необхідної точності.

Узагальнюючи вище зазначене на основі формули (2.17) сформульовано наступний вираз умови збіжності алгоритму градієнтного спуску [43, 46]:

$$\left|x_{des}-x(\theta^k)\right|<\varepsilon \wedge \left|y_{des}-y(\theta^k)\right|<\varepsilon \wedge \left|z_{des}-z(\theta^k)\right|<\varepsilon, \tag{2.31}$$

де $k \le 10^4$ – максимальна кількість ітерацій у циклі, $\varepsilon = 10^{-4}$ – значення необхідної точності при досягненні якої алгоритм зупиняється.

Ефективність алгоритму градієнтного спуску суттєво залежить від правильного вибору ітераційного коефіцієнта h у формулах (2.19)-(2.21). Некоректний вибір цього параметра призводить до збільшення кількості ітерацій, зростання часу обчислення та потенційної розбіжності алгоритму. Водночас, застосування адаптивного підбору h на кожній ітерації, хоча й частково вирішує цю проблему, вносить додаткові обчислювальні витрати – що є небажаним для систем керування реального часу на мікроконтролерах з обмеженими ресурсами.

Більш практичним підходом є визначення фіксованого значення h до початку виконання алгоритму, яке залишається незмінним протягом усього

ітераційного процесу, однак є достатньо універсальним для забезпечення високої точності та швидкої збіжності.

В даній кваліфікаційній роботі вибір оптимального фіксованого значення ітераційного коефіцієнту h є ключовою операцією для досягнення основної мети роботи.

Геометричний (аналітичний) метод. Оскільки досліджуваний просторовий триланковий робот-маніпулятор є складною конструкцією, у аналітичному методі здійснюється його декомпозиція на спрощені підсистеми. Зокрема, задача зводиться до послідовного визначення кутів повороту окремих ланок шляхом переходу від тривимірної задачі до плоскої.

Оскільки перша ланка L_1 розташована вертикально вздовж осі z, її внесок у горизонтальну проекцію відсутній, тому вертикальна складова визначається як $(z_{des} - L_1)$. Таким чином, відстань r від кінця першої ланки до цільової точки:

$$r = \sqrt{{x_{des}}^2 + {y_{des}}^2 + (z_{des} - L_1)^2}, \quad (2.32)$$

де $x_{des}, y_{des}, z_{des}$ – бажані координати цільової точки, L_1 – довжина першої ланки маніпулятора.

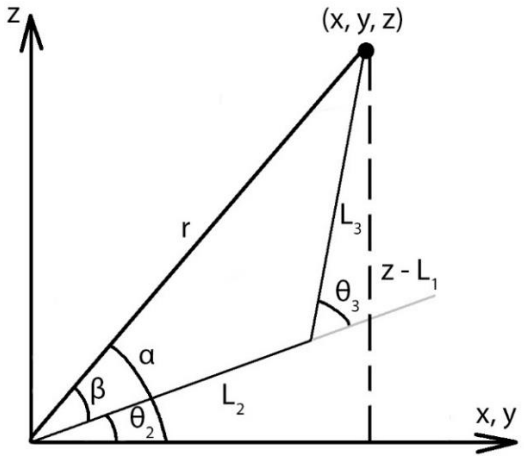


Рис. 2.2. Спрощене представлення кінематичної моделі досліджуваного триланкового маніпулятора

Розглянемо трикутник, утворений ланками L_2 та L_3, а також r (Рис.2.2). Застосовуючи теорему косинусів отримуємо:

$$r^2 = L_2^2 + L_3^2 - 2 \cdot L_2 \cdot L_3 \cdot cos(\pi - \theta_3). \quad (2.33)$$

Оскільки $\cos(\pi - \theta_3) = -\cos(\theta_3)$, отримуємо:

$$r^2 = L_2^2 + L_3^2 + 2 \cdot L_2 \cdot L_3 \cdot cos(\theta_3), \quad (2.34)$$

звідки:

$$cos(\theta_3) = \frac{r^2 - L_2^2 - L_3^2}{2 \cdot L_2 \cdot L_3}. \quad (2.35)$$

Остаточно формула для отримання значення кута θ_3 має наступний вигляд:

$$\theta_3 = \mp arccos(cos(\theta_3)), \quad (2.36)$$

де від'ємне значення кута відповідає конфігурації «лікоть вгору», а додатнє – «лікоть вниз».

Як видно з Рис.2.2 кут θ_2 визначається як різниця кутів α та β у тому ж плоскому трикутнику. Кут α є кутом підйому вектора r відносно горизонтальної площини *x,y*. Довжина вектора r отримується з формули (2.34), а вертикальна складнова – $(z_{des} - L_1)$, тому:

$$\alpha = arcsin\left(\frac{z_{des} - L_1}{r}\right). \quad (2.37)$$

Кут β, що лежить між ланкою L_2 та вктором r, знаходиться з теореми синусів:

$$\frac{L_3 \, sin(\theta_3)}{r} = sin(\beta) \Rightarrow \beta = arcsin\left(\frac{L_3 \, sin(\theta_3)}{r}\right). \quad (2.38)$$

Таким чином остаочна формула для знаходження кута θ_2 має наступний вигляд:

$$\theta_2 = \alpha - \beta = \\ = arcsin\left(\frac{z_{des} - L_1}{r}\right) - arcsin\left(\frac{L_3 \, sin(\theta_3)}{r}\right). \quad (2.39)$$

Для визначення кута повороту бази L_1 вводиться допоміжна функція Хевісайда, що враховує знаки координат x_{des} та y_{des}:

$$H(x) = \begin{cases} 0, & x \leq 0 \\ 1, & \text{інакше} \end{cases}. \quad (2.40)$$

Тоді:

$$\theta_1 = arctan\left(\frac{y_{des}}{x_{des}}\right) + H(-x) \cdot \frac{y}{|y|} \cdot \pi, \quad (2.41)$$

де доданок $H(-x) \cdot \frac{y}{|y|} \cdot \pi$ робить поправку $\pm\pi$, що забезпечує правильне визначення кута у всіх 4-х квадрантах площини XOY. У програмній реалізації визначення θ_1 здійснюється через еквівалентну функцію $atan2\left(\frac{y_{des}}{x_{des}}\right)$.

2.4. Аналіз сингулярних конфігурацій маніпулятора

Сингулярні конфігурації – це положення маніпулятора, у яких матриця Якобі J втрачає повний ранг, тобто $rank\,(J) < 3$ для досліджуваного триланкового маніпулятора. У таких конфігураціях захват втрачає можливість переміщуватися в одному або декількох напрямках декартового простору, а швидкості сполучень можуть необмежено зростати при скінченних швидкостях захвата [50]. Знання сингулярних зон є необхідною умовою для коректного планування траєкторій (п. 2.5) та забезпечення чисельної стійкості алгоритму розв'язання ЗЗК (п. 2.3).

Конфігурація θ вважаться сингулярною за виродження квадратної матриці Якобі [50]:

$$\det(J) = 0. \quad (2.42)$$

Для досліджуваного триланкового маніпулятора матриця Якобі J має наступний вигляд [50]:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_1} & \frac{\partial x}{\partial \theta_2} & \frac{\partial x}{\partial \theta_3} \\ \frac{\partial y}{\partial \theta_1} & \frac{\partial y}{\partial \theta_2} & \frac{\partial y}{\partial \theta_3} \\ \frac{\partial z}{\partial \theta_1} & \frac{\partial z}{\partial \theta_2} & \frac{\partial z}{\partial \theta_3} \end{bmatrix}, \quad (2.43)$$

де елементи $\frac{\partial x}{\partial \theta_i}, \frac{\partial y}{\partial \theta_i}, \frac{\partial z}{\partial \theta_i}$ є частковими похідними координат захвата, що наведені у формулах (2.22)-(2.30).

Детермінант матриці Якобі, отримуємо за допомогою наступних алгебраїчних перетворень:

$$\begin{aligned}\det(J) = \frac{\partial x}{\partial \theta_1}\cdot\frac{\partial y}{\partial \theta_2}\cdot\frac{\partial z}{\partial \theta_3} + \frac{\partial x}{\partial \theta_2}\cdot\frac{\partial y}{\partial \theta_3}\cdot\frac{\partial z}{\partial \theta_1} + \frac{\partial x}{\partial \theta_3}\cdot\frac{\partial y}{\partial \theta_1}\cdot\frac{\partial z}{\partial \theta_2} - \\ - \frac{\partial z}{\partial \theta_1}\cdot\frac{\partial y}{\partial \theta_2}\cdot\frac{\partial x}{\partial \theta_3} - \frac{\partial z}{\partial \theta_2}\cdot\frac{\partial y}{\partial \theta_3}\cdot\frac{\partial x}{\partial \theta_1} - \frac{\partial z}{\partial \theta_3}\cdot\frac{\partial y}{\partial \theta_1}\cdot\frac{\partial x}{\partial \theta_2} = \\ = (L_3\, cos(\theta_2+\theta_3) + L_2\, cos(\theta_2))\cdot \\ \cdot \big({L_3}^2\, sin(\theta_2+\theta_3)\, cos(\theta_2+\theta_3) + \\ + L_2 L_3\, sin(\theta_2)\, cos(\theta_2+\theta_3) - \\ - {L_3}^2\, sin(\theta_2+\theta_3)\, cos(\theta_2+\theta_3) - \\ - L_2 L_3\, cos(\theta_2)\, sin(\theta_2+\theta_3)\big) = \\ = L_2 L_3\, sin(\theta_3)\cdot(L_3\, cos(\theta_2+\theta_3) + L_2\, cos(\theta_2)).\end{aligned} \tag{2.44}$$

Отже, остаточно формула матриці Якобі досліджуваного робота-маніпулятора має наступний вигляд [50, 46]:

$$\det(J) = L_2 L_3\, sin(\theta_3)\cdot(L_3\, cos(\theta_2+\theta_3) + L_2\, cos(\theta_2)). \tag{2.45}$$

З виразу (2.45) випливає, що умова (2.42) виконується при двох незалежних геометричних умовах, кожна з яких породжує окремий тип кінематичної сингулярності [50].

*Лікть*ова *сингулярність.* Виникає коли перший множник у (2.45) звертається в нуль [50, 46]:

$$sin(\theta_3) = 0 \Longrightarrow \theta_3 = 0 \text{ або } \theta_3 = \pi. \tag{2.46}$$

Умова (2.46) відповідає конфігураціям, у яких ланки L_2 та L_3 є повністю витягнутими ($\theta_3 = 0$) або складаними у площині руху «плеча» і «ліктя». В обох випадках другого і третього сполучення виявляються кінематично еквівалентними – їх осі обертання стають паралельними, а маніпулятор втрачає один ступінь керованості у вертикальній площині. Геометрично даний вид

сингулярності відповідає граничним точкам робочого простору – максимальному та мінімальному вильоту захвата.

Сингулярність плеча. Виникає коли другий множник у (2.45) звертається в нуль [50, 46]:

$$L_3\,cos(\theta_2+\theta_3)+L_2\,cos(\theta_2)=0. \tag{2.47}$$

З виразів (2.14)-(2.15) видно, що умова (2.47) рівносильна зверненню в нуль горизонтальної складової положення захвата:

$$p_x=p_y=0. \tag{2.48}$$

Геометрично це означає, що захват знаходиться безпосередньо на вертикальній осі z_0 першого сполучення. У такій конфігурації кут θ_1 є невизначеним – будь-яке значення азимутального кута дає те саме положення захвата, і маніпулятор має нескінченну кількість розв'язків ЗЗК [50]. Фізично даний тип сингулярності виникає, коли центр зап'ястя знаходиться на осі обертання основи, і є неминучою для маніпуляторів без конструктивного зміщення ланок.

2.5. Планування траєкторії руху робочого органу маніпулятора

Оскільки кубічний поліном (1.11)-(1.14) не дозволяє задати прискорення у вузлових точках, для планування траєкторії маніпулятора в даній роботі використовується квінтичний поліном (1.15). Це забезпечує безперервність положення, швидкості та прискорення на всьому інтервалі руху.

У загальному випадку коефіцієнти полінома (1.15) визначаються з системи шести граничних умов: задані положення θ_0, θ_{end}, швидкість $\dot{\theta}_0, \dot{\theta}_{end}$ та прискорення $\ddot{\theta}_0, \ddot{\theta}_{end}$ на початку t_0 і в кінці t_{end} сегмента. Підстановка цих умов у вирази (1.15)-(1.17) дає наступну систему у матричній формі [51, 52]:

$$\begin{bmatrix} t_0^5 & t_0^4 & t_0^3 & t_0^2 & t_0 & 1 \\ t_{end}^5 & t_{end}^4 & t_{end}^3 & t_{end}^2 & t_{end} & 1 \\ 5t_0^4 & 4t_0^3 & 3t_0^2 & 2t_0 & 1 & 0 \\ 5t_{end}^4 & 4t_{end}^3 & 3t_{end}^2 & 2t_{end} & 1 & 0 \\ 20t_0^3 & 12t_0^2 & 6t_0 & 2 & 0 & 0 \\ 20t_{end}^3 & 12t_{end}^2 & 6t_{end} & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_{end} \\ \dot{\theta}_0 \\ \dot{\theta}_{end} \\ \ddot{\theta}_0 \\ \ddot{\theta}_{end} \end{bmatrix}, \quad (2.49)$$

Матриці (2.49) в загальному вигляді описуються наступним виразом:

$$B\boldsymbol{X} = \boldsymbol{C}, \quad (2.50)$$

де x – вектор шуканих коефіцієнтів поліному (1.15); B – матриця граничних умов розміром 6×6, рядки якої відповідають значенням полінома та його похідних (1.16), (1.17) у початковій та кінцевій точках сегмента; c – вектор граничних умов що містить задані положення швидкості та прискорення на межах сегмента.

З виразу (2.50) знаходимо значення шуканих коефіцієнтів поліному наступним чином:

$$B^{-1}\boldsymbol{C} = \boldsymbol{X}. \quad (2.51)$$

Оскільки у даній роботі загальний час руху t_f рівномірно розбивається на $j-1$ сегментів між j вузловими точками, а для кожного сегмента використовується локальний час $\tau = t - t_k$, де t_k – момент початку сегмента, то задля спрощення розрахунків доцільно ввести умову $t_0 = 0$ [46]. Як наслідок (2.51) набуває вигляду:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ t_{end}^5 & t_{end}^4 & t_{end}^3 & t_{end}^2 & t_{end} & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5t_{end}^4 & 4t_{end}^3 & 3t_{end}^2 & 2t_{end} & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20t_{end}^3 & 12t_{end}^2 & 6t_{end} & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_5 \\ a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} \theta_0 \\ \theta_{end} \\ \dot{\theta}_0 \\ \dot{\theta}_{end} \\ \ddot{\theta}_0 \\ \ddot{\theta}_{end} \end{bmatrix}, \quad (2.52)$$

Таким чином три з 6-ти коефіцієнтів визначаються безпосередньо з початкових умов і залишається розв’язати систему лише для a_3, a_4, a_5 через

кінцеві умови. Тоді коефіцієнти поліному (1.15) можна знайти наступним чином [53]:

$$a_0 = \theta_0, \tag{2.53}$$

$$a_1 = \dot{\theta}_0, a_2 = \frac{\ddot{\theta}_0}{2}, \tag{2.54}$$

$$a_3 = \frac{20\theta_{end} - 20\theta_0 - \left(8\dot{\theta}_{end} + 12\dot{\theta}_0\right)t_{end} - \left(3\ddot{\theta}_0 + 12\ddot{\theta}_{end}\right)t_{end}^2}{2t_{end}^3}, \tag{2.55}$$

$$a_4 = \frac{30\theta_0 - 30\theta_{end} - \left(14\dot{\theta}_{end} + 16\dot{\theta}_0\right)t_{end} - \left(3\ddot{\theta}_0 + 2\ddot{\theta}_{end}\right)t_{end}^2}{2t_{end}^4}, \tag{2.56}$$

$$a_5 = \frac{12\theta_{end} - 12\theta_0 - (6\dot{\theta}_{end} + 6\dot{\theta}_0)t_{end} - (\ddot{\theta}_0 + \ddot{\theta}_{end})t_{end}^2}{2t_{end}^5}. \tag{2.57}$$

2.6. Висновки до розділу 2

Розроблена математична модель руху послідовного триланкового маніпулятора охоплює кінематичний аналіз, вирішення зворотної задачі кінематики, аналіз сингулярних конфігурацій та планування траєкторії руху робочого органу.

Побудовано кінематичну схему маніпулятора з використанням ДХ-представлення та отримано матриці однорідного перетворення для кожної ланки. На їх основі розв’язано пряму задачу кінематики та виведено аналітичні вирази для координат вектора положення захвата.

Для вирішення зворотної задачі кінематики реалізовано два підходи. Перший – чисельний метод градієнтного спуску з транспонованою матрицею Якобі, що зводиться до мінімізації цільової функції шляхом ітераційного оновлення кутів зчленувань. Виявлено, що ефективність алгоритму суттєво залежить від вибору ітераційного коефіцієнта h. Другий – геометричний (аналітичний) метод, що базується на декомпозиції просторової задачі до плоскої із послідовним визначенням кутів $\theta_3, \theta_2, \theta_1$.

Проведено аналіз сингулярних конфігурацій маніпулятора на основі детермінанта матриці Якобі. Виявлено два типи сингулярностей: ліктьова – при $\theta_3 = 0$ або $\theta_3 = \pi$, коли ланки повністю витягнуті або складені, та сингулярність плеча – коли захват знаходиться на вертикальній осі першого сполучення. Знання цих зон є необхідною умовою для коректного планування траєкторій та забезпечення чисельної стійкості алгоритму ЗЗК.

Для планування траєкторії руху робочого органу обрано квінтичний поліном п’ятого ступеня, який на відміну від кубічного забезпечує безперервність не лише положення і швидкості, але й прискорення. З урахуванням умови $t_0 = 0$ для локального часу кожного сегмента отримано спрощені аналітичні вирази для шести коефіцієнтів полінома, три з яких визначаються безпосередньо з початкових умов.

РОЗДІЛ 3

ІФОРМАЦІЙНО-ОБЧИСЛЮВАЛЬНИЙ КОМПЛЕКС ДЛЯ МОДЕЛЮВАННЯ РУХУ РОБОТА-МАНІПУЛЯТОРА

3.1. Загальний алгоритм функціонування програмного комплексу та його архітектура

Розроблений інформаційно-обчислювальний комплекс реалізовано у середовищі MATLAB з використанням інструментарію GUIDE (Graphical User Interface Development Environment) та пакету Robotics Toolbox [58]. Комплекс являє собою єдиний програмний застосунок з графічним інтерфейсом користувача, архітектура якого побудована за рівневим принципом: кожен рівень включає в себе декілька програмних модулів із чітко визначеними вхідними та вихідними параметрами, що забезпечує їх незалежність між собою та спрощує подальше розширення або модифікацію. Загальна структурна схема комплексу наведена на Рис. 3.1.

Рівень графічного інтерфейсу. Цей рівень реалізований через систему функцій зворотного виклику (callbacks) і точкою взаємодії між користувачем і обчислювальним ядром системи. Ініціалізацію застосунку виконує функція Project_OpeningFcn, яка встановлює початковий стан системи та задає фіксоване початкове положення маніпулятора $\theta^0 = \left(-\frac{2\pi}{3}, \frac{\pi}{6}, -\frac{5\pi}{36}\right)^T$, що зберігається у полі handles.current_position. Введення геометричних параметрів маніпулятора – довжин ланок L_1, L_2, L_3; значень кутів θ_1, θ_2, θ_3 ; координат точок X, Y, Z здійснюється через текстові поля GUI. Обмеження допустимих діапазонів руху кутів сполучень $\theta_{1\_min}, \theta_{1\_max}$, θ_{2_min}, $\theta_{2\_max}, \theta_{3\_min}, \theta_{2\_max}$ задаються окремими полями введення для кожного зчленування і перевіряються перед кожним обчисленням.

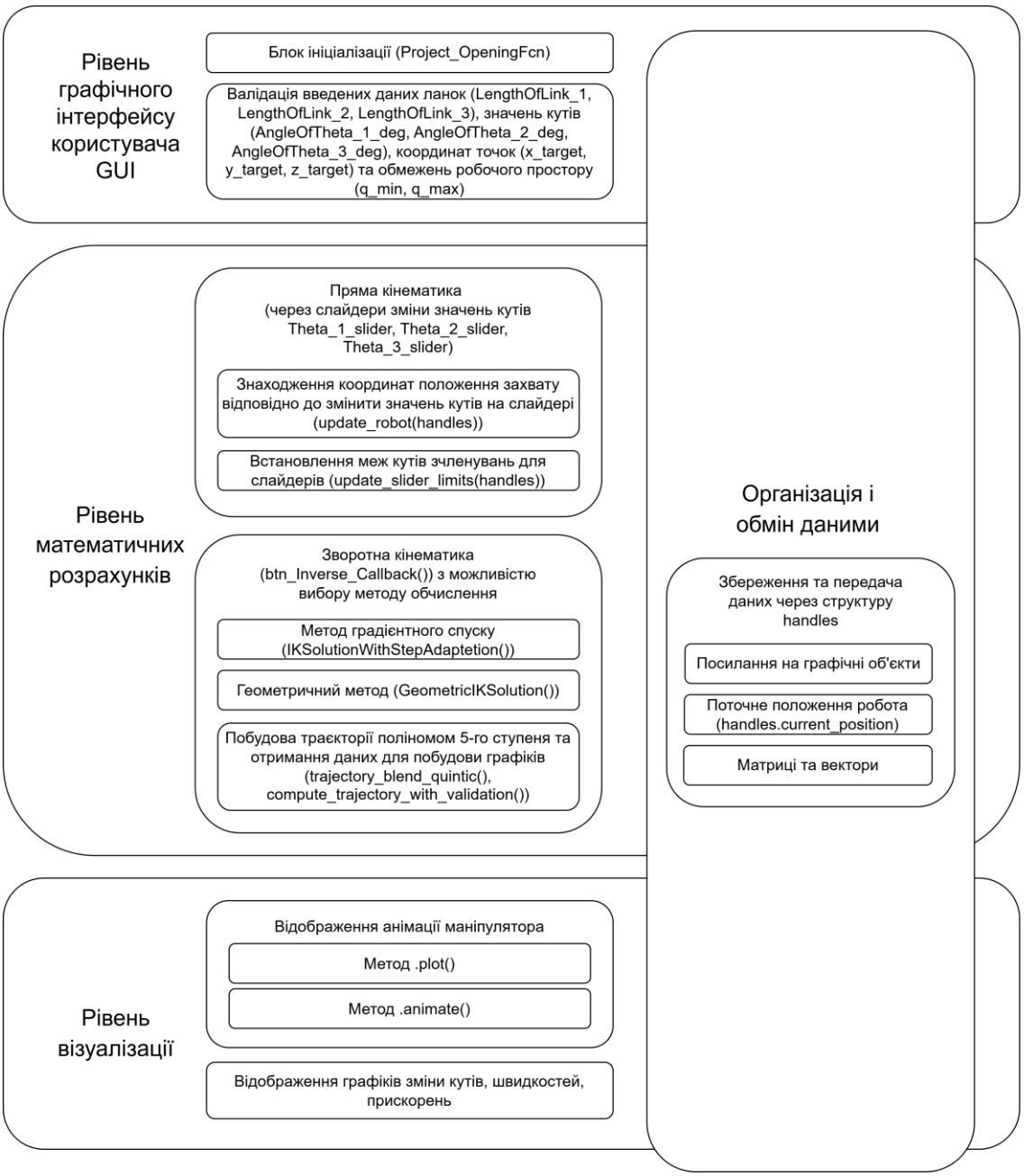


Рис. 3.1. Архітектура програмного комплексу

Рівень математичних розрахунків. Цей рівень є обчислювальним ядром системи і містить головні модулі всіх алгоритмів, що включають кінематичний аналіз маніпулятора.

Модуль прямої кінематики реалізує автоматичне обчислення у реальному часі через дві взаємопов'язані функції. Функція update_slider_limits() викликається при зміні будь-якого поля обмежень діапазонів руху зчленувань і

оновлює властивості Min та Max відповідних слайдерів; за порожніх полів встановлюються стандартні межі [−180°; 180°]. Функція update_robot() зчитує поточні значення кутів зі слайдерів, конвертує їх у радіани та будує кінематичну модель через об’єкт SerialLink, після чого методом fkine() розв’язується пряма задача кінематики і координати захвата миттєво відображаються у полях Pos_X, Pos_Y, Pos_Z без необхідності додаткового підтвердження з боку користувача.

Модуль зворотної кінематики (оброблювач btn_Inverse_Callback()) реалізує гібридну стратегію розв'язання ЗЗК. Першим виконується ітераційний метод градієнтного спуску (IKSolutionWithStepAdaptetion()), що забезпечує конфігурацію «лікоть вгору». Якщо результат є порожнім або знайдені кути виходять за допустимі діапазони, користувачеві пропонується перейти до геометричного методу (GeometricIKSolution()), який надає аналітичний розв'язок для обох конфігурацій. Кінцеві кути зберігаються у thetaFinal та відображаються у GUI.

Після успішного розв’язання ЗЗК автоматично виконується планування траєкторії між поточною конфігурацією handles.current_position та thetaFinal: підфункція trajectory_blend_quintic() розраховує коефіцієнти квінтичного полінома з нульовими граничними умовами за швидкістю та прискоренням, а compute_trajectory_with_validation() дискретизує траєкторію на 50 точок при tf = 10 с та перевіряє допустимість кутів зчленувань. Якщо частка порушень перевищує 30 % – користувачеві виводиться попередження, проте анімація виконується в обох випадках.

Рівень візуалізації. Цей рівень відповідає за графічне відображення результатів усіх обчислювальних модулів і є вбудованим у різні модулі. Анімація маніпулятора реалізована через методи .plot() та .animate() об’єкта пакету Robotics Toolbox: спочатку заздалегідь обчислюються декартові координати всіх 50 точок траєкторії захвата через fkine(), після чого в циклі анімації послідовно оновлюється положення маніпулятора та слід траєкторії захвата. Відображення траєкторії здійснюється засобами scatter3() з використанням кольорового

кодування кожної точки за спектральною палітрою jet – колір плавно змінюється від початкової до кінцевої точки, що дозволяє візуально оцінити напрямок та характер руху. Межі тривимірного графічного вікна фіксуються симетрично відносно початку координат із радіусом, рівним сумарній довжині ланок L_{sum}, що забезпечує стабільний масштаб при будь-якій конфігурації маніпулятора.

Результати кінематичного аналізу виводяться на три вбудованих графіки-осі: графіки зміни кутів $\theta_1, \theta_2, \theta_3$, кутових швидкостей $\omega_1, \omega_2, \omega_3$ та кутових прискорень $\alpha_1, \alpha_2, \alpha_3$ кожного сполучення вздовж траєкторії у функції часу.

Організація передачі даних між рівнями. Центральним елементом обміну даними в архітектурі MATLAB GUIDE є структура handles, яка передається між усіма функціями зворотного виклику. Вона зберігає посилання на всі графічні об’єкти інтерфейсу, поточний стан маніпулятора (handles.current_position) для забезпечення неперервності рухів між послідовними викликами, а також проміжні обчислювальні результати. Оновлення структури здійснюється через механізм guidata(hObject, handles) після кожного виклику, що гарантує актуальність стану для наступного модуля. Передача параметрів між модулями реалізована через структури даних: paramsLenghtOfJoins (довжини ланок), paramsPositionOfDesiredPoint (координати цільової точки), paramsStartPositionThetaInRad (початкова конфігурація) та paramsMotionRange (допустимі діапазони сполучень), що виключає використання глобальних змінних.

Загальний алгоритм функціонування комплексу наведено на Рис. 3.2.

При запуску застосунку виконується функція Project_OpeningFcn, яка ініціалізує GUI через вбудовані функції MATLAB та встановлює початкове положення маніпулятора handles.current_position = $\left(-\frac{2\pi}{3}, \frac{\pi}{6}, -\frac{5\pi}{36}\right)^{\top}$, що відповідає початковим значенням кутів сполучень $\theta_1^0, \theta_2^0, \theta_3^0$. Поля GUI заповнюються попередніми значеннями довжин ланок LengthOfLink_1, LengthOfLink_2, LengthOfLink_3 (L_1, L_2, L_3) та кутів сполучень AngleOfTheta_1,

AngleOfTheta_2, AngleOfTheta_3 ($\theta_1^0, \theta_2^0, \theta_3^0$). Стан застосунку зберігається у структурі handles через механізм guidata(hObject, handles).

Після ініціалізації застосунок переходить у режим очікування дій користувача. Залежно від активованого компонента користувацького інтерфейсу GUI запускається відповідний оброблювач події:

- зміна значень у слайдерах кутів «Theta_1», «Theta_2», «Theta_3» – запускає функцію update_robot(handles), що миттєво обчислює пряму кінематику через fkine() та виводить у поля GUI актуальні координати захвата X, Y, Z відповідно до поточних значень $\theta_1, \theta_2, \theta_3$ та довжин ланок L_1, L_2, L_3.
- кнопка «Зворотна кінематика» – запускає оброблювач btn_Inverse_Callback(), що зчитує з GUI координати цільової точки $(x_{des}, y_{des}, z_{des})$ та параметри маніпулятора, виконує розв'язання зворотної задачі кінематики; здійснює побудову траєкторії через функції trajectory_blend_quintic() та compute_trajectory_with_validation(); відображає анімацію руху, графіки зміни кутів $\theta_i(t)$, кутових швидкостей $\omega_i(t)$ та прискорень $\alpha_i(t)$.
- введення значень допустимих діапазонів кутів у поля «Th1_workspase_min», «Th1_workspase_max», «Th2_workspase_min», «Th2_workspase_max», «Th3_workspase_min», «Th3_workspase_max» – викликає функцію update_slider_limits(handles), яка автоматично обмежує значення параметрів «min» та «max» слайдерів. За пустих значень за замовчуванням застосовуються стандартні значення меж кутів сполучень у $[-180°;\ 180°]$.

Користувач має змогу вводити значення координат бажаної точки не тільки використовуючи відповідні слайдери, але й самостійно у відповідні поля. В такому разі алгоритм проводить додаткову перевірку на досяжність точки шляхом розрахунку можливого максимального та мінімального радіуса робочої області маніпулятора з урахуванням геометричних параметрів ланок і

встановлених обмежень на кути повороту суглобів. Якщо задана точка знаходиться поза межами досяжної робочої області, користувачеві виводиться текст повідомлення помилки і обчислення оберненої кінематики не виконується. У протилежному випадку здійснюється розрахунок кутів суглобів та оновлення положення моделі.

Робота комплексу продовжується до закриття вікна застосунку користувачем, після чого всі процеси завершуються.

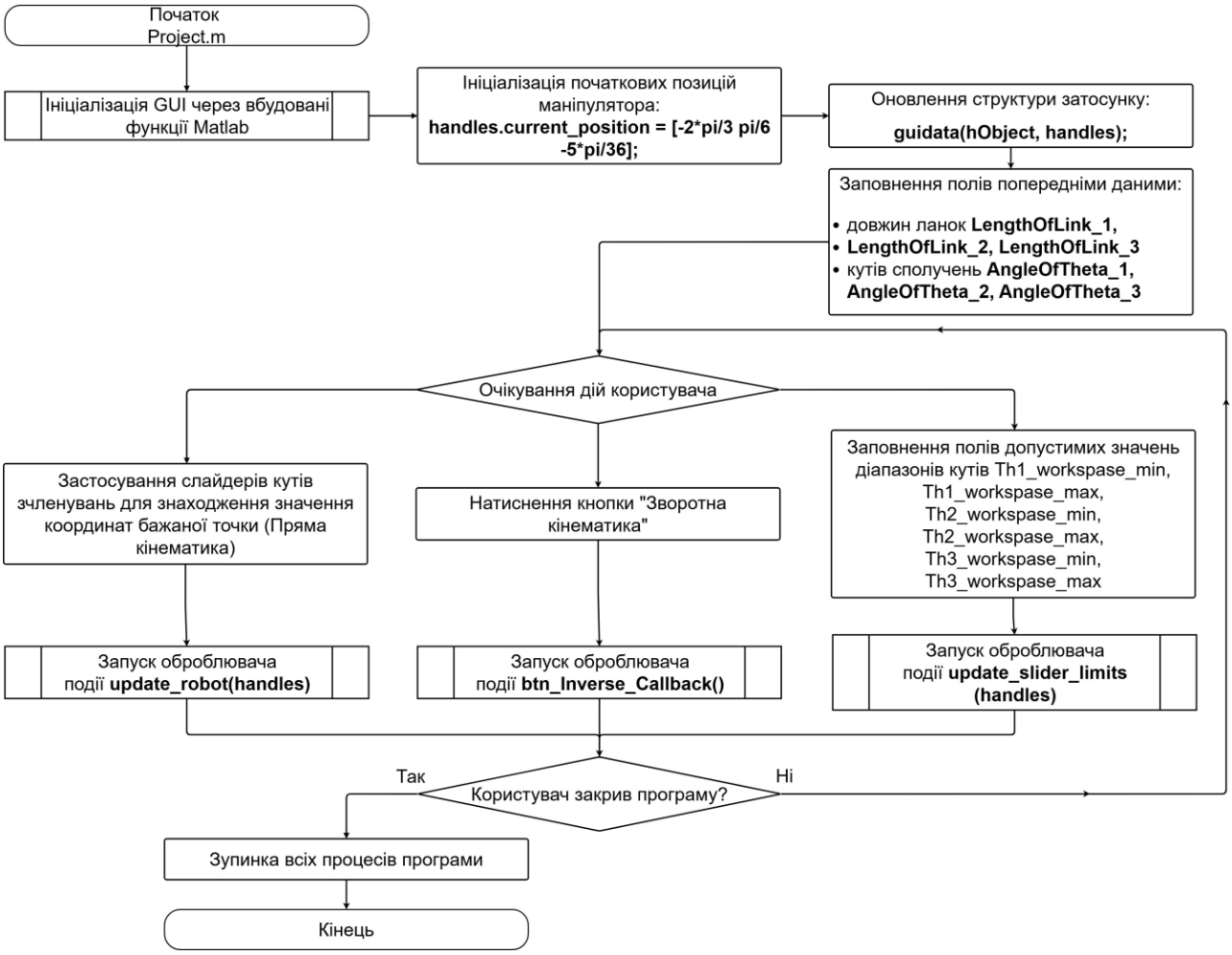


Рис. 3.2. Блок-схема загального алгоритму функціонування інформаційно-обчислювального комплексу

3.2. Модуль прямої кінематики

Модуль прямої кінематики реалізований без окремої кнопки підтвердження – обчислення виконуються автоматично у реальному часі через дві взаємопов’язані допоміжні підфункції: update_slider_limits() та update_robot() (Рис. 3.3-3.4).

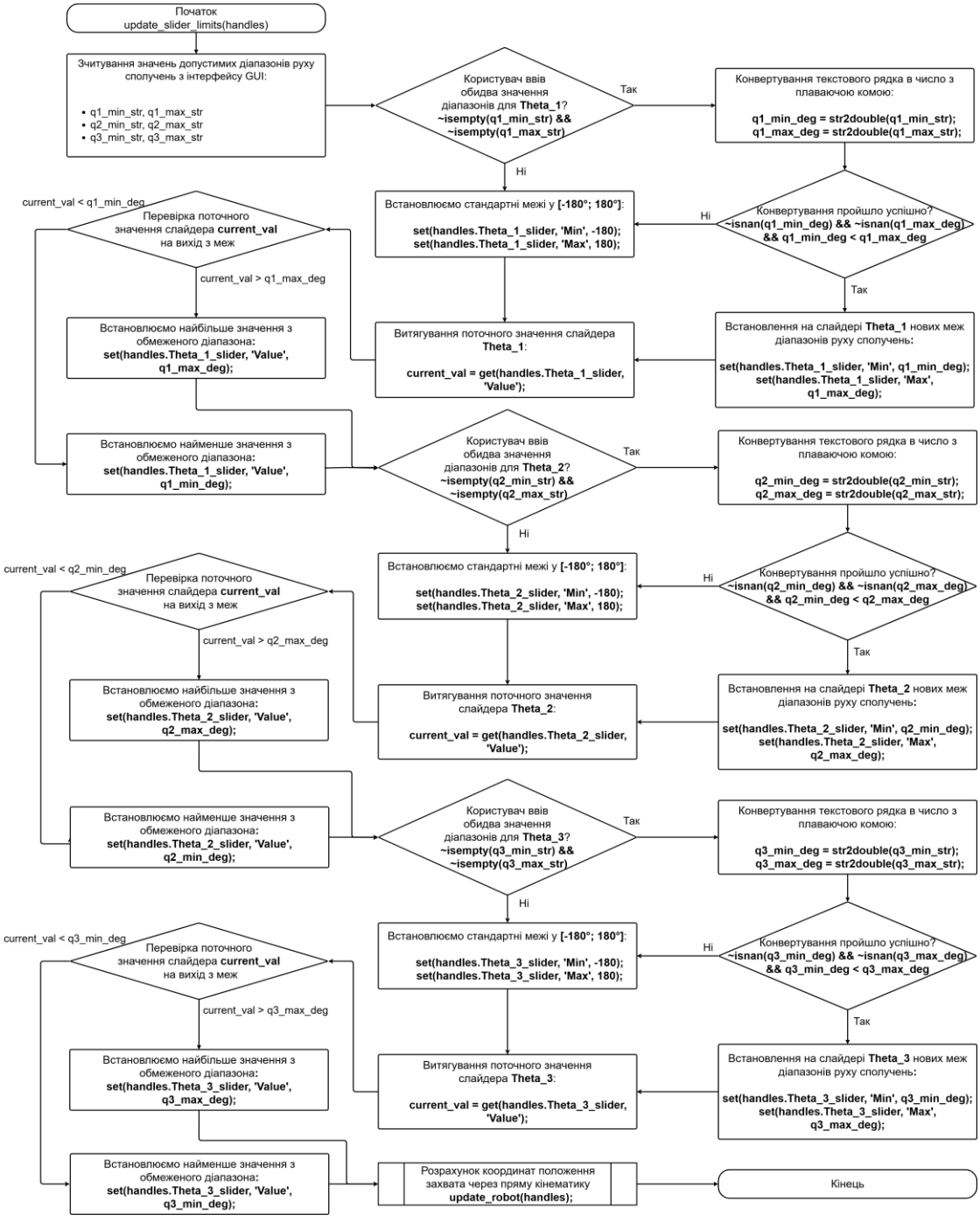


Рис. 3.3. Блок-схема роботи підфункції update_slider_limits(handles)

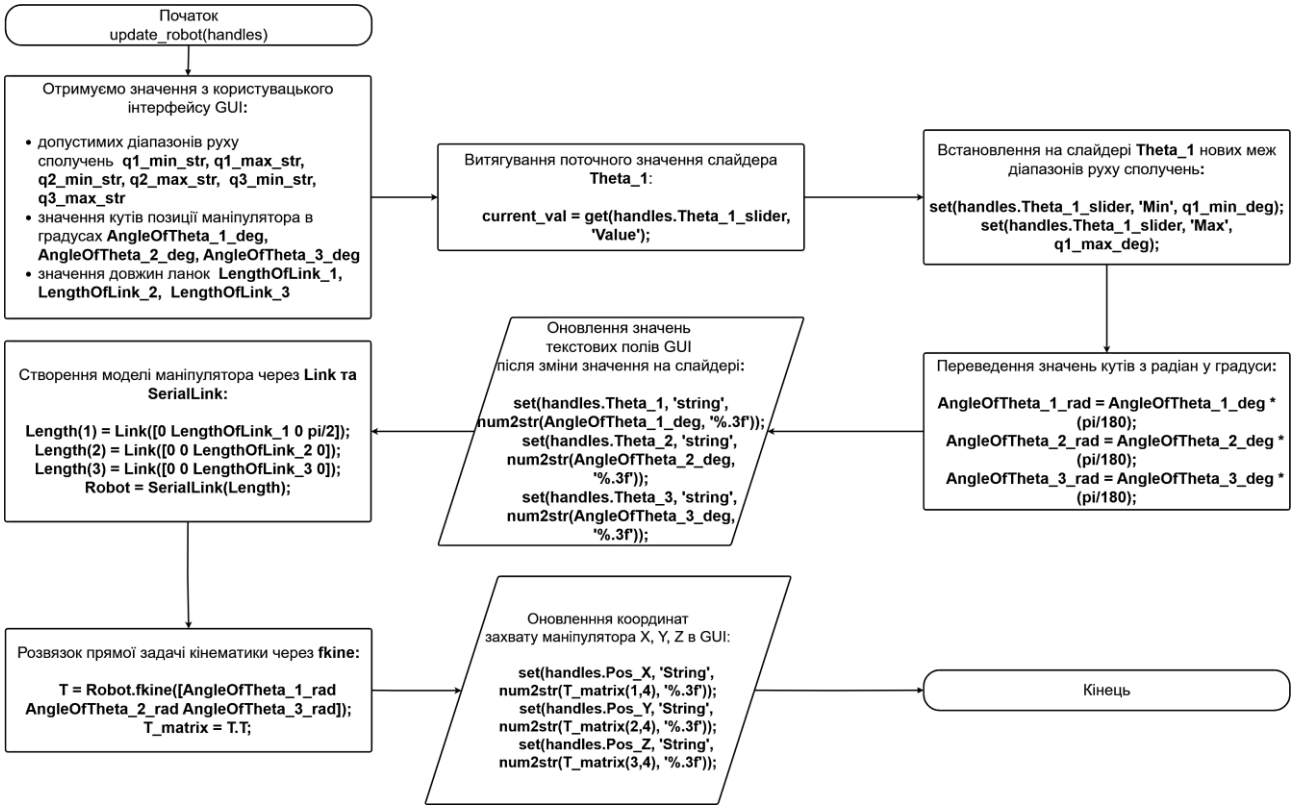


Рис. 3.4. Блок-схема роботи підфункції update_robot(handles) обчислення прямої кінематики

Підфункція обмеження значень допустимих значень руху сполучень update_slider_limits(handles).

Підфункція update_slider_limits(handles) викликається при зміні будь-якого поля обмежень діапазонів руху зчленувань і виконує оновлення властивостей «Min» та «Max» відповідних слайдерів Theta_1_slider, Theta_2_slider, Theta_3_slider.

Загальна послідовність виконання команд у даній функції наведена на Рис. 3.3.

Для кожного зчленування перевіряється коректність введених меж: якщо поля не порожні та містять числові значення, що задовольняють умову q_min <

q_max, – межі слайдера оновлюються відповідно до введених значень, а поточне положення повзунка затискається у допустимому діапазоні. Якщо поля обмежень порожні, то встановлюються стандартні межі у $[-180°;\ 180°]$. Після оновлення меж усіх трьох слайдерів автоматично викликається функція update_robot().

Підфункція розв'язання прямої задачі кінематики update_robot(handles).

Підфункція update_robot(handles) є основним обчислювальним ядром модуля і реалізує розв'язання прямої задачі кінематики згідно з теоретичними положеннями п.п. 2.2.1. Блок-схема роботи модуля наведена на Рис. 3.4.

Послідовність виконання така: зчитуються поточні значення кутів зі слайдерів у градусах (AngleOfTheta_1_deg, AngleOfTheta_2_deg, AngleOfTheta_3_deg) після чого значення конвертуються у радіани. Далі зчитуються довжини ланок L_1, L_2, L_3 з відповідних полів GUI та будується кінематична модель маніпулятора як об'єкт класу *SerialLink* засобами Robotics Toolbox із ДХ-параметрами відповідно до Табл. 2.1. Пряма задача кінематики розв'язується через метод fkine() об'єкта *SerialLink*, що реалізує послідовне множення матриць однорідного перетворення (1.1) за формулою (1.2), де кожна матриця A_i^{i-1} формується автоматично на основі переданих ДХ-параметрів. З отриманої результуючої матриці T знаходиться вектор положення захвата $p = (p_x, p_y, p_z)^T$ відповідно до структури матриці (1.3) як елементи четвертого стовпця T_matrix(1,4), T_matrix(2,4), T_matrix(3,4), значення яких негайно відображаються у полях Pos_X, Pos_Y, Pos_Z користувацького ітерфейсу GUI з точністю до 3-х знаків після коми. Таким чином, будь-яке переміщення слайдера або зміна поля обмежень спричиняє миттєве оновлення координат захвата без необхідності додаткового підтвердження з боку користувача. Такий підхід прискорює обчислення та спрощує взаємодію користувача з інтерфейсом обчислювального комплексу.

3.3. Модуль зворотної кінематики

У розробленому комплексі реалізовано два методи розв'язання ЗЗК: метод градієнтного спуску з підбором кроку (основний) та геометричний метод (резервний/альтернативний). Основним є ітераційний метод, оскільки він не залежить від конфігурації маніпулятора і в подальшому може бути розширений на системи з більшою кількістю ступенів вільності. Геометричний метод доступний користувачу як альтернативний і активується у двох випадках: перший – коли ітераційний метод не забезпечує необхідної точності (порушено умову збіжності за формулою (2.30)), другий – коли отримані кути виходять за межі допустимих діапазонів.

Модуль зворотної кінематики реалізований у вигляді трьох взаємопов'язаних функцій: головного оброблювача події btn_Inverse_Callback, функції ітераційного методу IKSolutionWithStepAdaptetion та функції геометричного методу GeometricIKSolution. Загальна логіка взаємодії між цими функціями відображена на Рис. 3.5-3.6, тоді як детальні алгоритми кожного з методів наведені на Рис. 3.7 та Рис. 3.8 відповідно.

Головний оброблювач події btn_Inverse_Callback.

Функція btn_Inverse_Callback є точкою входу модуля зворотньої кінематики та викликається при натисканні відповідної кнопки у GUI-інтерфейсі. Загальна послідовність виконання команд у даній функції наведена на Рис. 3.5-3.6.

На початку виконання здійснюється зчитування вхідних даних з GUI: координат цільової точки захвату x_target, y_target, z_target ($x_{des}, y_{des}, z_{des}$) та довжин ланок маніпулятора LengthOfLink_1, LengthOfLink_2, LengthOfLink_3 (L_1, L_2, L_3). Отримані значення пакуються у структури paramsLenghtOfJoins та paramsPositionOfDesiredPoint для передачі у підфункції. Також здійснюється розрахунок загальної довжини маніпулятора у змінній totalLenghtOfLinght. Початкове положення маніпулятора береться з поля handles.current_position, що

зберігає кути сполучень після попереднього руху; якщо поле ще не існує – за замовчуванням використовуються попередньо фіксовані значення $\theta_1 = -\frac{2\pi}{3}, \theta_2 = \frac{\pi}{6}, \theta_3 = -\frac{5\pi}{36}$.

Перед викликом ітераційного алгоритму виконується попередня перевірка на досяжність цільової точки. Обчислюються допоміжні величини:

$$R = \sqrt{x_{des}^2 + y_{des}^2}, \qquad h = \sqrt{R^2 + (z_{des} - L_1)^2}, \tag{3.1}$$

де R – горизонтальна відстань від осі z_0 до цільової точки, h – просторова відстань від початку другої ланки до цільової точки. Якщо виконується будь-яка з умов $h \leq |L_2 - L_3| + 0{,}001, h \geq L_2 + L_3 - 0{,}001, R \leq |L_2 - L_3| + 0{,}001$ або $R \geq L_2 + L_3 - 0{,}001$, точка вважається недосяжною і виконання функції припиняється з відображенням відповідного повідомлення. За успішного проходження перевірки здійснюється виклик підфункції IKSolutionWithStepAdaptetion. Результат зберігається у масиві thetaFromIK. У разі якщо ітераційний метод не досяг збіжності (масив порожній, isempty(thetaFromIK)) або отримані кути виходять за задані допустимі межі зчленувань (outOfRange = true), формується повідомлення failureReason і користувачу пропонується виконати розрахунок геометричним методом через підфункцію GeometricIKSolution.

Остаточні значення кутів зберігаються у змінній thetaFinal та відображаються у відповідних полях GUI і встановлюються у слайдерах. Після цього виконується:

- побудова кінематичної моделі маніпулятора через дані об’єкту класу *SerialLink*;
- формування граничних умов та планування траєкторії руху між поточним q_start і цільовим q_end положеннями на основі квінтичного полінома за допомогою підфункції trajectory_blend_quintic();
- дискретизація траєкторії на 50 точок при загальному часі руху tf = 10 с;

- перевірка кожної точки траєкторії на відп овідність допустимим межам зчленувань за допомогою підфункції compute_trajectory_with_validation();
- анімація руху маніпулятора у тривимірному графічному вікні;
- візуалізація траєкторії руху робочого органа кольоровими точками засобами функції scatter3() із використанням палітри jet;
- оновлення поточного положення маніпулятора у змінній handles.current_position після завершення анімації;
- побудова графіків зміни кутів $\theta_i(t)$, кутових швидкостей $\omega_i(t)$ та прискорень $\alpha_i(t)$ для кожного шарніра.

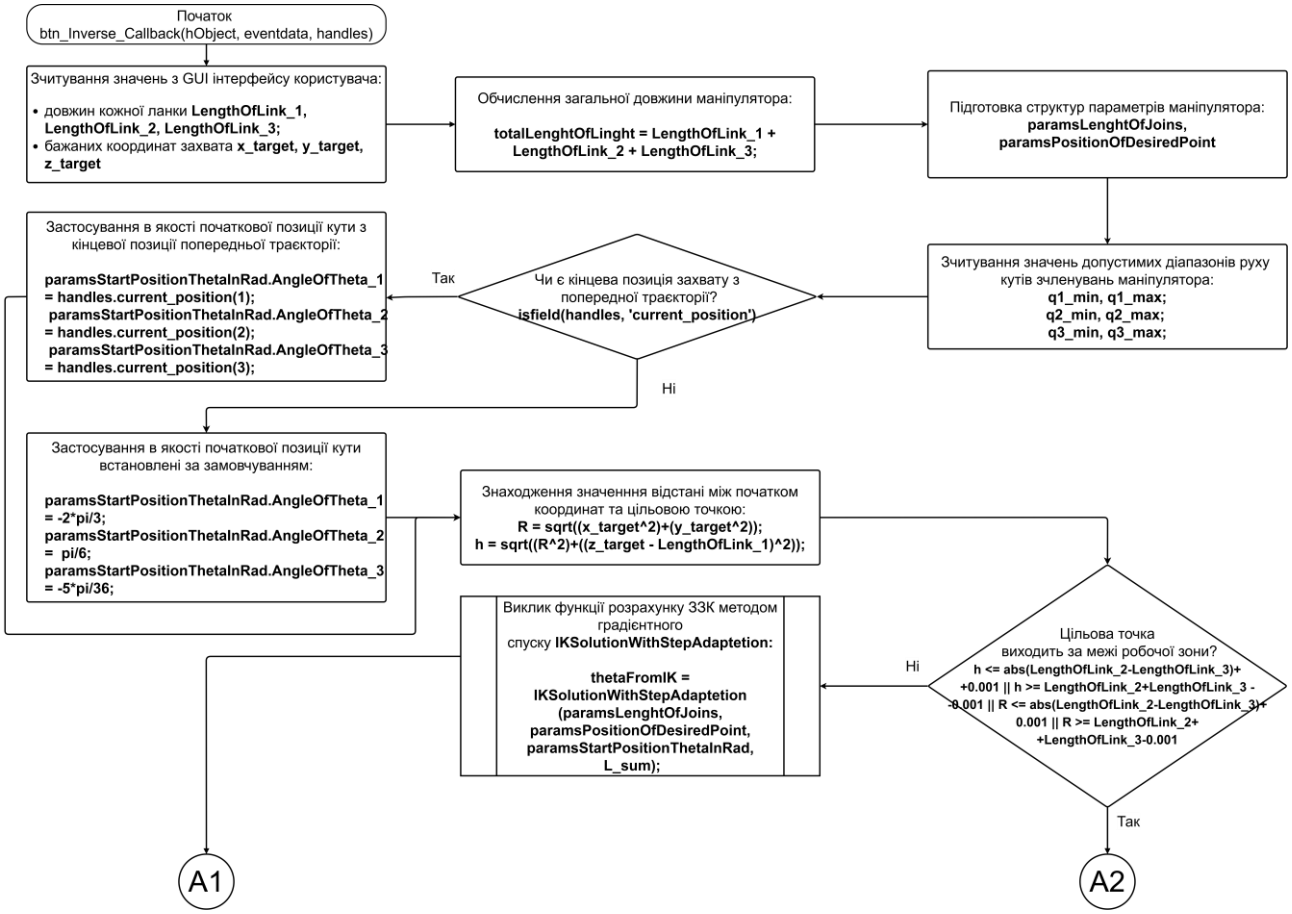


Рис. 3.5. Блок-схема загальної роботи модуля вирішення ЗЗК (головного оброблювача події btn_Inverse_Callback()) (Частина 1)

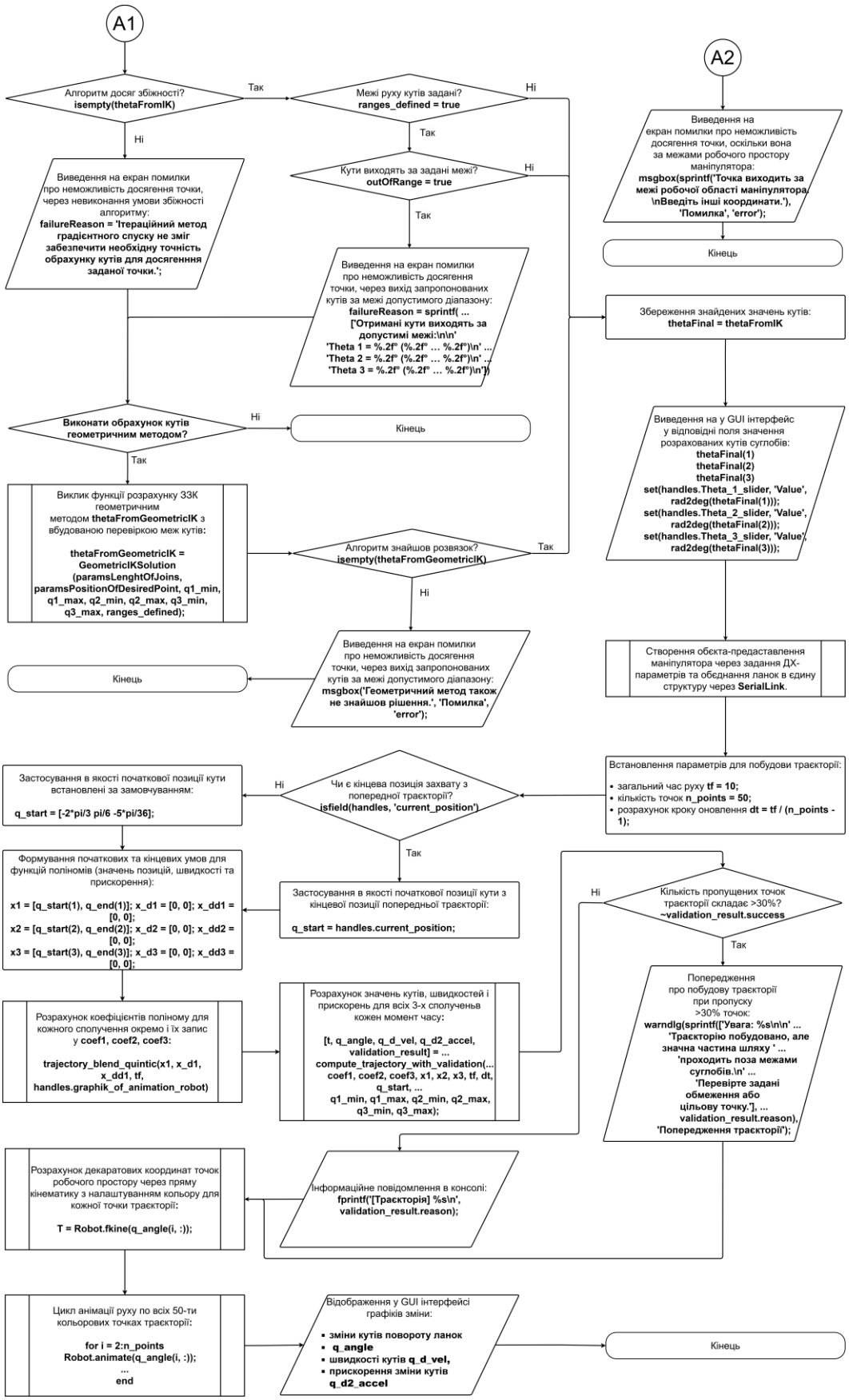


Рис.3.6. Блок-схема загальної роботи модуля вирішення ЗЗК (головного оброблювача події btn_Inverse_Callback()) (Частина 2)

Підфункція ітераційного методу IKSolutionWithStepAdaptetion.

Підфункція IKSolutionWithStepAdaptetion реалізує метод градієнтного спуску для розв'язання зворотної задачі кінематики. Детальна блок-схема алгоритму наведена на Рис. 3.7.

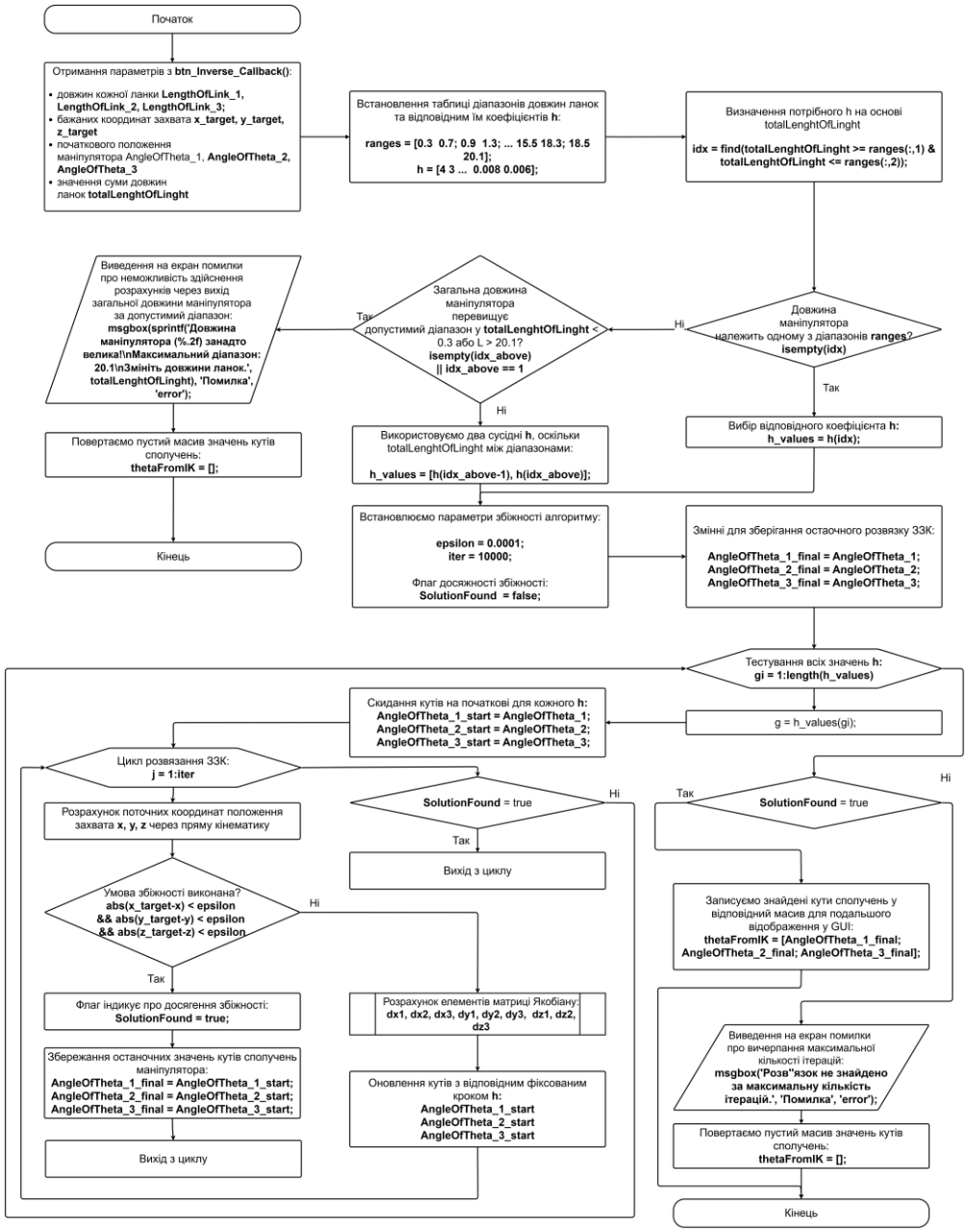


Рис.3.7. Блок-схема роботи підфункції розв'язку ЗЗК ітераційним методом градієнтного спуску IKSolutionWithStepAdaptetion()

На початку виконання з переданих структур отримуються довжини ланок L_1, L_2, L_3 (LengthOfLink_1, LengthOfLink_2, LengthOfLink_3), координати цільової точки x, y, z (x_target, y_target, z_target) та початкові значення кутів $\theta_1, \theta_2, \theta_3$ (AngleOfTheta_1, AngleOfTheta_2, AngleOfTheta_3) зі структури paramsStartPositionThetaInRad.

Встановлюються параметри алгоритму:

- точність збіжності 10^{-4} м (epsilon = 0.0001);
- максимальна кількість ітерацій 10^4 (iter = 10000).

Значення ітераційного коефіцієнта h визначається автоматично на основі сумарної довжини ланок L_{sum} (totalLenghtOfLinght) шляхом пошуку відповідного діапазону (зазначеного у Табл.2.2) у масивах ranges та h, що реалізує адаптивну процедуру підбору, описану у п. 2.3. Якщо значення L_{sum} потрапляє між двома сусідніми діапазонами – формується список із двох потенційних значень h, які послідовно перевіряються. Якщо L_{sum} виходить за межі дослідженого діапазону 0,3-20,1 м – функція завершується з повідомленням про помилку та повертає порожній масив.

На кожній ітерації внутрішнього циклу обчислюються поточні координати робочого органу за рівняннями прямої кінематики (формули (2.14)-(2.16)), після чого перевіряється умова збіжності: якщо похибки за всіма координатами одночасно стають меншими за epsilon – алгоритм зупиняється з флагом збіжності SolutionFound = true. Матриця Якобі формується аналітично через часткові похідні координат за кутами зчленувань (dx1-dz3) за формулами (2.22)-(2.30). Оновлення кутів виконується за правилом градієнтного спуску (2.19)-(2.21) з фіксованим кроком h.

За умови досягнення збіжності функція повертає масив thetaFromIK із знайденими значеннями кутів та виводить у консоль фінальні похибки

позиціонування і кількість виконаних ітерацій. Якщо жодне із значень h не забезпечило збіжності за максимальну кількість ітерацій – масив thetaFromIK повертається порожнім.

Підфункція геометричного (аналітичного) методу GeometricIKSolution.

Реалізує аналітичний замкнений розв'язок зворотної задачі кінематики на основі геометричних співвідношень. Детальна блок-схема алгоритму наведена на Рис. 3.8.

Після отримання L_1, L_2, L_3 та координат цільової точки $x_{des}, y_{des}, z_{des}$ обчислюється кут першого шарніра за формулою (2.41) та зберігається у змінній AngleOfTheta_1. Далі визначається просторова відстань r (змінна r) за формулою (2.34) та косинус кута 3-го шарніра θ_3 (cos_AngleOfTheta_3) за формулою (2.35). Додатково було введено вбудовану перевірку на можливість фізичної досяжності точки: якщо $|\cos\theta_3| > 1$, точка є недосяжною, функція повертає порожній масив. В іншому випадку обчислюються дві конфігурації маніпулятора – «лікоть вгору» (AngleOfTheta_3_up, де $\theta_3^{\uparrow} = -\arccos(\cos\theta_3)$ та «лікоть вниз» (AngleOfTheta_3_down, де $\theta_3^{\downarrow} = \arccos(\cos\theta_3)$ відповідно до формули (2.36). Відповідний кут другого шарніра, що є спільним для обох конфігурацій визначається за формулою (2.39) та записується у змінні AngleOfTheta_2_up, AngleOfTheta_2_down.

Перевірка отриманих конфігурацій на коректність виконується допоміжною вкладеною функцією checkConfig, яка перевіряє відсутність значень NaN та, за наявності заданих користувачем обмежень (ranges_defined), відповідність кутів $\theta_1, \theta_2, \theta_3$ допустимим діапазонам [q1_min…q1_max], [q2_min…q2_max], [q3_min…q3_max]. Пріоритет надається конфігурації «лікоть вгору»; якщо вона не задовольняє обмеженням, користувачу пропонується перейти до конфігурації «лікоть вниз». Допоміжна функція printErrors виводить у консоль фінальні похибки позиціонування шляхом підстановки знайдених кутів у рівняння прямої кінематики.

У разі успішної перевірки функція повертає масив thetaFromGeometricIK з розрахованими значенням кутів сполучень. Якщо жодна з конфігурацій не задовольняє накладеним обмеженням, масив thetaFromGeometricIK повертається порожнім із відповідним повідомленням про помилку.

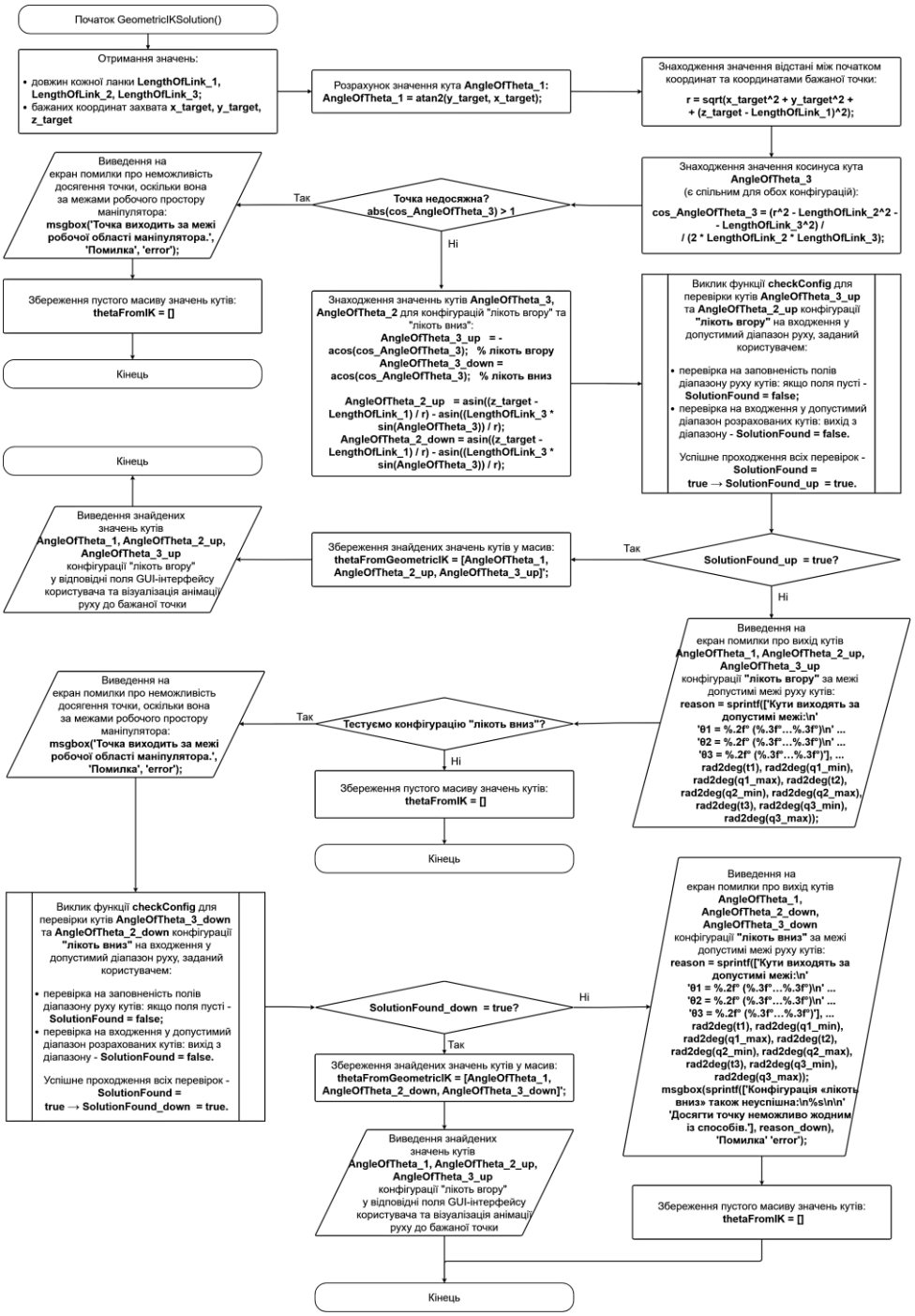


Рис.3.8. Блок-схема роботи підфункції розв’язку ЗЗК геометричним (аналітичним) методом GeometricIKSolution()

3.4. Розрахунок і побудова траєкторії

Розрахунок і побудова траєкторії реалізована у вигляді трьох взаємопов'язаних функцій: підфункції обчислення коефіцієнтів квінтичного полінома trajectory_blend_quintic(), підфункції дискретизації та валідації траєкторії compute_trajectory_with_validation(), а також блоку анімації та візуалізації, інтегрованого безпосередньо у головний оброблювач btn_Inverse_Callback(). Модуль викликається автоматично після успішного розв'язання ЗЗК.

Підфункція trajectory_blend_quintic().

Підфункція trajectory_blend_quintic() реалізує розрахунок коефіцієнтів квінтичного полінома п'ятого ступеня для кожного сегмента траєкторії відповідно до п. 2.5. Детальна блок-схема алгоритму наведена на Рис. 3.9.

Функція приймає на вхід вектори вузлових значень положення x, швидкості x_d та прискорення x_dd, а також загальний час руху t_end. Часові вузли рівномірно розподіляються на інтервалі $[0; t_{end}]$ для j вузлових точок, утворюючи $j - 1$ сегментів. Для кожного сегмента обчислюється його тривалість у змінній durationOfCurrentSegment та формується матриця граничних умов B розміром 6×6, що відповідає системі (2.52) при $t_0 = 0$.

Вектор граничних умов c формується з заданих значень положення, швидкості та прискорення на початку і в кінці кожного сегмента. Коефіцієнти полінома $a_0 \ldots a_5$ обчислюються шляхом розв'язання системи лінійних рівнянь за формулою (2.51), що еквівалентно зворотному діленню $B \backslash \boldsymbol{C}$ у MATLAB. Перші три коефіцієнти визначаються безпосередньо з початкових умов за формулами (2.53)-(2.54), тоді як a_3, a_4, a_5 знаходяться через кінцеві умови за формулами (2.55)-(2.57). Результат зберігається у матриці coef розміром $(j - 1) \times 6$, де кожен рядок містить коефіцієнти для відповідного сегмента.

У головному оброблювачі btn_Inverse_Callback() функція trajectory_blend_quintic() викликається окремо для кожного з трьох сполучень маніпулятора з нульовими граничними умовами за швидкістю та прискоренням на початку і в кінці руху (x_d = [0, 0], x_dd = [0, 0]), що забезпечує плавний старт і зупинку відповідно до вимог п. 2.5.

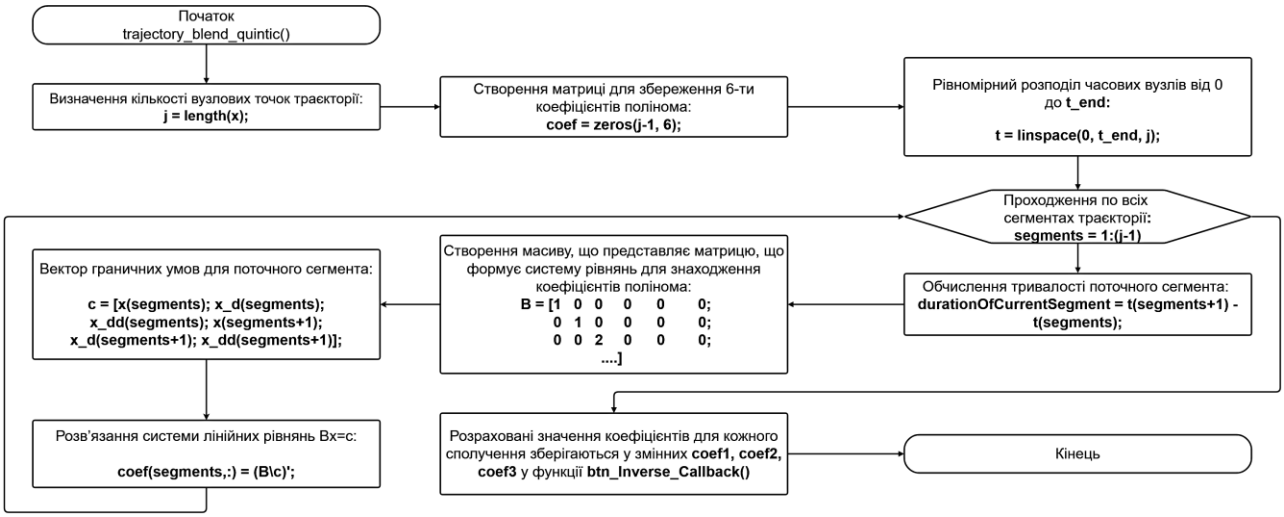


Рис. 3.9. Блок-схема роботи підфункції розрахунку коефіцієнтів поліному 5-го ступеня trajectory_blend_quintic()

Підфункція compute_trajectory_with_validation().

Підфункція compute_trajectory_with_validation() виконує дискретизацію траєкторії та покрокову перевірку допустимості кутів зчленувань. Детальна блок-схема алгоритму наведена на Рис. 3.10.

Часовий вектор формується з кроком dt, що забезпечує рівно 50 дискретних точок при загальному часі анімації tf = 10 с. Перша точка масиву ініціалізується значенням q_start – поточним положенням маніпулятора, взятим з handles.current_position.

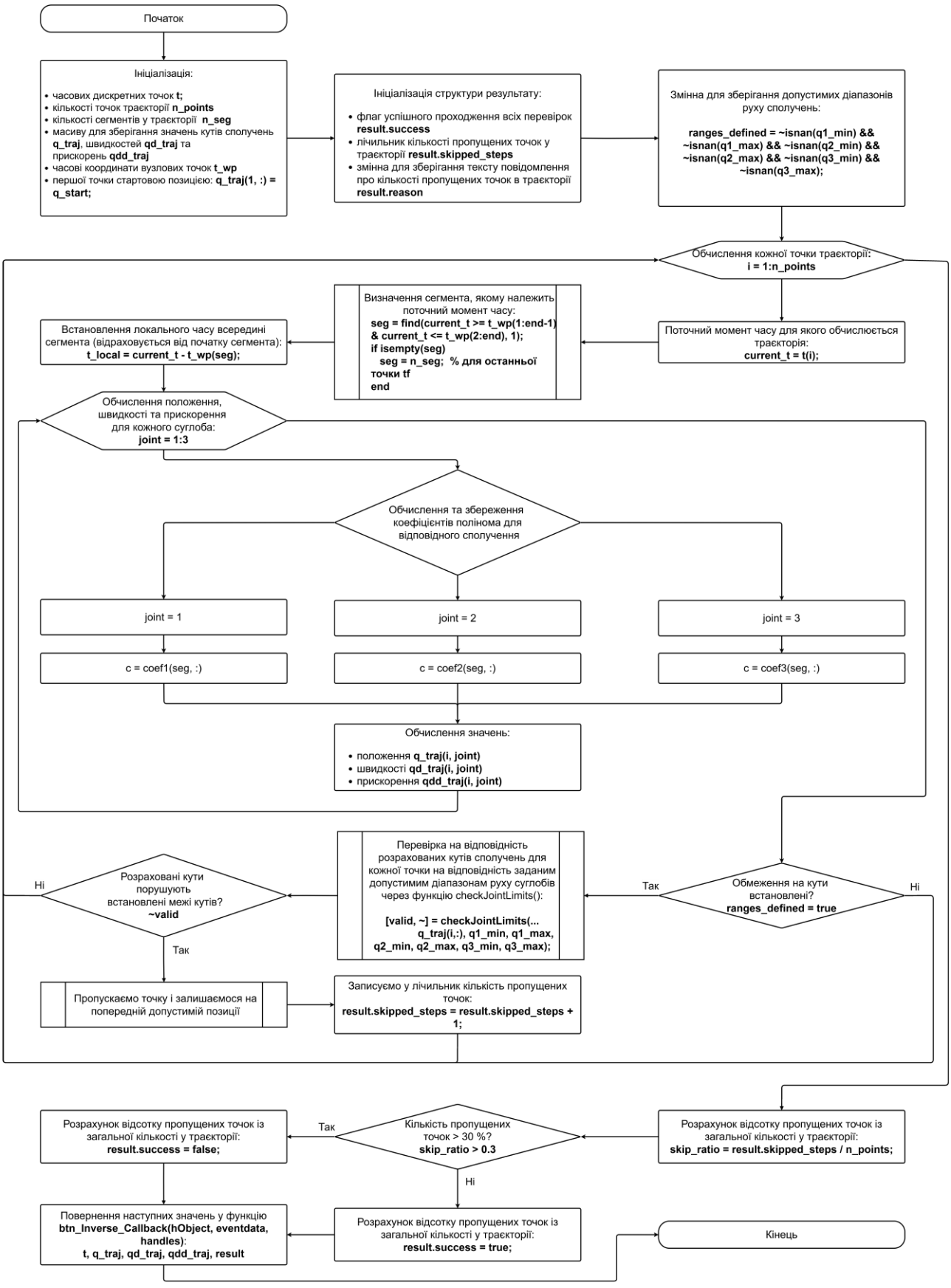


Рис. 3.10. Блок-схема роботи підфункції дискретизації траєкторії compute_trajectory_with_validation()

На кожній ітерації визначається сегмент, якому належить поточний момент часу, та обчислюється локальний час у змінній t_local. Для кожного зчленування обчислюються значення кута, кутової швидкості та кутового прискорення шляхом підстановки t_local у квінтичний поліном та його похідні відповідно до виразів (1.15)-(1.17).

Якщо користувачем задано обмеження діапазонів зчленувань, кожна точка перевіряється допоміжною функцією checkJointLimits() з допуском 2°. У разі порушення меж поточна точка замінюється попередньою допустимою позицією, а кутові швидкість і прискорення обнуляються. Лічильник замінених кроків накопичується у result.skipped_steps. Після завершення циклу обчислюється частка замінених кроків: якщо вона перевищує 30% – стан флагу індикації розв'язку змінюється на result.success = false і в головному оброблювачі користувачу виводиться діалогове попередження. В обох випадках рух не блокується.

3.5. Розробка графічного інтерфейсу користувача

Графічний інтерфейс користувача розроблено засобами MATLAB GUIDE. Зовнішній вигляд застосунку та вікно його розробки GUIDE наведено на Рис. 3.11 та Рис.3.12. Вікно застосунку поділене на ліву панель керування та праву панель візуалізації.

Ліва панель містить поля введення довжин ланок L1, L2, L3 (L_1, L_2, L_3), кутів сполучень Theta_1, Theta_2, Theta_3 ($\theta_1, \theta_2, \theta_3$) з відповідними слайдерами та поля координат захвату Pos_X, Pos_Y, Pos_Z (x, y, z). Кнопка «Зворотна кінематика» запускає оброблювач події btn_Inverse_Callback(), що розв'язує зворотну задачу кінематики за введеними координатами та запускає анімацію і побудову графіків.

Нижче розташована панель «Допустимий діапазон руху суглобів» з полями введення мінімальних та максимальних значень кутів для кожного зчленування Theta_1, Theta_2, Theta_3 (Th1_workspase_min/max, Th2_workspase_min/max, Th3_workspase_min/max). Задані діапазони використовуються при перевірці допустимості результатів ЗЗК та валідації траєкторії у підфункції compute_trajectory_with_validation().

Права панель містить вісь тривимірної анімації graphik_of_animation_robot, де відображається модель маніпулятора з анімацією руху та кольоровим слідом траєкторії захвата, а також три графічні осі для відображення результатів кінематичного аналізу вздовж траєкторії:

- movement_angle_of_each_join – кути $\theta_i(t)$;
- angular_velocity_of_each_join – кутові швидкості $\omega_i(t)$;
- angular_acceleration_of_each_join – кутові прискорення $\alpha_i(t)$.

Усі три графіки будуються одночасно після завершення анімації на основі масивів q_angle, q_d_vel та q_d2_accel, що обчислюються підфункцією compute_trajectory_with_validation().

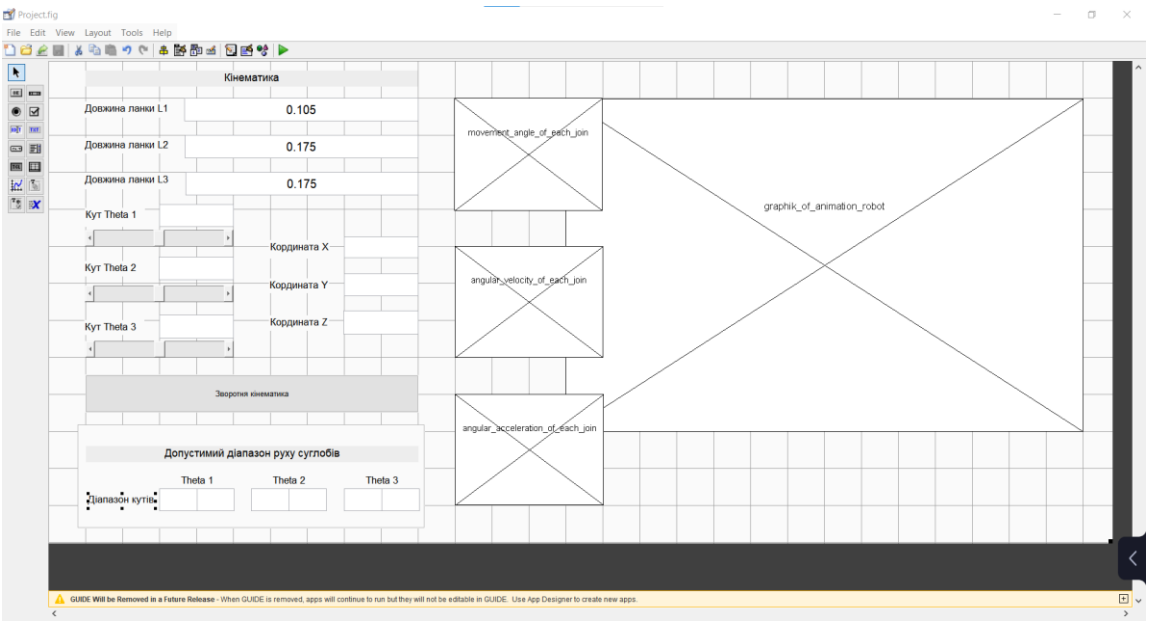


Рис. 3.11. Вікно розробки графічного інтерфейсу користувача GUIDE

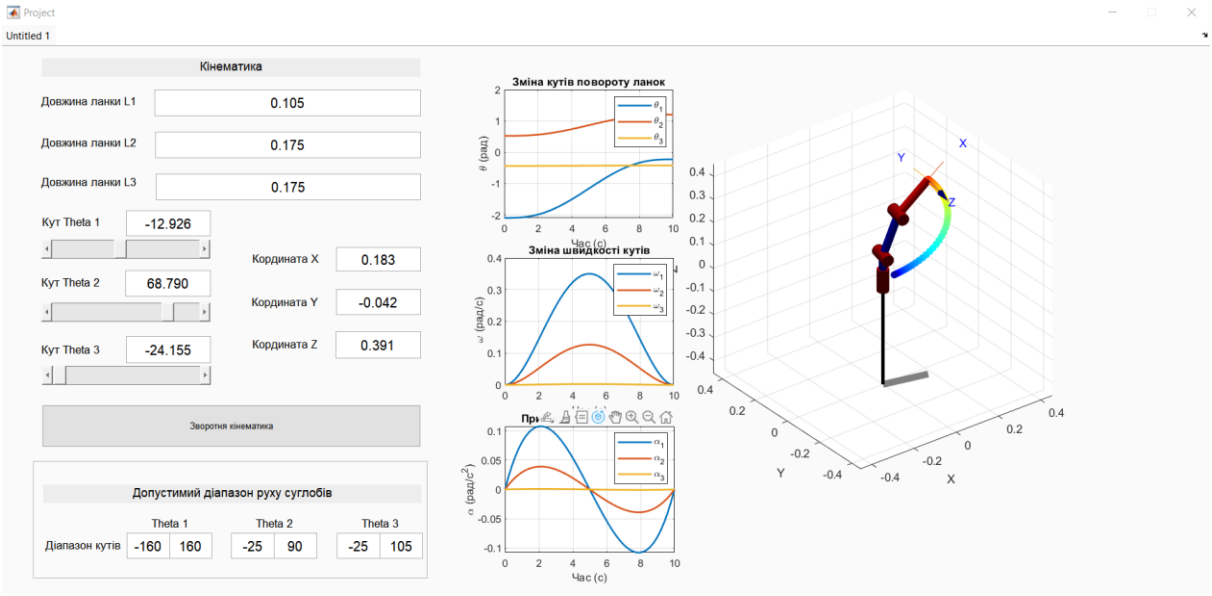


Рис. 3.12. Зовнішній вигляд розробленого інтерфейсу

3.6. Висновки за розділом 3

Розроблений інформаційно-обчислювальний комплекс реалізує повний цикл кінематичного аналізу триланкового маніпулятора антропоморфного типу у єдиному програмному середовищі MATLAB з використанням пакету Robotics Toolbox. Завдяки параметричній побудові кінематичної моделі на основі таблиці ДХ-параметрів комплекс є універсальним і може бути застосований до будь-якого послідовного триланкового маніпулятора – достатньо змінити довжини ланок та діапазони руху сполучень у полях графічного інтерфейсу.

Застосування рівневої архітектури з чітким розмежуванням модулів графічного інтерфейсу, математичних розрахунків та візуалізації забезпечує незалежність компонентів і спрощує подальше розширення системи. Модуль прямої кінематики працює в реальному часі без необхідного додаткового підтвердження з боку користувача; модуль розв’язання зворотної кінематики реалізує гібридну стратегію, де основним є ітераційний метод градієнтного спуску, а геометричний метод виступає резервним із можливістю вибору між

двома конфігураціями. Модуль планування траєкторії на основі полінома 5-го порядку гарантує плавність руху за положенням, швидкістю та прискоренням, а вбудована валідація попереджає про порушення допустимих меж зчленувань.

Графічний інтерфейс інтегрує всі модулі в єдине робоче середовище з тривимірною візуалізацією руху маніпулятора та графіками кінематичних характеристик кожного сполучення, що робить комплекс придатним для інженерного аналізу на етапі проектування робототехнічних систем для їх подальшої коректної фізичної реалізації.

РОЗДІЛ 4

ДОСЛІДЖЕННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ МОДЕЛЮВАННЯ МАНІПУЛЯТОРА

4.1. Проведення модельних досліджень

З метою верифікації розробленого інформаційно-обчислювального комплексу та підтвердження коректності реалізованих алгоритмів здійснено серію модельних досліджень у середовищі MATLAB (пакет Robotics Toolbox). Дослідження охоплює точність розв'язання зворотної задачі кінематики та поведінку алгоритму поблизу сингулярних конфігурацій.

Як зазначено у пп. 2.3, одним із ключових факторів, що визначає швидкість збіжності алгоритму градієнтного спуску для розв'язання зворотної кінематики, є вибір коефіцієнта ітераційного кроку h. У зв'язку з цим окрему увагу було приділено дослідженню впливу даного параметра на роботу алгоритму для маніпуляторів з різною геометрією.

В ході експериментів, проведених у попередніх роботах за цим напрямком [46, 54], емпіричним шляхом здійснено підбір необхідного коефіцієнта h, де в якості основного критерію вибору розміру кроку обрано значення загальної довжини маніпулятора L_{sum}. Для забезпечення репрезентативності результатів в дослідженнях тестування охоплювало конфігурації маніпулятора з варіативною сумарною довжиною ланок у діапазоні від 0,3 до 20,1 м з кроком 0,2 м.

Для кожної конфігурації методом прямої кінематики формувався робочий простір маніпулятора, що забезпечував набір із 48139 цільових точок, координати яких послідовно подавались на вхід алгоритму оберненої кінематики з метою калібрування значення ітераційного коефіцієнта h. При цьому в роботі [54] робочий простір будувався за фіксованими кутовими діапазонами $\theta_1 \in [-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [-90°; 0°]$ та початковим положенням $\theta_1 = 30°, \theta_2 = 30°, \theta_3 = -45°$. У роботі [46] використовувались аналогічні діапазони

для першого та другого суглобів, проте для третього суглоба було прийнято $\theta_3 \in [0°; 90°]$, а початкове положення задавалось як $\theta_1 = 30°, \theta_2 = 30°, \theta_3 = 45°$.

Критерій L_{sum} є фізично обґрунтованим показником масштабування, оскільки елементи матриці Якобі геометрично пропорційні довжинам ланок, і, отже, норма Якобі, яка визначає величину градієнта в оновленні спуску, змінюється відповідно до характерного розміру маніпулятора [55, 56, 57]. Оскільки L_{sum} апроксимує максимальний радіус робочої області, він стає природним проксі-параметром масштабу градієнта: значення кроку h, що не відповідає цьому масштабу, призводить або до перевищення граничної кількості ітерацій, або до надмірно повільної збіжності [54]. Важливим є те, що за результатами [54] алгоритм стабільно працює лише за дотримання умови, що жодна ланка не є домінуючою, тобто забезпечується обмеження відносного внеску окремих дистильних ланок $max\left(\frac{L_2}{L_3}, \frac{L_3}{L_2}\right) \leq 4$, що робить L_{sum} надійним параметром масштабу.

Відповідно, у [46, 54] отримано оптимальні значення статичного ітераційного коефіцієнта h для кожного діапазону L_{sum}; відповідні значення загальної довжини маніпулятора та коефіцієнта h наведено у Табл. 4.1.

Таблиця 4.1 – Значення фіксованого оптимального ітераційного коефіцієнту h відповідно до загальної довжини маніпулятора [54]

Загальна довжина маніпулятора L_{sum}, м	Значення ітераційного коефіцієнту h	Загальна довжина маніпулятора L_{sum}, м	Значення ітераційного коефіцієнту h
0,3-0,7	4	5,9-6,3	0,06
0,9-1,3	3	6,5-7,5	0,05
1,5-2,1	1	7,7-8,9	0,04
2,3-2,7	0,4	9,1-11,7	0,02
2,9-3,1	0,3	11,9-15,3	0,01
3,3-5,3	0,1	15,5-18,3	0,008
5,5-5,7	0,07	18,5-20,1	0,006

В подальшому в даній роботі досліджується саме модифікований алгоритм підбору оптимального фіксованого значення ітераційного коефіцієнту h, відповідно до даних, отриманих емпіричним шляхом у Табл. 4.1.

4.2. Дослідження точності розв'язання задачі зворотної кінематики

Для оцінки точності розроблених методів розв'язання зворотної задачі кінематики – ітераційного алгоритму градієнтного спуску та геометричного методу, принципи роботи яких детально описано у пп. 2.3 та 3.3, – проведено тестування на різних конфігураціях маніпулятора з використанням множини цільових точок у межах робочої області.

Результати тестування точності обох методів зведено до Табл. 4.2, де початковим наближенням (початковою позицією захвату) для ітераціного алгоритму градієнтного спуску взято $\theta_1^0 = -120°, \theta_2^0 = 30°, \theta_3^0 = -25°$. Для геометричного методу не має необхідності встановлення такого значення, згідно формул (2.36)-(2.41).

Таблиця 4.2 – Обчислювальна точність алгоритму розв'язання ЗЗК

№	Довжина ланки, м	Ітераційний коефіцієнт h	Цільова позиція	Кількість ітерацій	Обчислена позиція	Похибка позиціонування, м
Метод градієнтного спуску з адаптивним кроком						
1	L_1: 0,125 L_2: 0,165 L_3: 0,165	4	x_{des}: -0,13200000 y_{des}: 0,13200000 z_{des}: 0,39100000	1424	x: -0,13205331 y: 0,13205331 z: 0,39109979	ε_x: 5,331 $\cdot 10^{-5}$ ε_y: 5,331 $\cdot 10^{-5}$ ε_z: 9,979 $\cdot 10^{-5}$

			x_{des}: 0,08300000 y_{des}: -0,14300000 z_{des}: 0,29000000	221	x: 0,08305595 y: -0,14309639 z: 0,29006885	ε_x: 5,595 $\cdot 10^{-5}$ ε_y: 9,639 $\cdot 10^{-5}$ ε_z: 6,885 $\cdot 10^{-5}$
			x_{des}: 0,21900000 y_{des}: 0,00000000 z_{des}: -0,05900000	187	x: 0,21890817 y: -0,00000000 z: -0,05890163	ε_x: 9,183 $\cdot 10^{-5}$ ε_y: 0 ε_z: 9,837 $\cdot 10^{-5}$
2	L_1: 4,753 L_2: 4,753 L_3: 2,375	0,02	x_{des}: -3,08600000 y_{des}: 3,08600000 z_{des}: 10,26600000	953	x: -3,08594160 y: 3,08594160 z: 10,26590053	ε_x: 5,840 $\cdot 10^{-5}$ ε_y: 5,840 $\cdot 10^{-5}$ ε_z: 9,947 $\cdot 10^{-5}$
			x_{des}: 1,18800000 y_{des}: -2,05700000 z_{des}: 9,50600000	142	x: 1,18802881 y: -2,05704993 z: 9,50609334	ε_x: 2,881 $\cdot 10^{-5}$ ε_y: 4,993 $\cdot 10^{-5}$ ε_z: 9,334 $\cdot 10^{-5}$
			x_{des}: 5,49300000 y_{des}: 0,00000000 z_{des}: 1,69600000	151	x: 5,49309806 y: -0,00000000 z: 1,69593623	ε_x: 9,806 $\cdot 10^{-5}$ ε_y: 0 ε_z: 6,377 $\cdot 10^{-5}$
3	L_1: 4,000 L_2:12,075 L_3: 4,025	0,006	x_{des}: -7,24000000 y_{des}: 7,24000000 z_{des}: 16,18600000	1054	x: -7,23993878 y: 7,23993878 z: 16,18590008	ε_x: 6,122 $\cdot 10^{-5}$ ε_y: 6,122 $\cdot 10^{-5}$ ε_z: 9,992 $\cdot 10^{-5}$
			x_{des}: 2,01300000 y_{des}: -3,48600000 z_{des}: 16,07500000	143	x: 2,01301630 y: -3,48602997 z: 16,07509218	ε_x: 1,630 $\cdot 10^{-5}$ ε_y: 2,997 $\cdot 10^{-5}$ ε_z: 9,218 $\cdot 10^{-5}$
			x_{des}: 13,26800000 y_{des}: 0,00000000	135	x: 13,26809811	ε_x: 9,811 $\cdot 10^{-5}$

			z_{des}: -1,87900000		y: 0,00000000 z: -1,87904807	ε_y: 0 ε_z: 4,807 $\cdot 10^{-5}$
Геометричний метод						
1	L_1: 0,125 L_2: 0,165 L_3: 0,165	-	x_{des}: -0,13200000 y_{des}: 0,13200000 z_{des}: 0,39100000	0	x: -0,13200000 y: 0,13200000 z: 0,39100000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 0,08300000 y_{des}: -0,14300000 z_{des}: 0,29000000	0	x: 0,08300000 y: -0,14300000 z: 0,29000000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 0,21900000 y_{des}: 0,00000000 z_{des}: -0,05900000	0	x: 0,21900000 y: 0,00000000 z: -0,05900000	ε_x: 0 ε_y: 0 ε_z: 0
2	L_1: 4,753 L_2: 4,753 L_3: 2,375	-	x_{des}: -3,08600000 y_{des}: 3,08600000 z_{des}: 10,26600000	0	x: -3,08600000 y: 3,08600000 z: 10,26600000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 1,18800000 y_{des}: -2,05700000 z_{des}: 9,50600000	0	x: 1,18800000 y: -2,05700000 z: 9,50600000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 5,49300000 y_{des}: 0,00000000 z_{des}: 1,69600000	0	x: 5,49300000 y: 0,00000000 z: 1,69600000	ε_x: 0 ε_y: 0 ε_z: 0
3	L_1: 4,000 L_2:12,075 L_3: 4,025	-	x_{des}: -7,24000000 y_{des}: 7,24000000 z_{des}: 16,18600000	0	x: -7,24000000 y: 7,24000000 z: 16,18600000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 2,01300000 y_{des}: -3,48600000 z_{des}: 16,07500000	0	x: 2,01300000 y: -3,48600000 z: 16,07500000	ε_x: 0 ε_y: 0 ε_z: 0
			x_{des}: 13,26800000 y_{des}: 0,00000000 z_{des}: -1,87900000	0	x: 13,26800000 y: 0,00000000 z: -1,87900000	ε_x: 0 ε_y: 0 ε_z: 0

За результатами тестування, зведеними у Табл. 4.2, можна зробити наступні висновки щодо точності та обчислювальних характеристик реалізованих методів розв'язання зворотної задачі кінематики.

Ітераційний метод градієнтного спуску забезпечує позиційну похибку, що не перевищує встановленого порогу збіжності $\varepsilon = 10^{-4}$ м в усіх протестованих точках для всіх трьох конфігурацій маніпулятора з різними значеннями загальних довжин ланок. Складова похибки вздовж кожної з осей не перевищує 10^{-4} м, що підтверджує дотриманні правила умови зупинки алгоритму (2.31). При цьому кількість ітерацій суттєво залежить від положення цільової точки у робочому просторі. Для точок у внутрішній частині простору, де маніпулятор має задовільну маніпулятивність, алгоритм сходиться за 135-221 ітерацій. Натомість для точок поблизу меж робочого простору – кількість ітерацій зростає до 953-1424 ітерацій, що свідчить про уповільнення сходження поблизу сингулярних та граничних зон. Важливо зазначити, що незважаючи на значно більшу кількість ітерацій у таких точках, необхідна точність $\varepsilon = 10^{-4}$ м все одно досягається, що підтверджує стійкість реалізованого механізму підібраного кроку.

Геометричний (аналітичний) метод демонструє абсолютну точність позиціонування в усіх протестованих точках: похибка $\varepsilon_x = \varepsilon_y = \varepsilon_z = 0$ досягається в межах машинної точності обчислень. Метод не потребує ітераційних обчислень, що забезпечує миттєве отримання результату незалежно від положення цільової точки у робочому просторі. Це пояснюється замкненою аналітичною формою розв'язку (2.36)-(2.41), яка при підстановці у пряму кінематику відтворює задані координати з точністю до похибки округлення числових операцій.

4.3. Дослідження впливу сингулярності на збіжність алгоритму та його стійкості до зміни форми робочого простору

У попередніх дослідженнях [46, 54] розглядалися різні властивості алгоритму для різних геометрій робочого простору: у [46] – відсоток досяжних точок за різного розподілу довжин ланок та загальних значень L_{sum} та його

поведінка біля сингулярних зон для $\theta_3 \in [0°; 90°]$ за фіксованого початкового положення захвата; у [54] – його стійкість до зміни конфігурації маніпулятора в межах одного значення загальної довжини $L_{sum} = 0{,}3$ м та розподіл кількості ітерацій для $\theta_3 \in [-90°, 0°]$ за різних початкових положень захвату. З метою узагальнення цих результатів у даній роботі виконано порівняльний аналіз для обох варіантів геометрії робочого простору з різними значеннями сумарної довжини ланок.

Оскільки згідно з [54] ефективність алгоритму є нечутливою до вибору початкової конфігурації, тестування проводиться для двох початкових положень, що не використовувались у попередніх роботах: $\theta_1^0 = -120°, \theta_2^0 = 30°, \theta_3^0 = -25°$ для робочої області $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [-90°, 0°]$ та $\theta_1^0 = -120°, \theta_2^0 = 30°, \theta_3^0 = 25°$ для діапазону $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [0°; 90°]$, що дозволяє коректно порівняти отримані результати з даними робіт [46, 54].

Аналіз даних Табл. 4.3 та Табл. 4.4 дозволяє виявити закономірності збіжності алгоритму залежно від геометричного положення цільової точки у робочій області маніпулятора.

При правильно підібраному значенні h відповідно до Табл. 2.2 частка недосяжних точок у межах ліміту 10^4 ітерацій не перевищує 12 % для обох геометрій робочого простору. Водночас використання неоптимального значення h призводить до суттєвого погіршення результату: наприклад, для L_{sum} = 11,881 м при $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [0°; 90°]$ завищене значення h = 0,04 дає 49,17 % недосяжних точок, а занижене h = 0,008 – 18,34 %, що значно перевищує результат при оптимальному h = 0,02. Показово, що для геометрії робочого простору у $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [-90°, 0°]$ занижене значення ітераційного коефіцієнта також в деяких конфігураціях може не перевищувати 12 %, проте, оптимальне значення h все одно показує в 2 рази кращі показники,

аніж його занижене значення. Детально причина такої поведінки буде розглянута пізніше.

Все вище зазначене підтверджує, що правильний підбір ітераційного коефіцієнта є визначальним чинником ефективності алгоритму: завищене h спричиняє «перестрибування» мінімуму функції похибки (2.17) з подальшими осциляціями навколо розв'язку або розбіжністю алгоритму, тоді як занижене – надмірно сповільнює оновлення кутів зчленувань (2.19)-(2.21) [46].

Таблиця 4.3 – Результати моделювання при різних значеннях загальної довжини маніпулятора для робочого простору $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [0°; 90°]$ та $\theta_1^0 = -120°, \theta_2^0 = 30°, \theta_3^0 = 25°$

№	Ітераційний коефіцієнт h	Розподіл загальної довжини по окремих ланках L_{sum}, м	Кількість недосяжних точок	Кількість недосяжних точок, %
Загальна довжина ланок L_{sum}= 0,455 м				
1	7	L_1: 0,125 L_2: 0,165 L_3: 0,165	10170	21,13
2	**4**		**3777**	**7,85**
3	1		11070	23,00
Загальна довжина маніпулятора L_{sum}= 11,881 м				
4	0,04	L_1: 4,753 L_2: 4,753 L_3: 2,375	23671	49,17
5	**0,02**		**5750**	**11,94**
6	0,008		8829	18,34
Загальна довжина маніпулятора L_{sum}= 20,100 м				
7	0,008	L_1: 4,000 L_2: 12,075 L_3: 4,025	21709	45,10
8	**0,006**		**5599**	**11,63**
9	0,004		12653	26,28

Аналіз структури розподілу кількості ітерацій у робочій області маніпулятора, наведений на Рис. 4.1 та Рис. 4.2, дозволяє зробити висновок, про

належність зазначених 12 % недосяжних точок за оптимального значення ітераційного коефіцієнту сингулярним положенням робочого простору. Тобто присутня закономірність збіжності алгоритму залежно від геометричного положення цільової точки. На графіках застосовано кольорове кодування за кількістю ітерацій: зелений – швидка збіжність (до 10^4), рожевий – помірна (10^4-10^5), жовтий – повільна (10^5-10^6), червоний – критична (10^6-10^7) [46].

Таблиця 4.4 – Результати моделювання при різних значеннях загальної довжини маніпулятора для робочого простору $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [-90°; 0°]$ та $\theta_1^0 = -120°, \theta_2^0 = 30°, \theta_3^0 = -25°$

№	Ітераційний коефіцієнт h	Розподіл загальної довжини по окремих ланках L_{sum}, м	Кількість недосяжних точок	Кількість недосяжних точок, %
Загальна довжина ланок L_{sum}= 0,455 м				
1	7	L_1: 0,125 L_2: 0,165 L_3: 0,165	8370	17,39
2	**4**		**1906**	**3,96**
3	1		6332	13,15
Загальна довжина маніпулятора L_{sum}= 11,881 м				
4	0,04	L_1: 4,753 L_2: 4,753 L_3: 2,375	20930	43,48
5	**0,02**		**2403**	**4,99**
6	0,008		4777	9,92
Загальна довжина маніпулятора L_{sum}= 20,100 м				
7	0,008	L_1: 4,000 L_2: 12,075 L_3: 4,025	20928	43,47
8	**0,006**		**2093**	**4,35**
9	0,004		4069	8,45

Точки помірної, повільної та критичної збіжності зосереджені у двох характерних зонах: вздовж зовнішньої межі робочого простору та в околі осі z поблизу $x = 0, y = 0$. Перша зона відповідає ліктьовій сингулярності ($\theta_3 = 0°$ або $\theta_3 = \pm 90°$), друга – сингулярності плеча (координати положення

захвата (2.14)-(2.15) стають нульовими). Фундаментальна причина уповільнення збіжності у цих зонах полягає у виродженні матриці Якобі: детермінант J (2.42) прямує до нуля, внаслідок чого часткові похідні $\frac{\partial x}{\partial \theta_i}, \frac{\partial y}{\partial \theta_i}, \frac{\partial z}{\partial \theta_i}$ набувають малих значень, крок оновлення кутів за формулами (2.19)-(2.21) різко зменшується, і для досягнення заданої точності 10^{-4} м потрібна значно більша кількість ітерацій [46]. Як наслідок такі точки хибно класифікуються як недосяжні.

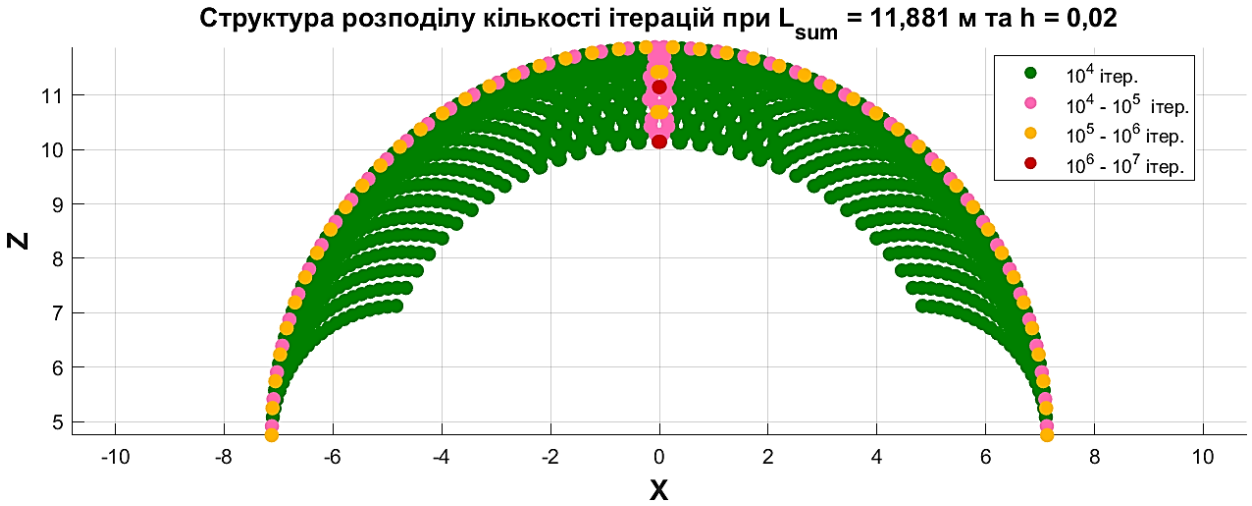


Рис. 4.1. Графік залежності розподілу кількості ітерацій алгоритму від положення точок у робочій області $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [0°; 90°]$ при L_{sum}= 11,881 м

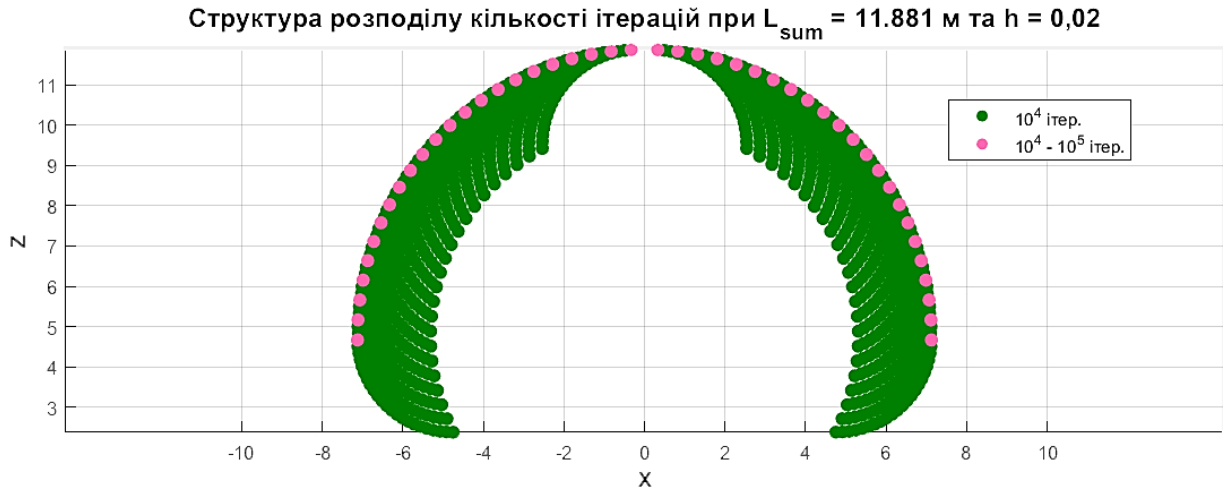


Рис. 4.2. Графік залежності розподілу кількості ітерацій алгоритму від положення точок у робочій області $[-180°; 180°], \theta_2 \in [0°; 90°], \theta_3 \in [-90°; 0°]$ при L_{sum}= 11,881 м

Як зазначалося вище, робочий простір з $\theta_3 \in [-90°; 0°]$ дає значно кращі результати досяжності точок робочого простору в порівнянні з $\theta_3 \in [0°; 90°]$. Причиною такої поведінки алгоритму є форма робочого простору, що визначає загальну кількість і розташування сингулярних точок. При $\theta_3 \in [0°; 90°]$ ланка L_3 відхилена вгору, що формує робочий простір з більшою концентрацією точок поблизу ліктьової сингулярності та сингулярності плеча. При $\theta_3 \in [-90°; 0°]$ отримана форма робочого простору є більш рівномірною, оскільки L_3 відхиляється «донизу» відносно L_2, що формує робочу область з більшою часткою точок у нижній та середній зонах простору, де маніпулятор має кращу маніпулятивність і уникає сингулярності плеча. Внаслідок цього частка недосяжних точок становить 3,96-4,99 % проти 7,85-11,94 % – тобто у 2-3 рази менше при тих самих L_{sum} та h.

4.4. Висновки за розділом 4

Здійснені тестування і валідація запропонованого алгоритму градієнтного спуску з адаптивним підбором ітераційного коефіцієнта підтверджують його ефективність. Аналіз результатів моделювання, наведених у Табл. 4.1, Табл. 4.3 та Табл. 4.4, показав, що при правильно обраному значенні h відповідно до значень у Табл. 2.2 частка недосяжних точок у межах ліміту 10^4 ітерацій не перевищує 12 % для обох досліджених геометрій робочого простору. Найбільші труднощі збіжності спостерігалися поблизу сингулярних конфігурацій – вздовж зовнішньої межі робочого простору (ліктьова сингулярність, $\theta_3 = 0°$) та в околі вертикальної осі першого зчленування (сингулярність плеча, $x = 0,\ y = 0$), що обумовлено виродженням матриці Якобі у цих зонах.

Встановлено, що ефективність алгоритму суттєво залежить від форми робочого простору, яка визначається допустимими діапазонами кутів зчленувань. Для діапазону $\theta_3 \in [-90°; 0°]$ частка недосяжних точок при

оптимальному h є систематично меншою (3,96-4,99 %), ніж для $\theta_3 \in [0°; 90°]$ (7,85-11,94%), що пояснюється меншою концентрацією точок поблизу сингулярних зон у першому випадку. Водночас використання неоптимального значення h призводить до різкого погіршення результату – частка недосяжних точок може зростати до 43-49%, що у 4-10 разів перевищує показник при оптимальному коефіцієнті.

Загалом результати моделювання засвідчили, що запропонована процедура адаптивного підбору h на основі сумарної довжини ланок L_{sum} забезпечує стабільну збіжність алгоритму в широкому діапазоні геометричних конфігурацій маніпулятора.

Подальша робота в цьому напрямку спрямована на розроблення стратегій раннього детектування та обходу сингулярних зон при плануванні траєкторії та розширення досліджуваного діапазону L_{sum} за межі 0,3-20,1 м.

РОЗДІЛ 5

ОХОРОНА ПРАЦІ

5.1. Загальні положення і поняття охорони праці

Охорона праці являє собою комплексну систему, що охоплює правові, соціально-економічні, організаційно-технічні, санітарно-гігієнічні та лікувально-профілактичні заходи і засоби, спрямовані на збереження життя, здоров'я і працездатності людини безпосередньо у процесі виконання нею трудових обов'язків [59]. Наведене визначення закріплено у статті 1 Закону України «Про охорону праці» № 2694-XII від 14 жовтня 1992 року та є базовим для всієї системи нормативно-правового регулювання у цій сфері.

Законодавча база з охорони праці в Україні є багаторівневою. Її основу формують:

- Конституція України, яка гарантує кожному громадянину право на безпечні та здорові умови праці;
- Закон України «Про охорону праці» № 2694-XII;
- Кодекс законів про працю України;
- Закон України «Про загальнообов'язкове державне соціальне страхування»;
- підзаконні нормативно-правові акти – правила, норми, інструкції та стандарти, затверджені відповідними центральними органами виконавчої влади.

Питання безпечної експлуатації комп'ютерної техніки та екранних пристроїв, що безпосередньо стосується умов виконання даної кваліфікаційної роботи, регулюються наказом Міністерства соціальної політики України від 14.02.2018 № 207 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Ключовими суб’єктами правовідносин у сфері охорони праці виступають роботодавець і працівник. Роботодавець – це власник підприємства, установи або організації чи уповноважений ним орган незалежно від форми власності та виду господарської діяльності, а також фізична особа, що використовує найману працю [60]. Працівник – особа, яка виконує трудові обов’язки на підставі укладеного трудового договору [60]. Обов’язки та права обох сторін чітко розмежовані статтями 13 та 14 Закону України «Про охорону праці».

На роботодавця покладається комплекс організаційних та технічних обов’язків [60]: створення безпечних умов праці на кожному робочому місці, функціонування системи управління охороною праці, організація медичних оглядів працівників відповідно до наказу МОЗ України № 1393, проведення навчання та перевірки знань з охорони праці згідно з Типовим положенням, затвердженим наказом Держнаглядохоронпраці від 26.01.2005 № 15, а також фінансування профілактичних заходів у розмірі не менше 0,5 відсотка від фонду оплати праці за попередній рік відповідно до статті 19 зазначеного Закону. Роботодавець несе пряму відповідальність за порушення цих вимог [60].

Працівник, у свою чергу, зобов’язаний дотримуватися вимог нормативно-правових актів з охорони праці, правил поводження з обладнанням та засобами виробництва, використовувати засоби індивідуального та колективного захисту, а також проходити у встановлені строки медичні огляди [60]. Невиконання цих вимог є порушенням трудової дисципліни і тягне за собою відповідальність відповідно до Кодексу України про адміністративні правопорушення.

Державний нагляд за дотриманням законодавства з охорони праці здійснюється центральним органом виконавчої влади, що реалізує державну політику у цій сфері, а громадський контроль – професійними спілками та уповноваженими найманими працівниками особами відповідно до статей 38-42 Закону України «Про охорону праці» [61]. За порушення законодавства про охорону праці передбачена дисциплінарна, адміністративна, матеріальна та кримінальна відповідальність згідно зі статтею 44 зазначеного Закону, а

максимальний розмір штрафу для юридичних осіб не може перевищувати п’яти відсотків середньомісячного фонду заробітної плати за попередній рік [60].

Таким чином, охорона праці є не лише правовою вимогою, а й необхідною умовою ефективної та безпечної трудової діяльності, що реалізується через узгоджену взаємодію держави, роботодавця та працівника в межах чинного законодавства України.

5.2. Аналіз небезпечних та шкідливих факторів, що діють на розробника інформаційно-обчислювального комплексу для моделювання руху маніпулятора

Тривала робота за комп’ютером супроводжується впливом цілого ряду небезпечних і шкідливих виробничих факторів, систематизація яких є необхідною умовою для розроблення ефективних заходів із забезпечення зменшення їх впливу та забезпечення безпеки праці розробника. Відповідно до ГОСТ 12.0.003-74 «Небезпечні та шкідливі виробничі фактори. Класифікація», усі фактори поділяються на чотири групи: фізичні, хімічні, біологічні та психофізіологічні [62].

Фізичні фактори.

Електричні та електромагнітні небезпеки: можливість ураження електричним струмом при несправностях обладнання або порушенні правил експлуатації; наявність статичної електрики на поверхнях обладнання; вплив електромагнітних полів, що створюються комп’ютерною технікою та периферійними пристроями [63].

Мікроклімат робочого середовища: відхилення температури повітря від оптимальних значень; знижена або підвищена відносна вологість; недостатня або надмірна швидкість руху повітря; підвищена запиленість повітря робочої зони.

Освітлення робочого місця: недостатній рівень освітленості, що призводить до зорового напруження; надмірна яскравість джерел світла; наявність прямих і відбитих відблисків на екрані монітора; нерівномірний розподіл освітлення в полі зору; пульсація світлового потоку штучного освітлення [63, 65].

Шумові впливи: шум від роботи систем охолодження комп'ютера, блоків живлення та периферійних пристроїв; тривалий вплив навіть помірного шуму, що може спричиняти втому [63].

Психофізіологічні фактори.

Нервово-емоційні навантаження: висока концентрація уваги під час розробки алгоритмів моделювання; інтелектуальне навантаження при аналізі математичних моделей та результатів обчислень; відповідальність за коректність програмної реалізації [63].

Зорове та розумове напруження: тривала робота з екраном дисплея; необхідність обробки значного обсягу інформації за короткий час; підвищене навантаження на зоровий аналізатор [63].

Статичні фізичні навантаження: тривале перебування у сидячому положенні; обмежена рухливість під час роботи; можливе перевантаження опорно-рухового апарату [63].

Організаційні фактори: монотонність виконуваних операцій; нераціональне планування робочого часу; недоліки в організації робочого місця [63].

Хімічні фактори.

Можливе накопичення в повітрі робочої зони продуктів, що виділяються електронною технікою (озон, формальдегід тощо); підвищений вміст вуглекислого газу при недостатній вентиляції приміщення [64].

Біологічні фактори.

Наявність мікроорганізмів у повітрі робочого приміщення при недостатній вентиляції та прибиранні [64].

5.3. Заходи зменшення впливу шкідливих та небезпечних факторів, що діють на розробника інформаційно-обчислювального комплексу для моделювання руху маніпулятора

Своєчасне виявлення шкідливих та небезпечних виробничих факторів є необхідною передумовою для розроблення та впровадження ефективних заходів щодо їх усунення або мінімізації. Це, у свою чергу, забезпечує стабільність роботи розробників, сприяє підвищенню продуктивності праці та позитивно впливає на якість програмного продукту, зменшуючи ймовірність виникнення помилок, зумовлених дією несприятливих умов праці.

З урахуванням зазначеного нижче наведено комплекс організаційних, технічних та санітарно-гігієнічних заходів безпеки, спрямованих на зниження впливу шкідливих і небезпечних факторів під час роботи з комп’ютерною технікою.

Організаційні заходи.

Режим праці та відпочинку. Тривалість безперервної роботи за комп’ютером не повинна перевищувати 4 годин. Загальний робочий час за ПК рекомендується обмежувати 6 годинами на добу. Щогодини необхідно робити перерву тривалістю не менше 10 хвилин, під час якої доцільно виконувати гімнастику для очей, кистей рук і хребта. Кожні 20-25 хвилин слід переводити погляд на віддалені предмети, а кожні 2-3 години – влаштовувати повноцінну паузу тривалістю 15-20 хвилин із ходьбою та розминкою всього тіла [65].

Ергономіка робочого місця. Правильна організація робочого місця є одним із найдієвіших інструментів профілактики професійних захворювань. Перед початком роботи необхідно:

- перевірити усунення зауважень щодо стану робочого місця, виявлених у попередній день [66];
- упевнитися у відсутності сторонніх предметів на робочій поверхні [66];
- відрегулювати висоту крісла так, щоб лінія погляду припадала дещо нижче верхнього краю монітора; спинка має підтримувати поперек, а стопи – вільно стояти на підлозі або підставці [66];
- розмістити монітор на відстані 600-700 мм від очей (не ближче 500 мм), розташувавши його під прямим кутом до джерела природного освітлення – переважно з лівого боку; нижній край екрана має бути дещо ближчим до користувача [66];
- клавіатуру розташовувати на відстані 100-300 мм від краю столу; лікті зігнути під прямим кутом і спирати на підлокітники, зап'ястя тримати горизонтально [66];
- маніпулятор типу «миша» розміщувати на спеціальному килимку на одному рівні з клавіатурою [66].

Нормативні вимоги до меблів: висота робочої поверхні столу – 725 мм, ширина – 800-1400 мм, глибина – 800-1000 мм; простір для ніг – заввишки не менше 600 мм, завширшки – не менше 500 мм. Відстань між робочими місцями – не менше 2,0 м, між бічними поверхнями моніторів – не менше 1,2 м. Площа одного робочого місця з плоским дисплеєм – не менше 4,5 м^2 [63, 67].

Мікроклімат і освітлення. Рівень загального освітлення має бути достатнім, але без різкого контрасту з яскравістю екрана [68]. Монітор не слід розташовувати навпроти вікна – це спричиняє відблиски; оптимально – під прямим кутом до нього [68]. На вікнах мають бути жалюзі або штори [68]. За необхідності роботи з паперовими документами додатково використовується настільна лампа потужністю від 60 Вт [68]. У приміщенні слід щодня проводити

вологе прибирання та систематично провітрювати його – не рідше ніж раз на годину.

Профілактика професійних захворювань. Тривала статична поза та обмежена рухова активність є основними факторами ризику для ІТ-фахівців. Найпоширеніші наслідки – комп'ютерний зоровий синдром, синдром зап'ясткового каналу, сколіоз і остеохондроз [69]. Для їх запобігання рекомендується:

- щогодини змінювати положення тіла, не допускаючи тривалої напруги в одній позі;
- виконувати вправи для очей (часте кліпання, короткочасне заплющування, фокусування на далеких предметах), для кистей (згинання-розгинання, обертання) та для хребта (нахили, повороти, потягування);
- за появи болю в спині або руках – зробити легкий самомасаж шиї та плечового поясу, умитися прохолодною водою.

Технічні заходи.

Електробезпека. До початку роботи робітник зобов'язаний провести візуальний огляд обладнання на якому працює та переконатися у наступному [70]:

- цілісності ізоляції всіх кабелів живлення, а також шнурів і штепсельних з'єднань;
- справності електричних розеток і вилок;
- наявності захисного заземлення у системному блоці та моніторі;
- перевірити правильність підключення всіх компонентів до електромережі;
- здійснювати підключення та відключення периферійних пристроїв (принтера, сканера та інших приладів) виключно при вимкненому живленні ПК.

Забороняється: застосовувати саморобні або невідповідні нормам подовжувачі чи перехідники; перевантажувати мережеві фільтри та розгалужувачі; використовувати нестандартне обігрівальне обладнання або лампи розжарювання для обігріву приміщення; підвішувати світильники безпосередньо за струмопровідні дроти, обгортати їх папером, тканиною або іншими горючими матеріалами, експлуатувати без захисних ковпаків; використовувати апаратуру в умовах, що суперечать рекомендаціям виробника [71].

Вимоги до технічного стану обладнання. Забороняється розпочинати роботу за наявності таких несправностей: виражене тремтіння або нестабільність зображення на екрані [69]; пошкоджені кабелі, роз'єми або штепсельні з'єднання; відсутність або несправність захисного заземлення; будь-які інші видимі дефекти обладнання; у разі виявлення несправності роботу необхідно припинити (або не розпочинати) та негайно повідомити відповідального керівника [72].

Санітарно-гігієнічні технічні вимоги. Поверхню екрана монітора слід очищати антистатичною серветкою. Застосування рідких або аерозольних засобів для очищення поверхонь комп'ютерної техніки – заборонено. Обладнання з підвищеним рівнем шуму (сервери, принтери, сканери) необхідно розміщувати за межами безпосередніх робочих зон персоналу. Пульсація яскравості підсвічування моніторів є задокументованим шкідливим фактором, тому слід обирати обладнання з мінімальним рівнем мерехтіння та за можливості регулювати яскравість екрана відповідно до умов освітленості приміщення [69].

5.4. Висновки за розділом 5

У розділі здійснено комплексний аналіз умов праці розробника інформаційно-обчислювального комплексу для моделювання руху маніпулятора з позиції чинного законодавства України про охорону праці.

Встановлено, що нормативно-правову основу безпеки праці на даному робочому місці формують Закон України «Про охорону праці» № 2694-XII, Кодекс законів про працю, ГОСТ 12.0.003-74 та наказ Міністерства соціальної політики України № 207 від 14.02.2018, що регулює вимоги до роботи з екранними пристроями.

За результатами аналізу виробничого середовища визначено основні групи шкідливих і небезпечних факторів, що впливають на розробника: фізичні (електромагнітне випромінювання, статична електрика, недостатнє або нерівномірне освітлення, шум від охолоджувальних систем, несприятливий мікроклімат), психофізіологічні (зорове та інтелектуальне навантаження, статична напруга опорно-рухового апарату, нервово-емоційний стрес), хімічні (можливе виділення шкідливих речовин електронною технікою) та біологічні (мікроорганізми при недостатній вентиляції приміщення).

З метою усунення або зниження впливу виявлених ризиків розроблено та обґрунтовано комплекс заходів трьох рівнів. Організаційні заходи передбачають регламентацію режиму праці та відпочинку з обмеженням тривалості безперервної роботи за ПК до 2 годин і обов’язковими перервами кожні 60 хвилин, а також коректне ергономічне налаштування робочого місця відповідно до нормативних вимог. Технічні заходи охоплюють дотримання правил електробезпеки, перевірку стану заземлення та цілісності ізоляції, своєчасне виявлення і усунення несправностей обладнання. Санітарно-гігієнічні заходи включають підтримання нормованих параметрів мікроклімату, регулярне провітрювання та вологе прибирання, а також забезпечення достатнього і рівномірного освітлення робочої зони.

Виконання сукупності наведених рекомендацій відповідає вимогам чинної нормативної бази та дозволяє суттєво знизити ймовірність виробничого травматизму, розвитку професійних захворювань і зменшити вплив несприятливих факторів, що загалом сприяє підвищенню ефективності та якості розробки програмного продукту.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено та експериментально досліджено роботу інформаційно-обчислювального комплексу для моделювання руху триланкового робота-маніпулятора.

Проведено аналіз існуючих конструктивних типів роботів-маніпуляторів та методів розв'язання задач кінематичного аналізу. За результатами аналізу обґрунтовано вибір послідовної триланкової конфігурації як базової платформи для дослідження. Для розв'язання зворотної задачі кінематики вибрано метод градієнтного спуску з використанням транспонованої матриці Якобі завдяки простоті його реалізації та придатності для роботи в системах реального часу.

Розроблено математичну модель триланкового маніпулятора на основі параметрів Денавіта-Хартенберга. Розв'язано пряму задачу кінематики та отримано аналітичні вирази для визначення координат вектора положення захвата маніпулятора. Проведено аналіз сингулярних конфігурацій через детермінант матриці Якобі. Виявлено два типи сингулярностей – ліктьову та сингулярність плеча – і визначено умови їх виникнення. Для планування траєкторії вибрано поліном п'ятого ступеня, що забезпечує безперервність положення, швидкості та прискорення на всьому інтервалі руху.

Для розв'язання зворотної задачі кінематики реалізовано два методи. Основним є ітераційний алгоритм градієнтного спуску з адаптивним підбором фіксованого ітераційного коефіцієнта h на основі сумарної довжини ланок L_{sum}. Резервним є геометричний (аналітичний) метод, що забезпечує користувачеві вибір між двома можливими способами досягнення бажаної точки - «лікоть вгору» та «лікоть вниз».

Розроблено програмний застосунок з графічним інтерфейсом користувача у середовищі MATLAB GUIDE за допомогою програмного пакету Robotics Toolbox, що реалізує повний цикл кінематичного аналізу маніпулятора. Комплекс є параметричним – підтримує довільні значення довжин ланок та

діапазонів кутів зчленувань, що забезпечує його універсальність для широкого класу триланкових маніпуляторів послідовного типу.

Проведено валідацію алгоритму на наборі з 48139 точок робочого простору для діапазону сумарних довжин ланок 0,3-20,1 м. Встановлено, що при оптимальному значенні h частка недосяжних точок у межах 10^4 ітерацій не перевищує 12 %, тоді як при неоптимальному – зростає до 43-49 %, тобто у 4-10 разів. Виявлено, що найбільші труднощі збіжності виникають поблизу сингулярних конфігурацій, а ефективність алгоритму суттєво залежить від форми робочого простору, що визначається допустимими діапазонами кутів зчленувань.

Практична цінність розробленого комплексу полягає у можливості його застосування як інструменту для дослідження та верифікації алгоритмів кінематичного аналізу на ранніх етапах проектування триланкових маніпуляційних систем – до їх фізичної реалізації. Це дозволяє оцінити коректність конструкції, оптимізувати геометричні параметри та перевірити алгоритми планування траєкторії без необхідності створення фізичного прототипу, що суттєво скорочує час і витрати на розробку робототехнічних систем.

Перспективними напрямками подальших досліджень є розроблення стратегій раннього виявлення та обходу сингулярних зон при плануванні траєкторії, розширення дослідженого діапазону L_{sum} за межі 0,3-20,1 м, а також масштабування розроблених підходів на маніпулятори з більшою кількістю ступенів вільності.

Крім того, проведений комплексний аналіз умов праці розробника інформаційно-обчислювального комплексу для моделювання руху маніпулятора з позиції чинного законодавства України про охорону праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Overview of Industrial Process Automation [Електронний ресурс]. – [Б. м.] : Elsevier, 2017. – Режим доступу: https://doi.org/10.1016/c2015-0-01929-3.

2. Захоплювальні пристрої промислових роботів: Навчальний посібник. – Тернопіль: Тернопільський державний технічний університет ім. І. Пулюя, 2008. – 232с.

3. Das R., Baishya N. J., Bhattacharya B. A review on tele-manipulators for remote diagnostic procedures and surgery. CSI Transactions on ICT. 2023. URL: https://doi.org/10.1007/s40012-023-00373-2.

4. Automating the Handling Process of a Coffee Machine Using a Robotic Manipulator / S. Michalková et al. 2024 XV International Symposium on Industrial Electronics and Applications (INDEL), Banja Luka, Bosnia and Herzegovina, 6–8 November 2024. 2024. P. 1–5. URL: https://doi.org/10.1109/indel62640.2024.10772691.

5. Collaborative Bricklaying with a Robotic arm Using a Head-Mounted Camera in Construction / Y. Hwang et al. Journal of Construction Engineering and Management. 2025. Vol. 151, no. 12. URL: https://doi.org/10.1061/jcemd4.coeng-16853.

6. Lightweight underwater robot developed for archaeological surveys and excavations / S. Hotta et al. ROBOMECH Journal. 2023. Vol. 10, no. 1. URL: https://doi.org/10.1186/s40648-023-00240-4.

7. Robots in Agriculture: Revolutionizing Farming Practices / A. J. Moshayedi et al. EAI Endorsed Transactions on AI and Robotics. 2024. Vol. 3. URL: https://doi.org/10.4108/airo.5855.

8. Sasiadek J. Z. Space robotics and manipulators – The past and the future. Control Engineering Practice. 1994. Vol. 2, no. 3. P. 491–497. URL: https://doi.org/10.1016/0967-0661(94)90787-0.

9. Underwater manipulators: A review / S. Sivčev et al. Ocean Engineering. 2018. Vol. 163. P. 431–450. URL: https://doi.org/10.1016/j.oceaneng.2018.06.018.

10. Ishola A. A., Whidborne J. F., Tang G. An Aircraft-Manipulator System for Virtual Flight Testing of Longitudinal Flight Dynamics. Robotics. 2024. Vol. 13, no. 12. P. 179. URL: https://doi.org/10.3390/robotics13120179.

11. SVM-Based Classification of sEMG Signals for Upper-Limb Self-Rehabilitation Training / S. Cai et al. Frontiers in Neurorobotics. 2019. Vol. 13. URL: https://doi.org/10.3389/fnbot.2019.00031.

12. Language Understanding for Field and Service Robots in a Priori Unknown Environments / M. Walter et al. Field Robotics. 2022. Vol. 2, no. 1. P. 1191–1231. URL: https://doi.org/10.55417/fr.2022040.

13. Symbiotic System of Systems Design for Safe and Resilient Autonomous Robotics in Offshore Wind Farms / D. Mitchell et al. IEEE Access. 2021. Vol. 9. P. 141421–141452. URL: https://doi.org/10.1109/access.2021.3117727.

14. Davis B., Johnson R. Development of a Workshop on Automated Vehicle Technologies. Minneapolis : University of Minnesota, 2019. 27 p. URL: https://cts-d10resmod-prd.oit.umn.edu/pdf/cts-19-23.pdf.

15. Automated Scanning Techniques Using UR5 / H. Lopez-Hawa et al. Journal of Robotics. 2019. Vol. 2019. P. 1–8. URL: https://doi.org/10.1155/2019/5304267.

16. Sharma S. What are manipulator robots? Understanding their Design, Types, and Applications. Wevolver. URL: https://www.wevolver.com/article/robot-manipulator.

17. What is Delta Robot? Advantages, disadvantages and practical applications. DNC. URL: https://www.dnc-automation.com/what-is-delta-robot/.

18. Advantages and Disadvantages of Articulated Robots - Robots Done Right. Robots Done Right. URL: https://robotsdoneright.com/Articles/advantages-

and-disadvantages-of-articulated-robots.html?srsltid=AfmBOorOu9W8ro4L08y8T400d2Xn0age-UuzHRpNq25HzxG1DG46zkYn.

19. SCARA robots | autonox Robotics. Startseite | autonox Robotics. URL: https://www.autonox.com/en/scara-robot.

20. Advantages and Disadvantages of Cobots - Robots Done Right. Robots Done Right - Used Robot Sales. URL: https://robotsdoneright.com/Articles/advantages-and-disadvantages-of-collaborative-robots.html?srsltid=AfmBOoopvHLJ6Yx6RZD6J0Tsvp_lj861JBp9v_OE1Rohs3vR5cGm51i7.

21. The high-mix low-volume production and cobots. industry24h. URL: https://www.industry24h.com/the-high-mix-low-volume-production-and-cobots/.

22. So J., Lee I.-B., Kim S. Federated Learning-Based Framework to Improve the Operational Efficiency of an Articulated Robot Manufacturing Environment. Applied Sciences. 2025. Vol. 15, no. 8. P. 4108. URL: https://doi.org/10.3390/app15084108.

23. Articulated Robot Market Size, Share | Industry Report, 2034. Consulting & Growth Advisory Services | Fortune Business Insights. URL: https://www.fortunebusinessinsights.com/articulated-robots-market-106143.

24. Articulated Robot Market: 2030 Global Insights & Forecast. nextmsc.com. URL: https://www.nextmsc.com/report/articulated-robot-market.

25. Fazilat M., Zioui N. Quantum-Inspired Sliding-Mode Control to Enhance the Precision and Energy Efficiency of an Articulated Industrial Robotic Arm. Robotics. 2025. Vol. 14, no. 2. P. 14. URL: https://doi.org/10.3390/robotics14020014.

26. Vijay M., Jena D. Backstepping terminal sliding mode control of robot manipulator using radial basis functional neural networks. Computers & Electrical

Engineering. 2018. Vol. 67. P. 690–707. URL: https://doi.org/10.1016/j.compeleceng.2017.11.007.

27. Sameh A., Fanni M., Rashad M. Advances in intelligent industrial manipulators for smart manufacturing and standardized automation technologies. Discover Robotics. 2025. Vol. 1, no. 1. URL: https://doi.org/10.1007/s44430-025-00012-2.

28. Articulated Robot Arm. Automated Solutions Australia. URL: https://automatedsolutions.com.au/articulated-robot-arm/.

29. Yen V. T., Nan W. Y., Van Cuong P. Recurrent fuzzy wavelet neural networks based on robust adaptive sliding mode control for industrial robot manipulators. Neural Computing and Applications. 2018. Vol. 31, no. 11. P. 6945–6958. URL: https://doi.org/10.1007/s00521-018-3520-3.

30. Фу К. Робототехника / К. Фу, Р. Гонсалес, К. Ли. – М.: Мир,– 1989. — 621 с.

31. Ащепкова Н. С. Mathcad in the kinematic and dynamic analysis of the manipulator. Eastern-European Journal of Enterprise Technologies. 2015. Vol. 5, no. 7(77). P. 54. URL: https://doi.org/10.15587/1729-4061.2015.51105.

32. Spinu G. A. (1991) Industrial works. 2nd ed., Kyiv: Higher School.

33. Spong M. W., Hutchinson S., Vidyasagar M. Robot Modeling and Control. New York: John Wiley & Sons, Inc., 2006.

34. Corke P. I. A Simple and Systematic Approach to Assigning Denavit–Hartenberg Parameters. IEEE Transactions on Robotics. 2007. Vol. 23, no. 3. P. 590–594. URL: https://doi.org/10.1109/tro.2007.896765.

35. Korovin O. S. Review of methods for solving the inverse kinematics problem for an arm with redundancy. Politechnical student journal. 2022. No. 77. URL: https://doi.org/10.18698/2541-8009-2022-12-846.

36. Kalaycioglu S., de Ruiter A., Fung E., Zhang H., Xie H. Analytical solution for inverse kinematics [Електронний ресурс] // arXiv:2410.22582 [cs.RO]. – 2024. – 9 p. – Режим доступу: https://doi.org/10.48550/arXiv.2410.22582

37. Floreano D., Husbands P., Nolfi S. Evolutionary Robotics. Springer Handbook of Robotics / 1th ed. by B. Siciliano, O. Khatib. 2008.

38. Wagaa N., Kallel H., Mellouli N. Analytical and deep learning approaches for solving the inverse kinematic problem of a high degrees of freedom robotic arm. Engineering Applications of Artificial Intelligence. 2023. Vol. 123. P. 106301. URL: https://doi.org/10.1016/j.engappai.2023.106301.

39. A Novel Method for Inverse Kinematics Solutions of Space Modular Self-Reconfigurable Satellites with Self-Collision Avoidance / J. An et al. Aerospace. 2022. Vol. 9, no. 3. P. 123. URL: https://doi.org/10.3390/aerospace9030123.

40. An efficient hybrid motion planning framework for redundant space manipulators in constrained environments / J. Zhao et al. Acta Astronautica. 2026. Vol. 240. P. 635–649. URL: https://doi.org/10.1016/j.actaastro.2025.12.025.

41. Courrieu P. Fast Computation of Moore-Penrose Inverse Matrices. Neural Information Processing - Letters and Reviews 2005. Vol. 8(2). PP. 25-29. DOI: https://doi.org/10.48550/arXiv.0804.4809.

42. Dormanl JD, Guptazb A, Gill HS, Kumar V. A novel adaptive Levenberg-Marquardt algorithm for industrial robot inverse kinematics. 1th ed. In: Integrated Technologies in Electrical, Electronics and Biotechnology Engineering. London: CRC Press; 2025, pp. 501–505. DOI: 10.1201/9781003606208-85.

43. Системи управління маніпуляційних роботів: Методичні вказівки до лабораторних робіт / О. О. Черно, А. П. Гуров, М. В. Покровський, О. А. Авдєєва. – Миколаїв: НУК, 2023. – 41 с.

44. Trajectory Generation (2/2). UCLA | Bionics Lab. URL: http://bionics.seas.ucla.edu/education/MAE_263D/_Robotics_05_Trajectory_09_Task_Part_2.pdf.

45. Lyche T., Mørken K. Spline Methods. Department of Informatics, Centre of Mathematics for Applications, University of Oslo, 2008.

46. Герасін О.С., Мельникова О.Є. та ін. Комп'ютерне моделювання кінематики триланкового робота-маніпулятора. *Електронне моделювання*. 2026. Т. 43, № 3. С. 111–126. URL: https://doi.org/10.15407/elmodel.48.03.111.

47. Designing and Performance-Analysis of a 3 DOF Robotic Manipulator Arm and its Higher Order Integration for 7 DOF Robotic Arm / A. K. Nuhel et al. 2022 4th International Conference on Sustainable Technologies for Industry 4.0 (STI), Dhaka, Bangladesh, 17–18 December 2022. 2022. URL: https://doi.org/10.1109/sti56238.2022.10103352.

48. Синтез роботехнічних систем в машинобудуванні / Л. Пелевін [та ін.]. – Київ : Київ. нац. ун-т буд-ва і архітектури, 2016. – 258 с.

49.Фу К., Гонсалес Р., Ли К. Робототехніка: Пер. з англ. – М.: Мир, 1989. – 624 с.

50. Bruno, S. et al. (2009) Robotics: Modelling, Planning and Control. London: Springer-Verlag.

51. Rao A. Trajectory Generation – Modeling, Motion Planning, and Control of Manipulators and Mobile Robots. Modeling, Motion Planning, and Control of Manipulators and Mobile Robots. URL: https://opentextbooks.clemson.edu/wangrobotics/chapter/trajectory-generation/.

52.Topic7_Trajectory_Generation. oramosp.epizy.com. URL: https://oramosp.epizy.com/teaching/18/robotics/lectures/Topic7_Trajectory_Generation.pdf?i=3.

53. Lyche T., Mørken K. Spline Methods. Department of Informatics, Centre of Mathematics for Applications, University of Oslo, 2008.

54. L. Tao, L. Liu, Z. Qin, O. Gerasin, O. Melnykova. Adaptive gradient descent algorithm for inverse kinematics of a 3-DOF robotic arm, PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A. (in print).

55. Salisbury J. K., Craig J. J. Articulated Hands. The International Journal of Robotics Research. 1982. Vol. 1, no. 1. P. 4–17. URL: https://doi.org/10.1177/027836498200100102.

56. Hosseini M. A., Daniali H. M. Weighted local conditioning index of a positioning and orienting parallel manipulator. Scientia Iranica. 2011. Vol. 18, no. 1. P. 115–120. URL: https://doi.org/10.1016/j.scient.2011.03.013.

57. Angeles J. Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms. Springer London, Limited, 2013.

58. Мельникова О.Є., Герасін О.С. Інформаційно-обчислювальний комплекс для моделювання кінематики робота-маніпулятора. Інформаційні технології в металургії та машинобудуванні. : Матеріали міжнар. науково-техн. конф., м. Дніпро, 23-24 квіт. 2025 р. 2025. С. 324–329. ISSN: 2708-0102. URL: https://journals.nmetau.edu.ua/index.php/itmm/issue/view/153.

59. Про охорону праці. Офіційний вебпортал парламенту України. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text.

60. Охорона праці – покроковий алгоритм впровадження. Довідник спеціаліста з охорони праці. URL: https://pro-op.com.ua/article/378-organzatsya-ohoroni-prats.

61. Заіченко В. І. Курс лекцій з дисципліни «Управління охороною праці». Харків : ХНУМГ ім. О. М. Бекет., 2017. 75 с.

62. ГОСТ 12.0.003-74 Небезпечні та шкідливі виробничі фактори. Класифікація. БУДСТАНДАРТ Online - нормативні документи будівельної галузі України. URL: https://online.budstandart.com/ua/catalog/doc-page?id_doc=48127.

63. Конспект лекцій з дисципліни «Охорона праці у невиробничий сфері». Кафедра охорони праці, промислової та цивільної безпеки | Офіційний сайт. URL: https://opcb.kpi.ua/wp-content/uploads/2014/09/KoHcnekT_OPG14.pdf.

64. Інструкція з безпеки життєдіяльності, охорони праці під час роботи на персональному комп’ютері. Освітній проект «На Урок» для вчителів. URL: https://naurok.com.ua/instrukciya-z-bezpeki-zhittediyalnosti-ohoroni-praci-pid-chas-roboti-na-personalnomu-komp-yuteri-501574.html.

65. КЗ «Татарбунарський ЗЗСО І-ІІІ ступенів ім. В.З.Тура». Інструкція № 26 з охорони праці при роботі з персональним комп'ютером. Татарбун. міськрада Од. обл.

66. Лекція № 14 Ергономічні вимоги до організації і обладнання робочих місць з комп`ютером. StudFiles. URL: https://studfile.net/preview/8990203/page:13/.

67. Ергономічні вимоги для організації робочого місця. Портал споживача. URL: https://gpp.livingplanet.org.ua/robota/ergonomichni-vimogi-dlya-organizatsiji-robochogo-mistsya.html.

68. Вимоги до освітлення робочого місця при роботі з пк. Штучне освітлення за устроєм і призначенням яке буває?. StudFiles. URL: https://studfile.net/preview/10057092/page:11/.

69. Робота за комп’ютером: вимоги охорони праці у 2026 році. 7eminar. URL: https://7eminar.ua/news/18368-robota-za-kompyuterom-vimogi-oxoroni-praci-u-2026-roci.

70. Інструкція з охорони праці при роботі з комп'ютером, принтером, ксероксом та іншою оргтехнікою. Інструкції для навчальних закладів України. URL: https://osvita-docs.com/node/41.

71. Юсіна А. Л. Охорона праці та безпека при надзвичайних ситуаціях: рекомендації щодо виконання розділу у дипломному проекті для студентів економічних спеціальностей. Краматорськ: ДДМА, 2016. 90 с.

72. Коли не дозволяється виконувати роботу з електроінструментом - Охорона праці і пожежна безпека. Охорона праці і пожежна безпека. URL: https://oppb.com.ua/news/koly-ne-dozvolyayetsya-vykonuvaty-robotu-z-elektroinstrumentom.

ДОДАТОК А

КОД ПРОГРАМИ ІНФОРМАЦІЙНО-ОБЧИСЛЮВАЛЬНОГО КОМПЛЕКСУ

```
%% Ініціалізація функції запуску інформаційно-обчислювального комплексу
function Project_OpeningFcn(hObject, eventdata, handles, varargin)

%% Ініціалізація початкової позиції робота
handles.current_position = [-2*pi/3 pi/6 -5*pi/36]; % Початкові кути



%% ГРАФІЧНИЙ ІНТЕРФЕЙС КОРИСТУВАЧА GUI (ІНІЦІАЛІЗАЦІЯ ВІДПОВІДНИХ ПОЛІВ ВВЕДЕННЯ ПАРАМЕТРІВ МАНІПУЛЯТОРА)


%% Поля введення довжин ланок

function L1_Callback(hObject, eventdata, handles)
function L2_Callback(hObject, eventdata, handles)
function L3_Callback(hObject, eventdata, handles)

%% Поля для введення кутів сполучень Theta1, Theta2, Theta3
function Theta_1_Callback(hObject, eventdata, handles)
function Theta_2_Callback(hObject, eventdata, handles)
function Theta_3_Callback(hObject, eventdata, handles)

%% Поля для введення бажаних координат захвату маніпулятора X, Y, Z
function Pos_X_Callback(hObject, eventdata, handles)
function Pos_Y_Callback(hObject, eventdata, handles)
function Pos_Z_Callback(hObject, eventdata, handles)

%% Ініціалізація та побудова графіків зміни кутів ланок (movement_angle_of_each_join), зміни швидкості кутів (angular_velocity_of_each_join), прискорення зміни кутів (angular_acceleration_of_each_join) та анімації робота-маніпулятора (graphik_of_animation_robot) відбувається через handles та animate у відповідних підфункціях та оброблювачах подій


%% Поля введення допустимих діапазонів руху суглобів для кутів Theta_1, Theta_2, Theta_3 з викликом функції update_slider_limits для обмеження Min Max значень відповідного слайдера
function Th1_workspase_max_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);
function Th1_workspase_min_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);

function Th2_workspase_max_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);
function Th2_workspase_min_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);

function Th3_workspase_max_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);
function Th3_workspase_min_Callback(hObject, eventdata, handles)
    update_slider_limits(handles);
```

```
%% МОДУЛЬ ПРЯМОЇ КІНЕМАТИКИ

%% Функція для зміни меж полів у слайдері update_slider_limits()
function update_slider_limits(handles)
    % Отримуємо значення меж з текстових полів
    q1_min_str = get(handles.Th1_workspase_min, 'String');
    q1_max_str = get(handles.Th1_workspase_max, 'String');
    q2_min_str = get(handles.Th2_workspase_min, 'String');
    q2_max_str = get(handles.Th2_workspase_max, 'String');
    q3_min_str = get(handles.Th3_workspase_min, 'String');
    q3_max_str = get(handles.Th3_workspase_max, 'String');

    % Оновлення меж для Theta 1
    if ~isempty(q1_min_str) && ~isempty(q1_max_str)

        % Конвертування значень з тексту в числа
        q1_min_deg = str2double(q1_min_str);
        q1_max_deg = str2double(q1_max_str);

        % Перевірка на успішність здійснення конвертування та відповідність
        % значень min та max
        if ~isnan(q1_min_deg) && ~isnan(q1_max_deg) && q1_min_deg < q1_max_deg

            % Встановлюємо нові значення меж
            set(handles.Theta_1_slider, 'Min', q1_min_deg);
            set(handles.Theta_1_slider, 'Max', q1_max_deg);

            % Перевіряємо поточне значення
            current_val = get(handles.Theta_1_slider, 'Value');

            % Встановлюємо найменше значення з обмеженого діапазона
            if current_val < q1_min_deg
                set(handles.Theta_1_slider, 'Value', q1_min_deg);
            % Встановлюємо найбільше значення з обмеженого діапазона
            elseif current_val > q1_max_deg
                set(handles.Theta_1_slider, 'Value', q1_max_deg);
            end
        end
    else
        % Якщо поля пусті, встановлюємо стандартні межі
        set(handles.Theta_1_slider, 'Min', -180);
        set(handles.Theta_1_slider, 'Max', 180);
    end

    % Оновлення меж для Theta 2
    if ~isempty(q2_min_str) && ~isempty(q2_max_str)
        q2_min_deg = str2double(q2_min_str);
        q2_max_deg = str2double(q2_max_str);

        if ~isnan(q2_min_deg) && ~isnan(q2_max_deg) && q2_min_deg < q2_max_deg
            set(handles.Theta_2_slider, 'Min', q2_min_deg);
            set(handles.Theta_2_slider, 'Max', q2_max_deg);

            % Перевіряємо поточне значення
            current_val = get(handles.Theta_2_slider, 'Value');
            if current_val < q2_min_deg
                set(handles.Theta_2_slider, 'Value', q2_min_deg);
            elseif current_val > q2_max_deg
                set(handles.Theta_2_slider, 'Value', q2_max_deg);
            end
        end
```

```
    else
        % Якщо поля пусті, встановлюємо стандартні межі
        set(handles.Theta_2_slider, 'Min', -180);
        set(handles.Theta_2_slider, 'Max', 180);
    end

    % Оновлення меж для Theta 3
    if ~isempty(q3_min_str) && ~isempty(q3_max_str)
        q3_min_deg = str2double(q3_min_str);
        q3_max_deg = str2double(q3_max_str);

        if ~isnan(q3_min_deg) && ~isnan(q3_max_deg) && q3_min_deg < q3_max_deg
            set(handles.Theta_3_slider, 'Min', q3_min_deg);
            set(handles.Theta_3_slider, 'Max', q3_max_deg);

            % Перевіряємо поточне значення
            current_val = get(handles.Theta_3_slider, 'Value');
            if current_val < q3_min_deg
                set(handles.Theta_3_slider, 'Value', q3_min_deg);
            elseif current_val > q3_max_deg
                set(handles.Theta_3_slider, 'Value', q3_max_deg);
            end
        end
    else
        % Якщо поля пусті, встановлюємо стандартні межі
        set(handles.Theta_3_slider, 'Min', -180);
        set(handles.Theta_3_slider, 'Max', 180);
    end

    % Оновлюємо робота після зміни меж
    update_robot(handles);
end


%% Значення координат захвату через пряму кінематику зі слайдерами зміни кутів ланок,
підфункція update_robot()
function update_robot(handles)
    % Отримуємо значення меж з текстових полів
    q1_min_str = get(handles.Th1_workspase_min, 'String');
    q1_max_str = get(handles.Th1_workspase_max, 'String');
    q2_min_str = get(handles.Th2_workspase_min, 'String');
    q2_max_str = get(handles.Th2_workspase_max, 'String');
    q3_min_str = get(handles.Th3_workspase_min, 'String');
    q3_max_str = get(handles.Th3_workspase_max, 'String');

    % Отримуємо значення кутів зі слайдерів
    AngleOfTheta_1_deg = get(handles.Theta_1_slider, 'Value');
    AngleOfTheta_2_deg = get(handles.Theta_2_slider, 'Value');
    AngleOfTheta_3_deg = get(handles.Theta_3_slider, 'Value');

    % Параметри довжин ланок
    LengthOfLink_1 = str2double(get(handles.L1, 'String'));
    LengthOfLink_2 = str2double(get(handles.L2, 'String'));
    LengthOfLink_3 = str2double(get(handles.L3, 'String'));

    % Конвертуємо в радіани
    AngleOfTheta_1_rad = AngleOfTheta_1_deg * (pi/180);
    AngleOfTheta_2_rad = AngleOfTheta_2_deg * (pi/180);
    AngleOfTheta_3_rad = AngleOfTheta_3_deg * (pi/180);

    % Оновлення значень текстових полів після зміни значення на слайдері
    set(handles.Theta_1, 'string', num2str(AngleOfTheta_1_deg, '%.3f'));
```

```
    set(handles.Theta_2, 'string', num2str(AngleOfTheta_2_deg, '%.3f'));
    set(handles.Theta_3, 'string', num2str(AngleOfTheta_3_deg, '%.3f'));

    % Створення моделі маніпулятора
    LengthOfLink(1) = Link([0 LengthOfLink_1 0 pi/2]);
    LengthOfLink(2) = Link([0 0 LengthOfLink_2 0]);
    LengthOfLink(3) = Link([0 0 LengthOfLink_3 0]);
    Robot = SerialLink(LengthOfLink);
    Manipulator_RRR_antropomorf.name = 'Manipulator_RRR';

    % Розвязок прямої задачі кінематики
    T = Manipulator_RRR_antropomorf.fkine([AngleOfTheta_1_rad AngleOfTheta_2_rad
AngleOfTheta_3_rad]);
    T_matrix = T.T;

    % Оновлюємо координати X, Y, Z в GUI
    set(handles.Pos_X, 'String', num2str(T_matrix(1,4), '%.3f'));
    set(handles.Pos_Y, 'String', num2str(T_matrix(2,4), '%.3f'));
    set(handles.Pos_Z, 'String', num2str(T_matrix(3,4), '%.3f'));
end
```

```
%% МОДУЛЬ ЗВОРОТНОЇ КІНЕМАТИКИ


%% Обролювач події btn_Inverse_Callback()
function btn_Inverse_Callback(hObject, eventdata, handles)
% Зчитування координат цілі з GUI
x_target = str2double(get(handles.Pos_X, 'String'));
y_target = str2double(get(handles.Pos_Y, 'String'));
z_target = str2double(get(handles.Pos_Z, 'String'));

% Зчитування довжин ланок з GUI
LengthOfLink_1 = str2double(get(handles.L1, 'String'));
LengthOfLink_2 = str2double(get(handles.L2, 'String'));
LengthOfLink_3 = str2double(get(handles.L3, 'String'));

% Обчислення загальної довжини маніпулятора
totalLenghtOfLinght = LengthOfLink_1 + LengthOfLink_2 + LengthOfLink_3;

%% Новий спосіб обчислення ІК
% Створення структури даних довжин ланок маніпулятора
paramsLenghtOfJoins.LengthOfLink_1 = LengthOfLink_1;
paramsLenghtOfJoins.LengthOfLink_2 = LengthOfLink_2;
paramsLenghtOfJoins.LengthOfLink_3 = LengthOfLink_3;

% Створення структури даних координат бажаної цільової точки
paramsPositionOfDesiredPoint.X = x_target;
paramsPositionOfDesiredPoint.Y = y_target;
paramsPositionOfDesiredPoint.Z = z_target;

if isfield(handles, 'current_position')
    paramsStartPositionThetaInRad.AngleOfTheta_1 = handles.current_position(1);
    paramsStartPositionThetaInRad.AngleOfTheta_2 = handles.current_position(2);
    paramsStartPositionThetaInRad.AngleOfTheta_3 = handles.current_position(3);
else
    % Резервні значення — лише якщо current_position ще не існує
    paramsStartPositionThetaInRad.AngleOfTheta_1 = -2*pi/3;
    paramsStartPositionThetaInRad.AngleOfTheta_2 =  pi/6;
    paramsStartPositionThetaInRad.AngleOfTheta_3 = -5*pi/36;
end

% Зчитування значень діапазонів руху кутів зчленувань маніпулятора GUI
q1_min = str2double(get(handles.Th1_workspase_min, 'String'))*pi/180;
q1_max = str2double(get(handles.Th1_workspase_max, 'String'))*pi/180;
q2_min = str2double(get(handles.Th2_workspase_min, 'String'))*pi/180;
q2_max = str2double(get(handles.Th2_workspase_max, 'String'))*pi/180;
q3_min = str2double(get(handles.Th3_workspase_min, 'String'))*pi/180;
q3_max = str2double(get(handles.Th3_workspase_max, 'String'))*pi/180;

%% Перевірка на досяжність точки та обчислення зворотної кінематики
R = sqrt((x_target^2)+(y_target^2));
h = sqrt((R^2)+((z_target - LengthOfLink_1)^2));

if h <= abs(LengthOfLink_2-LengthOfLink_3)+0.001 || h >= LengthOfLink_2+LengthOfLink_3-
0.001 || R <= abs(LengthOfLink_2-LengthOfLink_3)+0.001 || R >=
LengthOfLink_2+LengthOfLink_3-0.001
    msgbox(sprintf('Точка виходить за межі робочої області маніпулятора.\n Введіть інші
координати.'), 'Помилка', 'error');
    return;
else
    thetaFromIK = IKSolutionWithStepAdaptetion(paramsLenghtOfJoins,
paramsPositionOfDesiredPoint, paramsStartPositionThetaInRad, totalLenghtOfLinght);
end
```

```
%% Змінна для зберігання повідомлення причини помилки
failureReason = '';
ranges_defined = ~isnan(q1_min) && ~isnan(q1_max) && ~isnan(q2_min) && ~isnan(q2_max)
&& ~isnan(q3_min) && ~isnan(q3_max);

if isempty(thetaFromIK)
    failureReason = 'Ітераційний метод градієнтного спуску не зміг забезпечити
необхідну точність обрахунку кутів для досягенння заданої точки.';
end

if isempty(failureReason) && ranges_defined
    outOfRange = thetaFromIK(1) < q1_min || thetaFromIK(1) > q1_max || ...
                 thetaFromIK(2) < q2_min || thetaFromIK(2) > q2_max || ...
                 thetaFromIK(3) < q3_min || thetaFromIK(3) > q3_max;
    if outOfRange
        failureReason = sprintf( ...
        ['Отримані кути виходять за допустимі межі:\n\n' ...
         'Theta 1 = %.2f° (%.2f° … %.2f°)\n' ...
         'Theta 2 = %.2f° (%.2f° … %.2f°)\n' ...
         'Theta 3 = %.2f° (%.2f° … %.2f°)\n'], ...
        rad2deg(thetaFromIK(1)), rad2deg(q1_min), rad2deg(q1_max), ...
        rad2deg(thetaFromIK(2)), rad2deg(q2_min), rad2deg(q2_max), ...
        rad2deg(thetaFromIK(3)), rad2deg(q3_min), rad2deg(q3_max));
    end
end

if ~isempty(failureReason)
    choice = questdlg([failureReason 'Бажаєте виконати розрахунок геометричним
методом?'], ...
         'Помилка оберненої кінематики', 'Так', 'Ні', 'Так');
    switch choice
        case 'Так'
            thetaFromGeometricIK = GeometricIKSolution(paramsLenghtOfJoins,
paramsPositionOfDesiredPoint, ...
                                    q1_min, q1_max, q2_min, q2_max, q3_min, q3_max,
ranges_defined);
            if isempty(thetaFromGeometricIK)
                msgbox('Геометричний метод також не знайшов рішення.', 'Помилка',
'error');
                return;
            end
        otherwise
            return;
    end
end

if exist('thetaFromGeometricIK', 'var')
    thetaFinal = thetaFromGeometricIK;
else
    thetaFinal = thetaFromIK;
end

% Оновлення полів виведення кутів в GUI
set(handles.Theta_1, 'String', sprintf('%.3f', rad2deg(thetaFinal(1))));
set(handles.Theta_2, 'String', sprintf('%.3f', rad2deg(thetaFinal(2))));
set(handles.Theta_3, 'String', sprintf('%.3f', rad2deg(thetaFinal(3))));
set(handles.Theta_1_slider, 'Value', rad2deg(thetaFinal(1)));
set(handles.Theta_2_slider, 'Value', rad2deg(thetaFinal(2)));
set(handles.Theta_3_slider, 'Value', rad2deg(thetaFinal(3)));

%% Створення моделі маніпулятора
```

```
LengthOfLink(1) = Link([0 LengthOfLink_1 0 pi/2]);
LengthOfLink(2) = Link([0 0 LengthOfLink_2 0]);
LengthOfLink(3) = Link([0 0 LengthOfLink_3 0]);
Robot = SerialLink(LengthOfLink);
Manipulator_RRR_antropomorf.name = 'Manipulator_RRR';

axes(handles.graphik_of_animation_robot);
cla(handles.graphik_of_animation_robot);
view(3); grid on; axis equal;

%% Планування траєкторії (рівно 50 точок)
tf = 10;          % Загальний час руху (с)
n_points = 50;    % Фіксована кількість точок
dt = tf / (n_points - 1); % Розрахунок кроку для отримання рівно 50 точок

if isfield(handles, 'current_position')
    q_start = handles.current_position;
else
    q_start = [-2*pi/3 pi/6 -5*pi/36];
end
q_end = [thetaFinal(1) thetaFinal(2) thetaFinal(3)];

% Формування граничних умов для функцій поліномів
x1 = [q_start(1), q_end(1)]; x_d1 = [0, 0]; x_dd1 = [0, 0];
x2 = [q_start(2), q_end(2)]; x_d2 = [0, 0]; x_dd2 = [0, 0];
x3 = [q_start(3), q_end(3)]; x_d3 = [0, 0]; x_dd3 = [0, 0];

% Розрахунок коефіцієнтів для траєкторії кожного суглобу
coef1 = trajectory_blend_quintic(x1, x_d1, x_dd1, tf,
handles.graphik_of_animation_robot);
coef2 = trajectory_blend_quintic(x2, x_d2, x_dd2, tf,
handles.graphik_of_animation_robot);
coef3 = trajectory_blend_quintic(x3, x_d3, x_dd3, tf,
handles.graphik_of_animation_robot);

[t, q_angle, q_d_vel, q_d2_accel, validation_result] = ...
    compute_trajectory_with_validation(...
        coef1, coef2, coef3, x1, x2, x3, tf, dt, q_start, ...
        q1_min, q1_max, q2_min, q2_max, q3_min, q3_max);


if ~isempty(validation_result.reason)
    if ~validation_result.success
        % Більше 30% пропущено — серйозне попередження
        warndlg(sprintf(['Увага: %s\n\n' ...
            'Траєкторію побудовано, але значна частина шляху ' ...
            'проходить поза межами суглобів.\n' ...
            'Перевірте задані обмеження або цільову точку.'], ...
            validation_result.reason), 'Попередження траєкторії');
    else
        % Менше 30% — інформаційне повідомлення
        fprintf('[Траєкторія] %s\n', validation_result.reason);
    end
    % рух продовжується
end

workspace_limit = totalLenghtOfLinght;
axis_limits = [-workspace_limit workspace_limit -workspace_limit workspace_limit -
workspace_limit workspace_limit];

trajectory_points = zeros(n_points, 3);
```

```
% Попередній розрахунок декартових координат точок робочого органу
for i = 1:n_points
    T = Manipulator_RRR_antropomorf.fkine(q_angle(i, :));
    trajectory_points(i, :) = T.t';
end

%% Налаштування кольору кожної з 50 точок
% Матриця кольорів RGB розміром 50x3.
% Палітра 'jet' для плавного спектрального переходу
trajectory_colors = jet(n_points);

Manipulator_RRR_antropomorf.plot(q_angle(1, :), 'workspace', axis_limits, 'notiles',
'noname');
hold on;

% Графічна ініціалізація траєкторії за допомогою scatter3
% Розмір точок 40, маркер заповнений ('filled')
trajectory_handle = scatter3(trajectory_points(1,1), trajectory_points(1,2),
trajectory_points(1,3), ...
    40, trajectory_colors(1,:), 'filled');

axis(axis_limits);

%% Цикл анімації руху по 50 кольорових точках
for i = 2:n_points
    Manipulator_RRR_antropomorf.animate(q_angle(i, :));

    % Динамічне додавання точок на графік з оновленням масиву кольорів CData
    set(trajectory_handle, 'XData', trajectory_points(1:i,1), ...
                           'YData', trajectory_points(1:i,2), ...
                           'ZData', trajectory_points(1:i,3), ...
                           'CData', trajectory_colors(1:i,:));

    axis(axis_limits);
    pause(0.06); % Час затримки для плавної візуалізації 50 кроків
    drawnow;
end

hold off;
handles.current_position = q_end;
guidata(hObject, handles);

%% Оновлення графіків кутів повороту ланок, зміни швидкості кутів, прискорення зміни
кутів
axes(handles.movement_angle_of_each_join); cla(handles.movement_angle_of_each_join);
plot(t, q_angle, 'LineWidth', 1.5);
xlabel('Час (с)'); ylabel('\theta (рад)'); title('Зміна кутів повороту ланок');
legend('\theta_1','\theta_2','\theta_3'); grid on;

axes(handles.angular_velocity_of_each_join);
cla(handles.angular_velocity_of_each_join);
plot(t, q_d_vel, 'LineWidth', 1.5);
xlabel('Час (с)'); ylabel(' \omega (рад/с)'); title('Зміна швидкості кутів');
legend('\omega_1','\omega_2','\omega_3'); grid on;

axes(handles.angular_acceleration_of_each_join);
cla(handles.angular_acceleration_of_each_join);
plot(t, q_d2_accel, 'LineWidth', 1.5);
xlabel('Час (с)'); ylabel('\alpha (рад/с^2)'); title('Прискорення зміни кутів');
legend('\alpha_1','\alpha_2','\alpha_3'); grid on;
```

```
%% Алгоритм обчислення зворотної кінематики методом градієнтного спуску
IKSolutionWithStepAdaptetion

function thetaFromIK = IKSolutionWithStepAdaptetion(paramsLenghtOfJoins,
paramsPositionOfDesiredPoint, paramsStartPositionThetaInRad, totalLenghtOfLinght)

% Отримання довжин ланок
LengthOfLink_1 = paramsLenghtOfJoins.LengthOfLink_1;
LengthOfLink_2 = paramsLenghtOfJoins.LengthOfLink_2;
LengthOfLink_3 = paramsLenghtOfJoins.LengthOfLink_3;

% Отримання координат бажаної цільової точки
x_target = paramsPositionOfDesiredPoint.X;
y_target = paramsPositionOfDesiredPoint.Y;
z_target = paramsPositionOfDesiredPoint.Z;

% Початкове положення маніпулятора
AngleOfTheta_1 = paramsStartPositionThetaInRad.AngleOfTheta_1;
AngleOfTheta_2 = paramsStartPositionThetaInRad.AngleOfTheta_2;
AngleOfTheta_3 = paramsStartPositionThetaInRad.AngleOfTheta_3;


% Таблиця діапазонів і коефіцієнтів h
ranges = [0.3  0.7;
          0.9  1.3;
          1.5  2.1;
          2.3  2.7;
          2.9  3.1;
          3.3  5.3;
          5.5  5.7;
          5.9  6.3;
          6.5  7.5;
          7.7  8.9;
          9.1  11.7;
          11.9 15.3;
          15.5 18.3;
          18.5 20.1];
h = [4 3 1 0.4 0.3 0.1 0.07 0.06 0.05 0.04 0.02 0.01 0.008 0.006];

%  Визначення потрібного h на основі totalLenghtOfLinght
idx = find(totalLenghtOfLinght >= ranges(:,1) & totalLenghtOfLinght <= ranges(:,2));

if isempty(idx)
    idx_above = find(totalLenghtOfLinght < ranges(:,1), 1, 'first');

    if isempty(idx_above) || idx_above == 1
        fprintf('ПОМИЛКА: totalLenghtOfLinght = %.2f перевищує максимальний діапазон
(20.1).\n', totalLenghtOfLinght);
        msgbox(sprintf('Довжина маніпулятора (%.2f) занадто велика!\nМаксимальний
діапазон: 20.1\nЗмініть довжини ланок.', totalLenghtOfLinght), 'Помилка', 'error');
        thetaFromIK = [];
        return;
    else
        % totalLenghtOfLinght між діапазонами — беремо два сусідні h
        h_values = [h(idx_above-1), h(idx_above)];
    end
else
    h_values = h(idx);
end

%  Параметри збіжності алгоритму
epsilon = 0.0001;
```

```
iter = 10000;

fprintf('\ntotalLenghtOfLinght = %.4f | Тестуємо h: '); disp(h_values);

% Флаг досягення збіжності
SolutionFound  = false;

% Змінні для зберігання остаочного розвязку ЗЗК
AngleOfTheta_1_final = AngleOfTheta_1;
AngleOfTheta_2_final = AngleOfTheta_2;
AngleOfTheta_3_final = AngleOfTheta_3;

%  Перебір h зі списку
for gi = 1:length(h_values)
    g = h_values(gi);
    fprintf('Тестування h = %.4f...\n', g);

    % Скидання кутів на початкові для кожного h
    AngleOfTheta_1_start = AngleOfTheta_1;
    AngleOfTheta_2_start = AngleOfTheta_2;
    AngleOfTheta_3_start = AngleOfTheta_3;

    for j = 1:iter
        % Пряма кінематика
        x = cos(AngleOfTheta_1_start) * (LengthOfLink_2*cos(AngleOfTheta_2_start) +
LengthOfLink_3*cos(AngleOfTheta_2_start + AngleOfTheta_3_start));
        y = sin(AngleOfTheta_1_start) * (LengthOfLink_2*cos(AngleOfTheta_2_start) +
LengthOfLink_3*cos(AngleOfTheta_2_start + AngleOfTheta_3_start));
        z = LengthOfLink_1 + LengthOfLink_2*sin(AngleOfTheta_2_start) +
LengthOfLink_3*sin(AngleOfTheta_2_start + AngleOfTheta_3_start);

        % Умова збіжності
        if abs(x_target-x) < epsilon && abs(y_target-y) < epsilon && abs(z_target-z) <
epsilon
            SolutionFound = true;
            AngleOfTheta_1_final = AngleOfTheta_1_start;
            AngleOfTheta_2_final = AngleOfTheta_2_start;
            AngleOfTheta_3_final = AngleOfTheta_3_start;
            fprintf('Рішення знайдено! h=%.4f, ітерацій=%d\n', g, j);
            fprintf('  e_x=%.8f, e_y=%.8f, e_z=%.8f\n', abs(x_target-x), abs(y_target-
y), abs(z_target-z));
            break;
        end

        % Елементи матриці Якобі (часткові похідні)
        dx1 = -sin(AngleOfTheta_1_start)*(LengthOfLink_2*cos(AngleOfTheta_2_start) +
LengthOfLink_3*cos(AngleOfTheta_2_start+AngleOfTheta_3_start));
        dx2 = -cos(AngleOfTheta_1_start)*(LengthOfLink_2*sin(AngleOfTheta_2_start) +
LengthOfLink_3*sin(AngleOfTheta_2_start+AngleOfTheta_3_start));
        dx3 = -
LengthOfLink_3*cos(AngleOfTheta_1_start)*sin(AngleOfTheta_2_start+AngleOfTheta_3_start)
;

        dy1 =  cos(AngleOfTheta_1_start)*(LengthOfLink_2*cos(AngleOfTheta_2_start) +
LengthOfLink_3*cos(AngleOfTheta_2_start+AngleOfTheta_3_start));
        dy2 = -sin(AngleOfTheta_1_start)*(LengthOfLink_2*sin(AngleOfTheta_2_start) +
LengthOfLink_3*sin(AngleOfTheta_2_start+AngleOfTheta_3_start));
        dy3 = -
LengthOfLink_3*sin(AngleOfTheta_1_start)*sin(AngleOfTheta_2_start+AngleOfTheta_3_start)
;

        dz1 = 0;
```

```
        dz2 = LengthOfLink_2*cos(AngleOfTheta_2_start) +
LengthOfLink_3*cos(AngleOfTheta_2_start+AngleOfTheta_3_start);
        dz3 = LengthOfLink_3*cos(AngleOfTheta_2_start+AngleOfTheta_3_start);

        % Оновлення кутів з фіксованим кроком h
        AngleOfTheta_1_start = wrapToPi(AngleOfTheta_1_start + g*((x_target-x)*dx1 +
(y_target-y)*dy1 + (z_target-z)*dz1));
        AngleOfTheta_2_start = AngleOfTheta_2_start  + g*((x_target-x)*dx2 + (y_target-
y)*dy2 + (z_target-z)*dz2);
        AngleOfTheta_3_start = AngleOfTheta_3_start  + g*((x_target-x)*dx3 + (y_target-
y)*dy3 + (z_target-z)*dz3);

        % Виведення для відлагодження
        if mod(j, 1000) == 0
            fprintf('  h=%.4f | iter=%d | q1=%.4f q2=%.4f q3=%.4f\n', g, j,
AngleOfTheta_1_start, AngleOfTheta_2_start, AngleOfTheta_3_start);
        end
    end

    if SolutionFound
        break;
    else
        fprintf('h = %.4f не дало результату за %d ітерацій.\n', g, iter);
    end
end

%  Формування результату
if ~SolutionFound
    fprintf('Розв''язок не знайдено за жодним h.\n');
    msgbox('Розв''язок не знайдено за максимальну кількість ітерацій.', 'Помилка',
'error');
    thetaFromIK = [];
else
    thetaFromIK = [AngleOfTheta_1_final; AngleOfTheta_2_final; AngleOfTheta_3_final];

    % Фінальні похибки
    x_f = cos(AngleOfTheta_1_final)*(LengthOfLink_2*cos(AngleOfTheta_2_final) +
LengthOfLink_3*cos(AngleOfTheta_2_final+AngleOfTheta_3_final));
    y_f = sin(AngleOfTheta_1_final)*(LengthOfLink_2*cos(AngleOfTheta_2_final) +
LengthOfLink_3*cos(AngleOfTheta_2_final+AngleOfTheta_3_final));
    z_f = LengthOfLink_1 + LengthOfLink_2*sin(AngleOfTheta_2_final) +
LengthOfLink_3*sin(AngleOfTheta_2_final+AngleOfTheta_3_final);

    fprintf('\nОбраховані кути:\n');
    fprintf('  q1 = %.6f рад (%.3f°)\n', AngleOfTheta_1_final,
rad2deg(AngleOfTheta_1_final));
    fprintf('  q2 = %.6f рад (%.3f°)\n', AngleOfTheta_2_final,
rad2deg(AngleOfTheta_2_final));
    fprintf('  q3 = %.6f рад (%.3f°)\n', AngleOfTheta_3_final,
rad2deg(AngleOfTheta_3_final));
    fprintf('Похибки: X=%.8f  Y=%.8f  Z=%.8f\n', abs(x_f-x_target), abs(y_f-y_target),
abs(z_f-z_target));
    fprintf('Розраховані точки: X=%.8f  Y=%.8f  Z=%.8f\n', x_f, y_f, z_f);
end
```

```
%% Алгоритм обчислення зворотної кінематики геометричним (аналітичним) методом
```

```
function thetaFromGeometricIK = GeometricIKSolution(paramsLenghtOfJoins,
paramsPositionOfDesiredPoint, q1_min, q1_max, q2_min, q2_max, q3_min, q3_max,
ranges_defined)

% Отримання довжин ланок
LengthOfLink_1 = paramsLenghtOfJoins.LengthOfLink_1;
LengthOfLink_2 = paramsLenghtOfJoins.LengthOfLink_2;
LengthOfLink_3 = paramsLenghtOfJoins.LengthOfLink_3;

% Отримання координат бажаної цільової точки
x_target = paramsPositionOfDesiredPoint.X;
y_target = paramsPositionOfDesiredPoint.Y;
z_target = paramsPositionOfDesiredPoint.Z;

% AngleOfTheta_1 однаковий для обох конфігурацій
AngleOfTheta_1 = atan2(y_target, x_target);

r = sqrt(x_target^2 + y_target^2 + (z_target - LengthOfLink_1)^2);

% Косинус AngleOfTheta_3 (спільний для обох конфігурацій)
cos_AngleOfTheta_3 = (r^2 - LengthOfLink_2^2 - LengthOfLink_3^2) / (2 * LengthOfLink_2
* LengthOfLink_3);

% Перевірка досяжності точки
if abs(cos_AngleOfTheta_3) > 1
    fprintf('Точка недосяжна: cos(AngleOfTheta_3) = %.4f виходить за [-1, 1].\n',
cos_AngleOfTheta_3);
    msgbox('Точка виходить за межі робочої області маніпулятора.', 'Помилка', 'error');
    thetaFromGeometricIK = [];
    return;
end

% Дві конфігурації: лікоть вгору (+) та вниз (-)
AngleOfTheta_3_up   = -acos(cos_AngleOfTheta_3);   % лікоть вгору
AngleOfTheta_3_down = acos(cos_AngleOfTheta_3);   % лікоть вниз

AngleOfTheta_2_up   = asin((z_target - LengthOfLink_1) / r) - asin((LengthOfLink_3 *
sin(AngleOfTheta_3_up)) / r);

AngleOfTheta_2_down = asin((z_target - LengthOfLink_1) / r) - asin((LengthOfLink_3 *
sin(AngleOfTheta_3_down)) / r);


%% Допоміжна функція перевірки конфігурації
    function [SolutionFound, reason] = checkConfig(t1, t2, t3)
        % Перевірка на NaN
        if isnan(t1) || isnan(t2) || isnan(t3)
            SolutionFound = false;
            reason = 'Не введено значення кутів суглобів - містять NaN.';
            return;
        end
        % Перевірка діапазонів (якщо задані)
        if ranges_defined
            if t1 < q1_min || t1 > q1_max || ...
               t2 < q2_min || t2 > q2_max || ...
               t3 < q3_min || t3 > q3_max
                SolutionFound = false;
                reason = sprintf( ...
                    ['Кути виходять за допустимі межі:\n' ...
                     'θ1 = %.3f° (%.3f°…%.3f°)\n' ...
                     'θ2 = %.3f° (%.3f°…%.3f°)\n' ...
                     'θ3 = %.3f° (%.3f°…%.3f°)'], ...
```

```
                    rad2deg(t1), rad2deg(q1_min), rad2deg(q1_max), ...
                    rad2deg(t2), rad2deg(q2_min), rad2deg(q2_max), ...
                    rad2deg(t3), rad2deg(q3_min), rad2deg(q3_max));
                return;
            end
        end
        SolutionFound = true;
        reason = '';
    end


%% Допоміжна функція виведення похибок у консоль
    function printErrors(t1, t2, t3, label)
        x_f = cos(t1) * (LengthOfLink_2*cos(t2) + LengthOfLink_3*cos(t2+t3));
        y_f = sin(t1) * (LengthOfLink_2*cos(t2) + LengthOfLink_3*cos(t2+t3));
        z_f = LengthOfLink_1 + LengthOfLink_2*sin(t2) + LengthOfLink_3*sin(t2+t3);
        fprintf('\n--- %s ---\n', label);
        fprintf('θ1=%.4f° θ2=%.4f° θ3=%.4f°\n', rad2deg(t1), rad2deg(t2), rad2deg(t3));
        fprintf('Похибки: X=%.6f  Y=%.6f  Z=%.6f\n', ...
            abs(x_f-x_target), abs(y_f-y_target), abs(z_f-z_target));
    end


%% Перевірка конфігурації лікоть вгору
[SolutionFound_up, reason_up] = checkConfig(AngleOfTheta_1, AngleOfTheta_2_up,
AngleOfTheta_3_up);

if SolutionFound_up
    % Лікоть вгору не дав бажаного результату - повернення результату
    printErrors(AngleOfTheta_1, AngleOfTheta_2_up, AngleOfTheta_3_up, 'Лікоть вгору —
успіх');
    thetaFromGeometricIK = [AngleOfTheta_1, AngleOfTheta_2_up, AngleOfTheta_3_up]';
    return;
end

% Лікоть не дав бажаного результату - повідомляємо та пропонування конфігурації лікоть
вниз
msg_up_fail = sprintf(['Конфігурація «лікоть вгору» неуспішна:\n%s\n\n' ...
                       'Бажаєте спробувати конфігурацію «лікоть вниз»?'], reason_up);

choice = questdlg(msg_up_fail, 'Геометрична зворотна кінематика', ...
                  'Так', 'Ні', 'Так');

if ~strcmp(choice, 'Так')
    % Користувач відмовився
    thetaFromGeometricIK = [];
    return;
end

%% Перевірка лікоть вниз
[SolutionFound_down, reason_down] = checkConfig(AngleOfTheta_1, AngleOfTheta_2_down,
AngleOfTheta_3_down);

if SolutionFound_down
    printErrors(AngleOfTheta_1, AngleOfTheta_2_down, AngleOfTheta_3_down, 'Лікоть вниз
— успіх');
    thetaFromGeometricIK = [AngleOfTheta_1, AngleOfTheta_2_down, AngleOfTheta_3_down]';
else
    msgbox(sprintf(['Конфігурація «лікоть вниз» також неуспішна:\n%s\n\n' ...
                    'Досягти точку неможливо жодним із способів.'], reason_down), ...
           'Помилка', 'error');
    thetaFromGeometricIK = [];
```

```
end

end
```

```
%% Підфункція розрахунку коефіцієнтів квінтичного полінома з нульовими граничними
умовами за швидкістю та прискоренням

function [coef] = trajectory_blend_quintic(x, x_d, x_dd, tf, afigure)

    t_end = tf;

    % Визначення кількості вузлових точок траєкторії
    j = length(x);

    % Створення матриці для збереження коефіцієнтів полінома
    % j-1 - кількість сегментів між точками;
    % 6 - кількість коефіцієнтів полінома 5-го порядку.
    coef = zeros(j-1, 6);

    % Рівномірний розподіл часових вузлів від 0 до t_end
    t = linspace(0, t_end, j);

    % Цикл проходить по всіх сегментах траєкторії
    for segments = 1:(j-1)
        % Обчислення тривалості поточного сегмента
        durationOfCurrentSegment = t(segments+1) - t(segments);

        % Матриця формує систему рівнянь для знаходження коефіцієнтів полінома
        B = [1   0   0     0      0        0;
             0   1   0     0      0        0;
             0   0   2     0      0        0;
             1   durationOfCurrentSegment durationOfCurrentSegment^2
durationOfCurrentSegment^3   durationOfCurrentSegment^4     durationOfCurrentSegment^5;
             0   1  2*durationOfCurrentSegment 3*durationOfCurrentSegment^2
4*durationOfCurrentSegment^3   5*durationOfCurrentSegment^4;
             0   0   2    6*durationOfCurrentSegment  12*durationOfCurrentSegment^2
20*durationOfCurrentSegment^3];

        % Вектор граничних умов для поточного сегмента
        c = [x(segments); x_d(segments); x_dd(segments); x(segments+1);
x_d(segments+1); x_dd(segments+1)];

        % Розв’язання системи лінійних рівнянь Bx=c, B - матриця умов; x - шукані
коефіцієнти; c - вектор граничних умов
        coef(segments,:) = (B\c)';
    end
end
```

```
%% Підфункція дискретизації траєкторії та покрокової перевірки допустимості кутів
зчленувань

function [t, q_traj, qd_traj, qdd_traj, result] = ...
    compute_trajectory_with_validation(...
        coef1, coef2, coef3, x1, x2, x3, tf, dt, q_start, ...
```

```
    q1_min, q1_max, q2_min, q2_max, q3_min, q3_max)

%  Ініціалізація часового вектора та розмірів масивів
t        = 0:dt:tf;
n_points = length(t);
n_seg    = size(coef1, 1);

q_traj   = zeros(n_points, 3);
qd_traj  = zeros(n_points, 3);
qdd_traj = zeros(n_points, 3);

% Часові координати вузлових точок (рівномірно від 0 до tf)
t_wp = linspace(0, tf, length(x1));

% Ініціалізація першої точки стартовою позицією (усуває проблему: при
% i=1 немає "попередньої" точки)
q_traj(1, :) = q_start;

% Швидкість та прискорення на старті завжди = 0 (гранична умова)

%  Ініціалізація структури результату
result.success       = true;
result.skipped_steps = 0;
result.reason        = '';

% Перевірка, чи задані межі (якщо поля GUI порожні - NaN)
ranges_defined = ~isnan(q1_min) && ~isnan(q1_max) && ...
                 ~isnan(q2_min) && ~isnan(q2_max) && ...
                 ~isnan(q3_min) && ~isnan(q3_max);

%  Головний цикл обчислення траєкторії
for i = 1:n_points

    % Поточний момент часу для якого обчислюється траєкторія
    current_t = t(i);

    % Визначення сегмента, якому належить поточний момент часу
    seg = find(current_t >= t_wp(1:end-1) & ...
               current_t <= t_wp(2:end), 1);
    if isempty(seg)
        seg = n_seg;  % для останньої точки tf
    end

    % Локальний час всередині сегмента (відраховується від початку сегмента)
    t_local = current_t - t_wp(seg);

    % Обчислення положення, швидкості та прискорення для кожного суглоба
    for joint = 1:3
        switch joint
            case 1,  c = coef1(seg, :);
            case 2,  c = coef2(seg, :);
            otherwise, c = coef3(seg, :);
        end

        % Поліном 5-го порядку та його похідні
        q_traj(i, joint) = c(1) + c(2)*t_local   + c(3)*t_local^2 ...
                         + c(4)*t_local^3 + c(5)*t_local^4 + c(6)*t_local^5;

        qd_traj(i, joint) = c(2) + 2*c(3)*t_local   + 3*c(4)*t_local^2 ...
                          + 4*c(5)*t_local^3 + 5*c(6)*t_local^4;

        qdd_traj(i, joint) = 2*c(3) + 6*c(4)*t_local  + 12*c(5)*t_local^2 ...
```

```
                            + 20*c(6)*t_local^3;
        end

        %  Перевірка обмежень суглобів (якщо межі задані)
        if ranges_defined
            [valid, ~] = checkJointLimits(...
                q_traj(i,:), q1_min, q1_max, q2_min, q2_max, q3_min, q3_max);

            if ~valid
                % пропускаємо крок, залишаємо попередню допустиму позицію
                if i > 1
                    q_traj(i,:)   = q_traj(i-1,:);
                    qd_traj(i,:)  = zeros(1, 3);
                    qdd_traj(i,:) = zeros(1, 3);
                else
                    warning('Стартова позиція поза межами робочої області.');
                    break;
                end
                result.skipped_steps = result.skipped_steps + 1;
            end
        end
    end

    %  Аналіз результату після завершення циклу
    skip_ratio = result.skipped_steps / n_points;

    if result.skipped_steps > 0
        result.reason = sprintf(...
            '%d кроків із %d (%.0f%%) виходили за межі суглобів і були замінені
попередньою допустимою позицією.', ...
            result.skipped_steps, n_points, skip_ratio * 100);
    end

    % Якщо більше 30% кроків пропущено - вважаємо траєкторію незадовільною
    if skip_ratio > 0.3
        result.success = false;
    end
end
```