МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПрАТ «ПРИВАТНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ДО ЗАХИСТУ ДОПУЩЕНА

Зав.кафедрою _____
д.е.н.,доцент Левицький С.І.
«___» _____ 2023 р.

МАГІСТЕРСЬКА ДИПЛОМНА РОБОТА

РОЗРОБКА БІБЛІОТЕКИ ДЛЯ РОЗРАХУНКУ БАЛАНСУВАЛЬНИХ ХАРАКТЕРИСТИК КВАДРОКОПТЕРА

Виконав
ст. гр. КІ-212м _____ О.Ю. Курінной
(підпис)

Керівник
к.т.н. _____ О.А. Хараджян
(підпис)

Запоріжжя
2023

ПРАТ «ПВНЗ «ЗАПОРІЗЬКИЙ ІНСТИТУТ ЕКОНОМІКИ

ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ»

Кафедра Інформаційних технологій

ЗАТВЕРДЖУЮ
Зав. кафедрою

_____
д.е.н.,доцент Левицький С.І.
«___» _____ 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИПЛОМНУ РОБОТУ

Студенту гр. *КІ – 212м*, спеціальності *«Комп'ютерна інженерія»*

Курінной Олег Юрійович

1.Тема: *Розробка бібліотеки для розрахунку балансувальних характеристик квадрокоптера.*

затверджена наказом по інституту «___»_____ 2023 р. № ____

2. Термін здачі студентом закінченої роботи: «___»_____ 2023 р.

3. Перелік питань, що підлягають розробці:

1. Аналіз особливості квадрокоптерів.
2. Аналіз типових схем квадрокоптерів та систем управління.
3. Аналіз динамічних моделей квадрокоптерів.
4. Розробка алгоритму обчислення характеристик квадрокоптера.
5. Аналіз особливостей програмування у Matlab.
6. Розробка модуля обчислення дінаміки квадрокоптера.

7. Розробка модуля графічного інтерфейсу обчислення дінаміки квадрокоптера.
8. Тестування розробленої системи.

4. Календарний графік підготовки кваліфікаційної роботи

№ етапу	Зміст	Терміни виконання	Готовність по графіку %, підпис керівника	Підпис керівника про повну готовність етапу, дата
1	Збір практичного матеріалу за темою кваліфікаційної бакалаврської роботи	04.09.23-17.10.23		
2	І атестація І розділ кваліфікаційної бакалаврської роботи	23.10.23-28.10.23		
3	ІІ атестація ІІ розділ кваліфікаційної бакалаврської роботи	20.11.23-25.11.23		
4	ІІІ атестація ІІІ розділ кваліфікаційної бакалаврської роботи, висновки та рекомендації, додатки, реферат	18.12.23-23.12.23		
5	Перевірка кваліфікаційної бакалаврської роботи на оригінальність	18.12.23-23.12.23		
6	Доопрацювання кваліфікаційної бакалаврської роботи, підготовка презентації, отримання відгуку керівника і рецензії	25.12.23-06.01.24		
7	Попередній захист кваліфікаційної бакалаврської роботи	08.01.24-13.01.24		
8	Подача кваліфікаційної бакалаврської роботи на кафедру	за 3 дні до захисту		
9	Захист кваліфікаційної бакалаврської роботи	15.01.24-20.01.24		

Дата видачі завдання: 16.01.2023 р.

Керівник кваліфікаційної бакалаврської роботи _____ (підпис) О.А. Хараджян (прізвище та ініціали)

Завдання отримав до виконання _____ (підпис) О.Ю. Курінной (прізвище та ініціали)

РЕФЕРАТ

Дипломна робота містить 63 стор., 8 рис., 1 таблиць, 2 додаток, 6 використаних джерел.

Об’єкт роботи: аеродинамічні властивості квадрокоптеру.

Предмет роботи: балансувальні характеристики квадрокоптеру.

Мета роботи: розробка програми моделювання балансувальних характеристик квадрокоптеру.

Задачі роботи: аналіз особливості квадрокоптерів; аналіз типових схем квадрокоптерів та систем управління; аналіз динамічних моделей квадрокоптерів; розробка алгоритму обчислення характеристик квадрокоптера; аналіз особливостей програмування у Matlab; розробка модуля обчислення дінаміки квадрокоптера; розробка модуля графічного інтерфейсу обчислення дінаміки квадрокоптера; тестування розробленої системи.

На сьогоднішній день квадрокоптери із засобів для розваг перетворилися на літаючі апарати, які допомагають людині в різних видах діяльності. Квадрокоптер має слабку стійкість, оскільки він без системи керування є нестійким. Система управління квадрокоптером принципово має вирішувати завдання кутової та просторової стабілізації, та реалізацію траекторних задач.

У роботі розроблено модель динаміки та програму розрахунку балансувальних характеристик квадрокоптеру.

Розроблена програма дозволяє виконувати дослідження квадрокоптерів різних конфігурацій.

БАЛАНСУВАЛЬНІ ХАРАКТЕРИСТИКИ, ГРАФІЧНИЙ ІНТЕРФЕЙС, ДИНАМІЧНА МОДЕЛЬ, КВАДРОКОПТЕР, MATLAB

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Слово / словосполучення	Скорочення	Умови використання
Р		
Proportional – Integral – Derivative controller	PID	
О		
Об’єктно-орієнтоване програмування	ООП	

ВСТУП

На сьогоднішній день квадрокоптери із засобів для розваг перетворилися на літаючі апарати, які допомагають людині в різних видах діяльності. Квадрокоптер - це високоманеврений літальний апарат, який має слабку стійкість, оскільки він без системи керування є нестійким. Система управління квадрокоптером принципово має вирішувати завдання кутової та просторової стабілізації, виходу квадрокоптера на задану висоту (зліт), забезпечувати посадку, зависання та політ по заданій траєкторії. У загальному випадку до системи управління квадрокоптера пред'являються досить високі вимоги щодо точності та швидкодії з урахуванням заданих обмежень.

Відпрацювання законів управління здійснюється методами математичного, імітаційного та комп'ютерного моделювання. Для вирішення цього завдання необхідно розуміння принципів роботи квадрокоптера, механіки польоту та наявність математичної моделі, що описує його динаміку. Існує досить велика кількість близьких за структурою математичних моделей квадрокоптерів. Ряд математичних моделей лінеаризують або перетворюють для зручності розв'язання спеціалізованих завдань.

Керування квадрокоптером – принципово складна і цікава проблема. Квадрокоптери мають шість ступенів свободи (три поступальних і три обертальних) і чотири незалежні входи (швидкості ротора) керування. Щоб досягти шести ступенів свободи, керування обертальним і поступальним рухом поєднується. Отримана динамічна система дуже нелінійна, особливо після врахування складних аеродинамічних ефектів.

З розвитком технологій і зниженням цін на електронні пристрої стає все більш поширеним використання цих пристроїв як для любителів, так і для професіоналів. Крім цих пристроїв, група під назвою квадрокоптери (або квадротори) все частіше стає предметом досліджень і використовується для багатьох різних завдань. Це особливий тип багатогвинтових вертольотів із

чотирма гвинтами, які створюють підйомну силу за допомогою рухомих крил на відміну від літальних апаратів із нерухомим крилом.

Об’єкт роботи: аеродинамічні властивості квадрокоптеру.

Предмет роботи: балансувальні характеристики квадрокоптеру.

Мета роботи: розробка програми моделювання балансувальних характеристик квадрокоптеру.

Задачі роботи:

- аналіз особливості квадрокоптерів;
- аналіз типових схем квадрокоптерів та систем управління;
- аналіз динамічних моделей квадрокоптерів;
- розробка алгоритму обчислення характеристик квадрокоптера;
- аналіз особливостей програмування у Matlab;
- розробка модуля обчислення дінаміки квадрокоптера;
- розробка модуля графічного інтерфейсу обчислення дінаміки квадрокоптера;
- тестування розробленої системи.

РОЗДІЛ 1
КВАДРОКОТПЕР

1.1. Особливості квадрокоптерів

На сьогоднішній день квадрокоптери із засобів для розваг перетворилися на літаючі апарати, які допомагають людині в різних видах діяльності. Так, квадрокоптери активно використовуються для кадастрової аерофотозйомки, дозволяють здійснювати пошук людей та моніторинг пожеж у віддалених ділянках лісових зон, пошук та облік диких тварин, доставку вантажів на невеликі відстані, а також застосовуються для аматорської та професійної фото- та відеозйомки.

Квадрокоптер - це високоманеврений літальний апарат, який має слабку стійкість, оскільки він без системи керування є нестійким. Система управління квадрокоптером принципово має вирішувати завдання кутової та просторової стабілізації, виходу квадрокоптера на задану висоту (зліт), забезпечувати посадку, зависання та політ по заданій траєкторії. У загальному випадку до системи управління квадрокоптера пред'являються досить високі вимоги щодо точності та швидкодії з урахуванням заданих обмежень.

Відпрацювання законів управління здійснюється методами математичного, імітаційного та комп'ютерного моделювання. Для вирішення цього завдання необхідно розуміння принципів роботи квадрокоптера, механіки польоту та наявність математичної моделі, що описує його динаміку. Існує досить велика кількість близьких за структурою математичних моделей квадрокоптерів. Ряд математичних моделей лінеаризують або перетворюють для зручності розв'язання спеціалізованих завдань.

Гелікоптер — це літальний транспортний засіб, який використову гвинти, щоб створити силу тяги, яка утримує гелікоптер у повітрі. Звичайні вертольоти

мають два гвинти. Вони можуть бути організовані у вигляді двох компланарних роторів, які забезпечують тягу вгору, але обертаються в протилежних напрямках (щоб збалансувати крутні моменти, що діють на корпус вертольота). Два ротори також можуть бути скомпоновані з одним основним ротором, що забезпечує тягу, і меншим боковим ротором, орієнтованим перпендикулярно до осі несучого гвинта та протидіючи крутному моменту, створюваному головним ротором.

Однак ці конфігурації вимагають складної техніки для контролю напрямку руху; для зміни кута атаки гвинтів використовується автомат перекосу. Щоб створити крутний момент, кут атаки лопаті змінюється в залежності від її розташування на кожному оберті, таким чином, що більша тяга створюється з одного боку площини ротора, ніж з іншого. Складна конструкція ротора та автомату перекосу створює певні проблеми, збільшуючи вартість та ускладнюючи конструкцію.

Квадророторний гелікоптер (квадрокоптер) — вертоліт, який має чотири рівновіддалені гвинти, зазвичай розташовані по кутах квадратного корпусу. Завдяки чотирьом незалежним роторам потреба в автоматі перекосу відпадає. Автомат перекосу був необхідний, щоб дозволити вертольоту використовувати більше ступенів свободи, але той самий рівень контролю можна отримати, додавши ще два гвинти.

Розробка квадрокоптерів виявилась складною і неможливою без електронних систем. Зменшення вартості сучасних мікропроцесорів зробило використання квадрокоптерів можливим для комерційних, військових і навіть аматорських цілей.

Керування квадрокоптером – принципово складна і цікава проблема. Квадрокоптери мають шість ступенів свободи (три поступальних і три обертальних) і чотири незалежні входи (швидкості ротора) керування. Щоб досягти шести ступенів свободи, керування обертальним і поступальним рухом поєднується. Отримана динамічна система дуже нелінійна, особливо після врахування складних аеродинамічних ефектів.

З розвитком технологій і зниженням цін на електронні пристрої стає все більш поширеним використання цих пристроїв як для любителів, так і для професіоналів. Крім цих пристроїв, група під назвою квадрокоптери (або квадротори) все частіше стає предметом досліджень і використовується для багатьох різних завдань. Це особливий тип багатогвинтових вертольотів із чотирма гвинтами, які створюють підйомну силу за допомогою рухомих крил на відміну від літальних апаратів із нерухомим крилом.

Загалом любителі використовують квадрокоптери для дозвілля (наприклад, для авіамоделювання) або для простих експериментів, як-от перегляд від першої особи під час польоту через передачу відео в реальному часі.

Що стосується дослідників і галузей промисловості, квадрокоптери можуть використовуватися в багатьох різних видах завдань, як-от автономна доставка продуктів або досягнення складних місць для виконання певних завдань (наприклад, фотографування зовнішніх частин високої будівлі для її огляду).

Щоб мати можливість використовувати такий пристрій, ми повинні мати систему, здатну підтримувати як стабільність польоту, так і обмін даними між пристроєм і його хостом, який відповідає за передачу пристрою вказівок польоту або отримання даних про політ.

Для більшості любителів це зводиться до квадрокоптера з кількома датчиками та радіокеруванням, який виконує роль хоста, який безпосередньо керує літальним пристроєм у режимі реального часу.

Зазвичай дані не витягуються та не використовуються в більш складній системі керування за межами квадрокоптера. Ці дані використовуються лише внутрішньо у вбудованій системі в простій системі керування, посилання якої надається радіостанцією кожного моменту. Іншими словами, вбудована система керування вже запрограмована в пристрої, і основна функція любителів полягає в тому, щоб керувати радіостиками, щоб надавати еталонні сигнали квадрокоптеру для відстеження в режимі реального часу. Тому користувач

повинен знати, як керувати квадрокоптером, щоб запобігти його падінню на землю.

Для більш просунутих аматорів або дослідників це завдання стає складнішим. Майже у всіх випадках цікаво отримати дані датчиків. Типи датчиків можуть відрізнятися залежно від мети, але зазвичай дуже корисно отримати деякі дані, які можна використовувати як для обробки в реальному часі, так і для цілей аналізу.

Зростає кількість досліджень у цій галузі, спрямованих на пошук кращих методів керування всією системою. Більшість завдань можна вирішити за допомогою простих систем або методів, які легко знайти в літературі, але все більше досліджень у цій галузі все ще проводяться з метою вдосконалення існуючих методів, залишаючи систему більш точною або навіть більш автономною. Значна частина цих досліджень зосереджена на проблемах керування та стабільності квадрокоптера під час його польоту.

Деякі з них виконають певний набір тестів, щоб можна було отримати достатньо даних, щоб побудувати математичну модель пристрою, щоб зробити можливим проектування та моделювання його системи керування перед безпосереднім використанням. Зазвичай це вимагає певної обережності під час виконання тестів і додаткової роботи з їх підготовки. Однак, якщо провести достатню кількість тестів і створити добре наближену модель, можна побудувати надійну й оптимальну систему керування.

Проте слід зазначити, що спроектована система керування в цьому випадку буде специфічною для цього пристрою, коли теоретичну модель буде витягнуто з набору випробувань із пристроєм. Таким чином, зміна фізичних властивостей квадрокоптера може вимагати повної переробки цієї системи.

Інший підхід полягає у виконанні деяких тестів і вилученні даних датчиків, як і раніше, але замість пошуку математичної моделі фізичного пристрою можна спроектувати контролер безпосередньо з витягнутих даних. Таким чином, не потрібно було б витрачати час на пошук теоретичної моделі квадрокоптера. Цей тип алгоритму контролера називається керуванням на

основі даних і є альтернативою згаданому вище методу на основі моделі. Важливо відзначити, що оскільки нам більше не потрібна математична модель пристрою, набагато простіше створити алгоритм, здатний знаходити контролер на основі витягнутих даних у реальному часі. Таким чином, такий алгоритм може бути реалізований в різних типах квадрокоптера з різними фізичними властивостями, оскільки нам не потрібна його математична модель, як раніше.

Нарешті, ще одним широко використовуваним методом є використання отриманих даних у режимі реального часу, щоб система постійно навчалася та вдосконалювала свої контрольні параметри. Це загальний підхід, який може включати як рішення на основі моделі, так і рішення на основі даних, як правило, в ітераційному процесі. Варто зазначити, що незалежно від обраного методу можна вибрати проектування контролера з підходом штучного інтелекту (наприклад, штучні нейронні мережі) або з підходом класичної теорії управління (наприклад, PID).

Розглянемо квадрокоптер як об'єкт управління, механіку його польоту та принцип управління, на основі нелінійної математичної моделі у вигляді системи диференціальних рівнянь з урахуванням динаміки двигунів.

1.2. Типові схеми квадрокоптерів

Квадрокоптер — це літальний апарат із фіксованим кутом нахилу і чотирма гвинтами, як показано на малюнку 1. Такий транспортний засіб, як квадрокоптер, не є легким завданням через його складну конструкцію. Метою є розробка моделі транспортний засіб якомога реалістичніше. Типовий квадрокоптер має чотири ротори з фіксованими кутами, тому квадрокоптер має чотири вхідні сили, які в основному є тяга, що забезпечується кожним гвинтом, як показано на рис.1.1. Існує дві можливі конфігурації для більшість дизайнів квадрокоптерів «+» і «×». Квадрокоптер X-конфігурації вважається більш стабільним у порівнянні конфігурація до +, яка є більш акробатичною

конфігурацією. Пропелери 1 і 3 обертаються проти годинникової стрілки (CW), 2 і 4 обертається проти годинникової стрілки (CCW). Щоб квадрокоптер міг підтримувати рух вперед (назад). шляхом збільшення (зменшення) швидкості обертання передніх (задніх) роторів при зменшенні (збільшенні) швидкості заднього (переднього) ротора одночасно, що означає зміну кута нахилу. Цей процес необхідний для компенсації дії/реакції ефект (третій закон Ньютона). Гвинти 1 і 3 мають протилежний крок відносно 2 і 4, тому всі тяги мають той же напрямок.

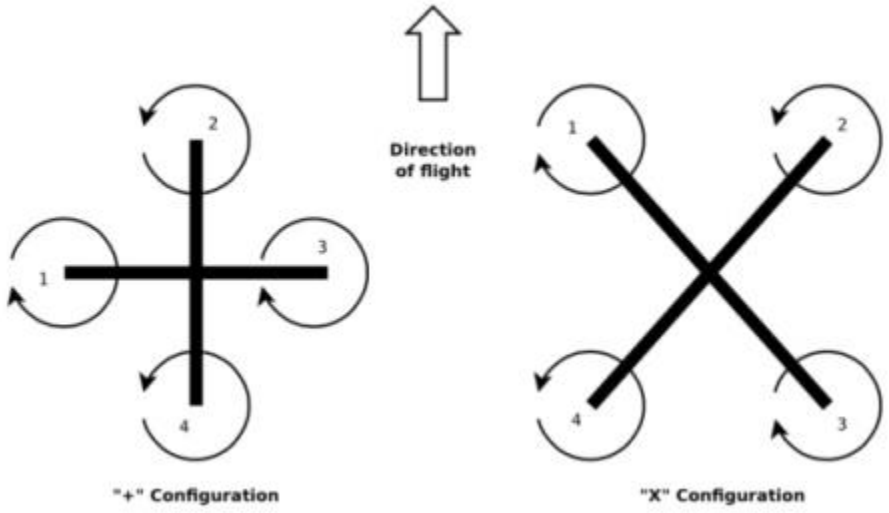

Рис. 1.1 Типи конфігурацій квадрокоптерів

РОЗДІЛ 2
РОЗРОБКА АЛГОРИТМУ ОБЧИСЛЕННЯ ХАРАКТЕРИСТИК КВАДРОКОПТЕРА

2.1. Динамічна модель квадрокоптера

Багатороторні літальні апарати можуть мати різні конструктивні схеми. Квадрокоптер як окремий випадок багатороторних літальних апаратів може будуватися за ікс- і плюс- подібної схеми (рис. 2.1). Відмінною рисою даної схеми і те, що осі OX і OZ перетинають центри двигунів.

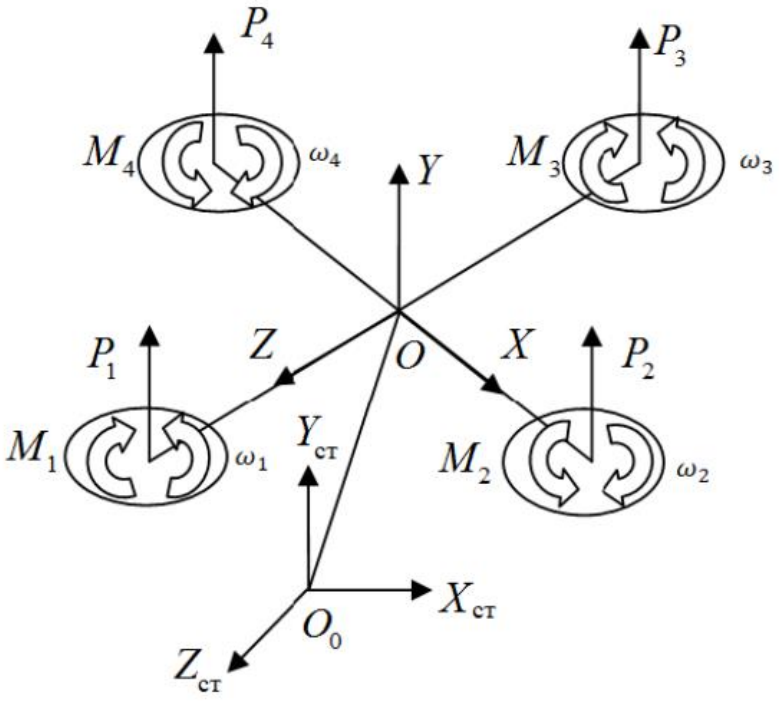

Рис. 2.1. Кінематична схема квадрокоптера

На кінематичній схемі осі OXYZ утворюють пов'язану систему координат яка жорстко пов'язана з квадрокоптером, осі O X Y Z ст ст формують стартову систему координат, в якій знаходиться спостерігач; P1, P2, P3, P4 підйомні

сили, що виникають за рахунок обертання відповідних гвинтів з швидкостями ω1, ω2, ω3, ω4; M1, M2, M3, M4 - реактивні моменти. Орієнтація квадрокоптера у просторі визначається трьома кутами:

кутом рисканія ψ - кутом між віссю OZ і площиною O Y Z 0 ст ст,

кутом тангажу ϑ - кутом між віссю OX і площиною O X Z 0 ст ст,

і кутом крену ϕ – кутом між віссю OX та площиною O X Y 0 ст ст.

Керування рухом квадрокоптера відбувається шляхом подачі керуючих напруг на двигуни, які приводять до обертання гвинтів, що призводить до виникнення кутових швидкостей ω1, ω2, ω3, ω4, внаслідок чого формуються підйомні сили P1, P2, P3, P4 і, в результаті цього виникає сила та відповідні моменти, які виникають унаслідок реактивної реакції, породженої обертанням гвинтів.

Пересування у вертикальній площині здійснюється за допомогою проекції загального вектора підйомної сили P на вертикальну вісь, а зміна орієнтації відбувається через різницю між значеннями кутових швидкостей ω1, ω2, ω3, ω4, причому для повороту в позитивному напрямку необхідно, щоб кутові швидкості ω2 та ω4 були більше, ніж ω1 і ω3, а для повороту в негативному напрямку навпаки. Іншими словами, поворот здійснюється за рахунок різниці реактивних моментів, які виникають при обертанні двигунів.

Зміна кутової орієнтації виконується зміною різниць пар підйомних сил: для кута тангажу P2, P4 і для кута крену P1, P3.

Перетворення систем координат. Співвідношення між стартовою та пов'язаною системами координат можна записати у вигляді матриці напрямних косінусів:

$$\mathbf{R} = \begin{pmatrix} c(\psi)c(\vartheta) & s(\vartheta) & -s(\psi)c(\vartheta) \\ s(\psi)s(\varphi) - c(\psi)s(\vartheta)c(\varphi) & c(\vartheta)c(\varphi) & s(\vartheta)s(\psi)c(\varphi) + c(\psi)s(\varphi) \\ c(\varphi)s(\psi) + c(\psi)s(\vartheta)s(\varphi) & -c(\vartheta)s(\varphi) & c(\varphi)c(\psi) - s(\varphi)s(\vartheta)s(\psi) \end{pmatrix} \quad (2.1)$$

Відповідно, перерахунок деякого вектора $\mathbf{r} = [x, y, z]^{\mathrm{T}}$ значень з стартової у зв'язану систему координат здійснюється наступним чином:

$$\mathbf{r}_{\text{св}} = \mathbf{R}\mathbf{r}_{\text{ст}}, \tag{2.2}$$

а для зворотного переходу з урахуванням ортогональності двох систем координат:

$$\mathbf{r}_{\text{ст}} = \mathbf{R}^{\mathrm{T}}\mathbf{r}_{\text{св}}, \tag{2.3}$$

де матриця направляючих косінусів для зворотного переходу має такий вигляд:

$$\mathbf{R}^{\mathrm{T}} = \begin{pmatrix} c(\psi)c(\vartheta) & s(\psi)s(\varphi) - c(\psi)s(\vartheta)c(\varphi) & c(\varphi)s(\psi) + c(\psi)s(\vartheta)s(\varphi) \\ s(\vartheta) & c(\vartheta)c(\varphi) & -c(\vartheta)s(\varphi) \\ -s(\psi)c(\vartheta) & s(\vartheta)s(\psi)c(\varphi) + c(\psi)s(\varphi) & c(\varphi)c(\psi) - s(\varphi)s(\vartheta)s(\psi) \end{pmatrix}. \tag{2.4}$$

Математична модель динаміки квадрокоптера. У випадку рух квадрокоптера як твердого тіла складається з руху центру мас і обертального руху щодо центру мас. Для отримання рівнянь руху центру мас квадрокоптера запишемо вирази для сили тяги таким чином:

$$\mathbf{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} 0 \\ P \\ 0 \end{bmatrix};$$
$$P = P_1 + P_2 + P_3 + P_4 = \sum_{i=1}^{4} c_{pi}\omega_i^2; \quad (2.5)$$
$$c_{pi} = \frac{1}{2}\rho C_Y S_i r_i^2,$$

де P - вектор сили тяги у зв'язковій системі координат;

P - сумарна тяга - одержувана в результаті дії чотирьох двигунів у пов'язаній системі координат;

cpi - коефіцієнт сили тяги;

ρ — густина повітря (1,225 кг/м3), яка відповідає густині сухого повітря при 15 °C і тиску 101330 Па;

C_Y - коефіцієнт підйомної сили - визначається емпірично, для квадрокоптерів зазвичай знаходиться в межах від 0,2 до 1,3;

Si - площа, кола, яке описує крутний i-й гвинт радіусом ri .

Проекція сили тяги у стартовій системі координат визначається так:

$$\mathbf{P}_{ст} = \mathbf{R}^{\mathrm{T}}\mathbf{P} = \begin{bmatrix} P\left(s(\psi)s(\varphi) - c(\psi)s(\vartheta)c(\varphi)\right) \\ P\left(c(\vartheta)c(\varphi)\right) \\ P\left(s(\vartheta)s(\psi)c(\varphi) + c(\psi)s(\varphi)\right) \end{bmatrix}. \quad (2.6)$$

Сила опору повітря f і сила тяжкості G визначаються наступним чином:

$$\mathbf{f} = \begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix}; \ \mathbf{G} = \begin{bmatrix} 0 \\ -mg \\ 0 \end{bmatrix}, \tag{2.7}$$

а кожна проекція сили опору повітря на осі пов'язаної системи координат визначається, виходячи з наступного співвідношення:

$$f_i = C_{x0} \frac{\rho V_i^2}{2} S, \tag{2.8}$$

де Vi - Проекція лінійної швидкості на відповідну вісь;

S - характерна площа поверхні квадрокоптера;

Cx0 - безрозмірний коефіцієнт лобового (аеродинамічного) опору, що враховує обтічність, який знаходиться у діапазоні від 0,2 до 1,2.

Максимальна швидкість руху квадрокоптера становить 263 км/год, або 73 м/с, проте для серійних моделей максимальне значення швидкості не перевищує 40 м/с, отже, максимальну силу опору повітря при характерній площі 0,01 м2 можна визначити як

$$f_{\max} = 0,4 \frac{1,225 \cdot 40^2}{2} 0,01 = 3,92 \text{ Н}. \tag{2.9}$$

З огляду на те, що максимальна сила опору повітря на порядок менше величини підйомної сили, а також більшу обчислювальну та просторову складність комп'ютерного моделювання атмосферної обстановки та вітрових обурень для дослідження динаміки квадрокоптера можна прийняти fi = 0.

Однак виключати fi з наступних рівнянь не будемо для збереження адекватності математичної моделі, що отримується.

Таким чином, рівняння руху центру мас у векторному вигляді можна записати так:

$$m\ddot{\mathbf{r}} = \mathbf{R}^{\mathrm{T}}\mathbf{P}_{\mathrm{ст}} - \mathbf{f} - \mathbf{G}. \tag{2.10}$$

У поелементному вигляді отримаємо наступну систему із трьох диференціальних рівнянь:

$$\begin{cases} \ddot{x} = \dfrac{P\left(s(\psi)s(\varphi) - c(\psi)s(\vartheta)c(\varphi)\right) - f_x}{m}; \\ \ddot{y} = \dfrac{P\left(c(\vartheta)c(\varphi)\right) - f_y - mg}{m}; \\ \ddot{z} = \dfrac{P\left(s(\vartheta)s(\psi)c(\varphi) + c(\psi)s(\varphi)\right) - f_z}{m}. \end{cases} \tag{2.11}$$

Кутовий рух квадрокоптера у зв'язаній системі координат описується системою з трьох рівнянь - динамічних рівнянь Ейлера:

$$\begin{cases} I_x\dot{\omega}_x + (I_z - I_y)\omega_y\omega_z = M_x; \\ I_y\dot{\omega}_y + (I_x - I_z)\omega_z\omega_x = M_y; \\ I_z\dot{\omega}_z + (I_y - I_x)\omega_y\omega_x = M_z, \end{cases} \tag{2.12}$$

де Ix , I y , Iz - моменти інерції щодо відповідних осей OX,

x ωy , ωz - проекції кутової швидкості; Mx , M y , Mz - проекції сумарного моменту, що діє на квадрокоптер, які можна записати в наступному вигляді:

$$\begin{aligned} M_x &= M_{qx} + M_{mx} + M_{px}; \\ M_y &= M_{qy}; \\ M_z &= M_{qz} + M_{mz} + M_{pz}, \end{aligned} \tag{2.13}$$

M - моменти, що створюються різницею сил тяги, що формуються відповідними парами гвинтів, за допомогою яких відбувається керуванням кутами крену, нишпорення та тангажу; Mmi та M pi - гіроскопічні моменти двигунів та гвинтів. Всі ці моменти можуть бути обчислені таким чином:

$$\begin{aligned} M_{qx} &= (P_3 - P_1)l; \\ M_{qy} &= M_2 + M_4 - M_1 - M_3; \\ M_{qz} &= (P_2 - P_4)l; \\ M_i &= m_{pi}\omega_i^2; \\ m_{pi} &= \frac{1}{2}\rho C_Y S_i r_i^3; \\ M_{mx} &= I_m \omega_z (\omega_1 + \omega_3 - \omega_2 - \omega_4); \\ M_{mz} &= I_m \omega_x (\omega_2 + \omega_4 - \omega_1 - \omega_3); \\ M_{px} &= I_p \omega_z (\omega_1 + \omega_3 - \omega_2 - \omega_4); \\ M_{pz} &= I_p \omega_x (\omega_2 + \omega_4 - \omega_1 - \omega_3), \end{aligned} \tag{2.14}$$

mpi - коефіцієнт моменту; l - відстань від осі обертання валу двигуна до центру мас квадрокоптера; Im - осьовий момент інерції валу ротора двигуна;

Ip - осьовий момент інерції гвинта. Величини Imz і Ipz є кінетичними моментами двигунів і гвинтів відповідно. Якщо розкрити дужки в рівняннях гіроскопічних моментів Mmx, Mmz, Mpx, Mpz , то стає видно, що сумарні гіроскопічні моменти є сумою гіроскопічних моментів від валів роторів двигунів Mmx, Mmz і обертових гвинтів Mpx Mpz.

Незважаючи на те, що динамічні рівняння Ейлера описують динаміку обертального руху квадрокоптера, їх недостатньо для визначення кутів тангажу, крену та нишпорення, які визначаються щодо стартової чи деякої навігаційної системи координат. Динаміка кутів описується системою з трьох диференціальних рівнянь - кінематичних рівнянь Ейлера, що мають такий вигляд:

$$\begin{cases} \dot{\varphi} = \omega_x - tg(\vartheta)(\omega_y c(\varphi) - \omega_z s(\varphi)); \\ \dot{\psi} = \dfrac{\omega_y c(\varphi) - \omega_z s(\varphi)}{c(\vartheta)}; \\ \dot{\vartheta} = \omega_z s(\varphi) + \omega_y s(\varphi). \end{cases} \tag{2.15}$$

Таким чином, повна математична модель, що описує просторову та кутову динаміку квадрокоптера, складається з дев'яти диференціальних рівнянь:

$$
\begin{cases}
\ddot{x} = \dfrac{P\big(s(\psi)s(\varphi) - c(\psi)s(\vartheta)c(\varphi)\big) - f_x}{m}; \\
\ddot{y} = \dfrac{P\big(c(\vartheta)c(\varphi)\big) - f_y - mg}{m}; \\
\ddot{z} = \dfrac{P\big(s(\vartheta)s(\psi)c(\varphi) + c(\psi)s(\varphi)\big) - f_z}{m}; \\
\dot{\omega}_x = \dfrac{(I_y - I_z)\omega_y\omega_z + M_x}{I_x}; \\
\dot{\omega}_y = \dfrac{(I_z - I_x)\omega_x\omega_z + M_y}{I_y}; \\
\dot{\omega}_z = \dfrac{(I_x - I_y)\omega_y\omega_x + M_z}{I_z}; \\
\dot{\varphi} = \omega_x - tg(\vartheta)(\omega_y c(\varphi) - \omega_z s(\varphi)); \\
\dot{\psi} = \dfrac{\omega_y c(\varphi) - \omega_z s(\varphi)}{c(\vartheta)}; \\
\dot{\vartheta} = \omega_z c(\varphi) + \omega_y s(\varphi).
\end{cases}
\tag{2.16}
$$

Математична модель двигунів. Управління швидкістю обертання гвинтів квадрокоптера здійснюється зміною рівня напруг, що управляють (або шпаруватістю, якщо керуючий сигнал широтно-імпульсний). Як правило, двигуни квадрокоптера - це двигуни постійного струму, що мають свою динаміку. Тому в математичній моделі квадрокоптера необхідно враховувати також динаміку двигунів.

Найчастіше у квадрокоптерах застосовують безколекторні двигуни, якір яких є набір неодимових магнітів, а статор складається з обмотки збудження, на який подається керуюча напруга. На відміну від колекторних двигунів, де якір обертається всередині статора, у безколекторних якір обертається навколо статора, що знаходиться усередині. У такому разі необхідно описати динаміку ланцюга статора, а неякоря.

Електрична складова двигуна описується наступним диференціальним рівнянням:

$$U = RI + L\frac{dI}{dt} + E_{\text{пр}}. \tag{2.17}$$

Тут U - керуюча напруга, що надходить на вхід двигуна; R - активний опір обмотки статора; L - індуктивна складова обмотки статора; Епр - протиЕРС, що наводиться магнітним полем Ф якоря в обмотці статора, яке розраховується наступним чином:

$$E_{\text{пр}} = 30C_e\Phi, \tag{2.18}$$

де Се - постійна електрична двигуна.

Для двигунів малої та середньої потужності та ндуктивна складова $E_{\text{пр}}$ на кілька порядків менше за решту складових у рівнянні (3), а отже, їй можна знехтувати. Механічна складова описується наступним диференціальним рівнянням:

$$C_{\text{м}}\Phi I = J\frac{d\omega}{dt} + \mu\omega \tag{2.19}$$

де См - Електромеханічна постійна;

J - сумарний момент інерції ротора двигуна та гвинта;

μ - коефіцієнт тертя підшипника ротора;

I - струм у обмотці статора;

ω - кутова швидкість обертання гвинта.

Поєднуючи рівняння (2.18) і (2.19) для електричної та механічної частин двигуна, можна записати математичну модель кожного двигуна квадрокоптера у вигляді системи із двох диференціальних рівнянь першого порядку:

$$\begin{cases} \dfrac{dI}{dt} = \dfrac{U - RI}{L}; \\ \dfrac{d\omega}{dt} = \dfrac{C_м \Phi I - \mu\omega}{J}. \end{cases} \tag{2.20}$$

Величини R, C , Ф, L, μ вважатимемо рівними для кожного двигуна черездосить високої якості технологічних процесів виробництва, а також враховуючи те, що навіть вплив невеликих девіацій цих параметрів на динаміку системи можуть бути легко компенсовані системою управління. Для всіх чотирьох двигунів можна записати таку систему диференціальних рівнянь:

$$\begin{cases} \dfrac{dI_1}{dt} = \dfrac{U_1 - RI_1}{L}; & \dfrac{d\omega_1}{dt} = \dfrac{C_м \Phi I_1 - \mu\omega_1}{J}; \\ \dfrac{dI_2}{dt} = \dfrac{U_2 - RI_2}{L}; & \dfrac{d\omega_2}{dt} = \dfrac{C_м \Phi I_2 - \mu\omega_2}{J}; \\ \dfrac{dI_3}{dt} = \dfrac{U_3 - RI_3}{L}; & \dfrac{d\omega_3}{dt} = \dfrac{C_м \Phi I_3 - \mu\omega_3}{J}; \\ \dfrac{dI_4}{dt} = \dfrac{U_4 - RI_4}{L}; & \dfrac{d\omega_4}{dt} = \dfrac{C_м \Phi I_4 - \mu\omega_4}{J}; \end{cases} \tag{2.21}$$

Повна математична модель квадрокоптера у вигляді Коші. На дослідження динаміки квадрокоптера методами чисельного моделювання, тобто чисельного вирішення завдання Коші, опис система має бути наведено до нормальній формі Коші, тобто до системи диференціальних рівнянь першого порядку. Для цього для систем рівнянь (2.16) та (2.21) сформуємо наступні вектори (вектори стану) для заміни змінних:

$$\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \\ \omega_x \\ \omega_y \\ \omega_z \\ \varphi \\ \psi \\ \vartheta \end{bmatrix}; \quad \mathbf{Y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \end{bmatrix} = \begin{bmatrix} I_1 \\ \omega_1 \\ I_2 \\ \omega_2 \\ I_3 \\ \omega_3 \\ I_4 \\ \omega_4 \end{bmatrix}. \tag{2.22}$$

Тоді повна математична модель квадрокоптера, що описує динаміку самого квадрокоптера та динаміку двигунів з урахуванням заміни змінних із рівняння (2.22), у нормальній формі Коші матиме такий вигляд:

$$\begin{bmatrix}\dot{\mathbf{X}}\\ \dot{\mathbf{Y}}\end{bmatrix}=\begin{cases}\dot{x}_1 = x_2;\\ \dot{x}_2 = \dfrac{P\left(s(x_{11})s(x_{10}) - c(x_{11})s(x_{12})c(x_{10})\right) - f_x}{m};\\ \dot{x}_3 = x_4;\\ \dot{x}_4 = \dfrac{P\left(c(x_{12})c(x_{10})\right) - f_y - mg}{m};\\ \dot{x}_5 = x_6;\\ \dot{x}_6 = \dfrac{P\left(s(x_{12})s(x_{11})c(x_{10}) + c(x_{11})s(x_{10})\right) - f_z}{m};\\ \dot{x}_7 = \dfrac{(I_y - I_z)x_9x_8 + M_x}{I_x};\\ \dot{x}_8 = \dfrac{(I_z - I_x)x_7x_9 + M_y}{I_y};\\ \dot{x}_9 = \dfrac{(I_x - I_y)x_8x_7 + M_z}{I_z};\\ \dot{x}_{10} = x_7 - tg(x_{12})(x_8c(x_{10}) - x_9s(x_{10}));\\ \dot{x}_{11} = \dfrac{x_8c(x_{10}) - x_9s(x_{10})}{c(x_{12})};\\ \dot{x}_{12} = x_9c(x_{10}) + x_8s(x_{10});\\ \dot{y}_1 = \dfrac{U_1 - Ry_1}{L};\\ \dot{y}_2 = \dfrac{C_м\Phi y_1 - \rho y_2}{J};\\ \dot{y}_3 = \dfrac{U_2 - Ry_3}{L};\\ \dot{y}_4 = \dfrac{C_м\Phi y_3 - \rho y_4}{J};\\ \dot{y}_5 = \dfrac{U_3 - Ry_5}{L};\\ \dot{y}_6 = \dfrac{C_м\Phi y_5 - \rho y_6}{J};\\ \dot{y}_7 = \dfrac{U_4 - Ry_7}{L};\\ \dot{y}_8 = \dfrac{C_м\Phi y_7 - \rho y_8}{J},\end{cases} \tag{2.23}$$

де величина P визначається на підставі рівняння (1), яке з урахуванням прийнятої заміни змінних набуде вигляду

$$P = P_1 + P_2 + P_3 + P_4 = c_p\omega_i^2 + c_p\omega_i^2 + c_p\omega_i^2 + c_p\omega_i^2 = c_p y_2^2 + c_p y_4^2 + c_p y_6^2 + c_p y_8^2. \tag{2.24}$$

2.2. Особливості програмування у Matlab

MATLAB — це високопродуктивна мова для технічних обчислень. Система інтегрує обчислення, візуалізацію та програмування в простому у

використанні середовище, де завдання та розв’язки виражені в звичайній математичній нотації. Типове використання включає

- математика та обчислення
- розробка алгоритму
- збір даних
- моделювання, симуляція та прототипування
- аналіз даних, дослідження та візуалізація
- наукова та інженерна графіка
- розробка додатків, включаючи побудову графічного інтерфейсу користувача

MATLAB — це інтерактивна система, основним елементом даних якої є масив, який не потребує вказування розмірів. Це дозволяє вирішити багато технічних обчислювальних проблем, особливо з матричними та векторними виразами при інтеграції з скалярними неінтерактивними програмами на мовах, наприклад, C або Fortran.

Назва MATLAB розшифровується як матрична лабораторія. Спочатку був MATLAB написаний для забезпечення легкого доступу до матричного програмного забезпечення, розробленого LINPACK та проекти EISPACK. Сьогодні двигуни MATLAB включають LAPACK і бібліотеки BLAS, впроваджуючи сучасне програмне забезпечення для матриці обчислення.

MATLAB розвивався протягом багатьох років за допомогою багатьох користувачів. В університетському середовищі, це стандартний навчальний інструмент для вирішення задач з математики, інженерії та природничих наук. У промисловості MATLAB є інструментом вибору для високопродуктивних досліджень, розробки та аналізу.

MATLAB містить сімейство додаткових інструментів. Пакети інструментів дозволяють застосовувати спеціальні технології. Набори інструментів комплексні колекції функцій MATLAB (M-файли), які розширюють MATLAB середовища для вирішення окремих класів проблем.

Області, в яких доступні блоки обробки сигналів, системи керування, нейронні мережі, нечітка логіка, вейвлети, моделювання та багато іншого.

Matlab (скорочення від matrix laboratory) — це спеціалізоване числове обчислювальне середовище та мова програмування. Він має вбудовану підтримку для роботи з матрицями, комплексними числами та візуалізації даних. Matlab широко використовується вченими та інженерами в промисловості та академічних колах. Працює на більшості популярних операційних систем. Можна використовувати як потужний калькулятор і інструмент візуалізації даних. Що ще важливіше, ви можете програмувати в Matlab. Добре підходить для швидкого прототипування та «програмування в малому». Багато сучасних функцій програмування використано (масиви клітинок, структури, об’єкти, вкладені функції, анонімні функції, покажчики на функції, перевантаження операторів), але езотеричні та суперечливі позначення. У результаті багато програмістів Matlab ніколи їх не використовують.

Доступність. Це власне комерційне програмне забезпечення, яке розповсюджується The MathWorks. Багато університетів мають ліцензію на сайт або лабораторії, де він встановлений.

Matlab не компілюється явно. Інтерактивний інтерпретатор: введіть команди у вікні команд. Є можливість зберегти послідовність команд у файлі Script-M.

Вбудовані типи даних. «Все є масив». Matlab має 15 основних типів: single, double, logical, char, cell, int8,uint8, int16, uint16, int32, uint32, int64, uint64structure та дескриптор функції. Усі вони мають форму масиву (від мінімального розміру 0 на 0 до d-вимірного масиву будь-якого розміру). Також може отримати доступ до визначених користувачем типів: класів користувачів і класів Java. Типом даних за замовчуванням є 2D array (or matrix) of doubles. Рядки є масивом символів (хоча для створення масиву рядків різного розміру використовуйте масив клітинок).

У Matlab комплексні числа є double числами з дійсною та уявною частинами. (Matlab зберігає 2 паралельних подвійних вектора, один для дійсної частини, а інший для уявної частини.) Досить корисно виконувати set i = sqrt(-1) перед використанням оскільки могло змінитися значення і раніше.

Числа з плаваючою комою використовують формат IEEE 754; цілі числа є доповненням до двох.

Скалярні оператори. +, -, *, / (ділення з плаваючою комою), ^ (для піднесення до степеня).

Оператори порівняння. < <= > >= == ~= (замість !=). Повертає логічне значення, яке має значення 1 для істинного та 0 для хибного.

Логічні оператори. &&, ||, ~ (замість !)

Математична бібліотека. sqrt(), rand(), pi, abs() Далі показано, як виконувати основні арифметичні обчислення.

```
% matlab
format compact
>> besselk(1, -3)
ans =  -0.0402 -12.4199i
>> erf(0.9)
ans =   0.7969
```

рядки. Рядковий літерал: 'привіт' замість "привіт". Конкатенація рядків:

```
a = ['hello' 'world']
```

Порядок пріоритету. Порядок пріоритету, дужки, асоціативність.

Заяви. Твердження закінчуються крапкою з комою; якщо ви опустите крапку з комою, ви отримаєте ехо-відповідь.

Змінні. Чутливий до регістру.

```
radius = 25.3;
area = pi * radius^2;
```

Динамічно введений. Java строго типізована та статично типізована. Кожен вираз і змінна мають відомий (і перевірений) тип і під час компіляції. Matlab динамічно типізований. Змінні оголошені без визначення їх типів. Тип визначається під час виконання. Здається простішим для програміста, але значно більш схильним до помилок. Коли Matlab стикається з новою змінною, вона автоматично виділяє потрібний обсяг пам’яті; коли змінна змінюється, вона за потреби перерозподіляє пам’ять. Слабко типізований (тип змінної не застосовується, наприклад, можна об’єднати рядок «12» і ціле число 3, щоб отримати «123», а потім розглядати його як ціле число, все без будь-яких явних перетворень). Matlab дуже слабо типізований.

```
a = 5;          % 1-by-1 double
a = [2 3 4];    % 1-by-3 double
a = 10 < 17;    % 1-by-1 logical
a = a + a;      % 1-by-1 double
class(a)
```

Недолік: помилки не виявляються до моменту виконання. Наприклад, константа і попередньо визначена як квадратний корінь з -1. Однак ви можете перевизначити його як ціле число або масив. Або, якщо назвати змінну так само, як і функцію, це приховає функцію, унеможливлюючи її виклик.

```
min = 17;
min(rand(100, 1));  % error
class(a)
% awkward type checking is leap year
if (round(year) ~= year))
    error('Year must be an integer.')
end
```

Умови та цикли. Умови: якщо оператори схожі на Java.

```
if (x < 0)
   x = -x;
end

if (x > y)
   max = x;
else
   max = y;
end
if     (income <      0) rate = .00;
elseif (income <  47450) rate = .25;
elseif (income < 174700) rate = .28;
elseif (income < 311950) rate = .33;
else                     rate = .35;
end
```

Петлі. Хоча цикл аналогічний; цикл for менш виразний (може лише перебирати змінну по елементах масиву).

```
% print largest power of two less than or equal to N
v = 1;
while (v <= N/2)
   v = 2 * v
end
v

% compute a finite product N! = 1 * 2 * ... * N
product = 1;
for i = 1:N
   product = product * i;
end
```

```
product

% or more simply product = prod(1:N)
```

Масиви. Вектори і матриці.

Створення векторів і матриць. (пробіл, двокрапка, одиниці, нулі, ранд).

```
A = [ 1 2 3; 4 5 6 ];      % 2-by-3 matrix
B = [ 1 2; 3 4; 5 6 ];     % 3-by-2 matrix
x = [ 1 2 3 ];             % row vector
y = [ 1; 2; 3 ];           % column vector

A = rand(3, 4);            % random 3-by-4 matrix
y = zeros(1, 10);          % length 10 row vector of all zeros
x = 1:10;                  % length 10 vector of 1, 2, ..., 10
z = linspace(0, 10, 100); % length 100 vector of values
evenly between 0 and 10
```

Матричні та векторні операції. Індексація: використовуйте a(i) для доступу до і-го елемента. Java базується на 0; Matlab — це індексація на основі 1. Підмасив і підматриці з двокрапкою.

Векторні операції: більшість функцій працюють з векторними аргументами. Також mean(), min(), max(), std(), median().

```
% plot sinc function = sin x / x, from pi/100 to 10pi
x = pi/100 : pi/100 : 10*pi;
% x = linspace(pi/100, 10*pi, 1000);
y = sin(x) ./ x;
plot(x, y)
```

Масштабування матриці, додавання, множення, транспонування, поелементна арифметика.

Векторизація. До Matlab 6.5 цикли виконувались досить довго. JIT-Accelerator покращив продуктивність циклів на порядки. Необхідно попередньо виділити пам'ять перед циклом.

```
x = linspace(0, 2*pi, N);
y = sin(x);
```

```
x = 0;
y = zeros(1, N);
for i = 1:N
y(i) = sin(2 * pi * (i-1) / N);
end
```

Вхід і вихід.

```
x = input('Enter a value\n');
Enter a value
3.14
```

`System.out.println("Hello")` `disp('Hello')` форматований друк `System.out.printf("%8.4f\n", x)` `fprintf('%8.4f\n', x)`

Функції. Matlab надає багато вбудованих функцій. Як і в Java, Matlab не може містити функції, які можуть вам знадобитися. Відповідно, ви можете реалізувати власні функції за допомогою функцій M-file. Функція має таке ж ім'я, як і M-файл.

Подібно до Java, функції M-file можуть мати будь-яку кількість вхідних аргументів. На відміну від Java, вони також можуть мати будь-яку кількість вихідних аргументів.

```
% keyword, output argument, function name, input arguments
function c = hypotenuse(a, b)
c = sqrt(a*a + b*b);
```

Може повертати кілька значень із функції.

```
function [c, d] = f(a, b)     % multiple output arguments
function g(a, b)              % no output arguments
```

Перевантаження. Неприємно підтримувати дві функції з однаковою назвою, але різними підписами. Ось хак.

```
function y = phi(x, mu, sigma)
   if nargin == 1
       y = exp(-x.^2 / 2) / sqrt(2 * pi);
   else
       y = exp(-((x-mu)/sigma).^2 / 2) / sqrt(2 * pi * sigma);
end
```

(У Matlab дійсно є каталоги класів, куди ви розміщуєте різні перевантажені функції, розміщуючи окремі підкаталоги з іменами \@int32 і \@double.)

звукові хвилі. Matlab особливо підходить для обробки сигналів.

```
function a = note(pitch, duration)
   hz = 440 * 2^(pitch / 12);
   mid = tone(1.0*hz, duration);
   hi  = tone(2.0*hz, duration);
   lo  = tone(0.5*hz, duration);
   a = .5*mid + .25*hi + .25*lo;
end
```

```
function a = tone(hz, duration)
   SAMPLES_PER_SECOND = 44100;
   N = floor(SAMPLES_PER_SECOND * duration);
   i = 0:N;
   a = sin (2 * pi * i * hz / SAMPLES_PER_SECOND);
end
```

Потім, щоб відтворити ноту:

```
plot(note(0, .005));
sound(note(0, 1), 44100);
```

Перейти за значенням. Коли ви викликаєте функцію (або виконуєте призначення), аргументи копіюються. Якщо ви змінюєте аргумент у функції, ви змінюєте локальну копію аргументу функції. Щоб змінити змінні у функції, потрібно повернути результат після завершення.

```
x = f(x);
```

Matlab використовує схему під назвою «копіювати при записі» для оптимізації, коли ви передаєте великий масив у функцію, він копіює його, лише якщо ви змінюєте одну зі змінних. Таким чином, ви можете вільно передавати великі масиви функціям, не турбуючись про накладні витрати (за умови, що ви їх не змінюєте).

Але будьте обережні, якщо ви передаєте великі масиви, а потім змінюєте змінні. Функція обміну дуже неефективна. Наступна функція займає час і простір, пропорційні довжині вектора x. Matlab постійно передає за значенням, тому якщо функція змінює масив, вона повинна повернути результат через вихідні параметри.

```
function x = swap(x, i, j)
%SWAP  Swap elements i and j in vector x
```

```
%   x = swap(x, i, j) swap elements i and j in array x

x([i j]) = x([j i]);
end

function x = shuffle(x)
%SHUFFLE  Shuffle an array of values.
%   x = shuffle(x) shuffles the vector x in random order.

%   Since this function changes the vector x,
%   it creates a copy of it.

n = length(x);
disp n
for index = 1:n

    % a random number between i and n
    r = index + floor((n-index+1) * rand());

    % swap elements index and r
    x([index r]) = x([r index]);
end
```

Підфункції. Може містити допоміжну функцію в тому самому М-файлі, що й основна функція. У Matlab це називається підфункціями. Їх не можна викликати безпосередньо з-поза М-файлу, наприклад, приватні. Лише перша функція, визначена у файлі, є публічною.

Застереження: y = f(x) може бути або викликом функції, або доступом до масиву, залежно від контексту.

Рекурсія. Рекурсія повністю підтримується.

```
function d = mygcd(x, y)
%MYGCD  Return the gcd of two integers.
```

```
%   d = gcd(x, y) returns the greatest common divisor
%   of x and y.

% awkward type checking
if (round(x) ~=x) || (round(y) ~=y)
    error('Requires integer input arguments.')
end

% Euclid's algorithm
if y == 0
    d = x;
else
    d = mygcd(y, rem(x, y));
end
```

Бібліотеки. Ключовою особливістю Matlab є його великі числові бібліотеки. Обробка зображень, оптимізація, ...

Багаті бібліотеки для наукових обчислень: besselj(), gcd(),

Кожен файл М-функцій може мати лише одну функцію, видиму зовнішньому світу. Призводить до надлишку файлів. Програмісти Matlab можуть організувати купу пов’язаних М-файлів у наборі інструментів, що фактично зводиться до зберігання багатьох файлів в одному каталозі.

Лінійна алгебра. У Matlab основним типом даних є складна матриця. Скаляр — це матриця 1 на 1; вектор — це матриця 1 на N або N на 1. Далі показано, як розв’язувати системи лінійних рівнянь за допомогою оператора \. Коли система надмірно визначена, результатом є рішення за методом найменших квадратів. (Див. розділ xyz.)

```
>> A = [0, 1, 1; 2, 4, -2; 0, 3, 15]
A =  0     1     1
     2     4    -2
     0     3    15
```

```
>> b = [4; 2; 36]
b =  4
     2
    36

>> x = A \ b
x = -1
     2
     2

>> A * x
ans = 4.0000
      2.0000
     36.0000
```

Однією з найпотужніших функцій Matlab є його велика бібліотека чисельної лінійної алгебри.

```
>> cond(A)
ans = 58.8670

>> eig(A)
ans =  13.5765
       -0.6419
        2.0654

>> [Q R] = qr(A)
Q =  -0.1078    0.5683   -0.8157
     -0.6470   -0.6631   -0.3765
     -0.7548    0.4872    0.4392

R =  -9.2736   -9.4893   -7.7640
           0    1.7186    5.4264
           0         0    1.1294
```

```
>> svd(A)
ans =  15.8836
        4.2000
        0.2698
```

Бібліотеки Matlab. Matlab також має численні бібліотеки, призначені для наукових і комерційних застосувань, включаючи: розв'язування систем ODE, обробку сигналів, вейвлети, розв'язування рівнянь, лінійну та нелінійну оптимізацію, нейронні мережі, обробку зображень, інтерполяцію, поліноми, аналіз даних, перетворення Фур'є, елементарні та спеціальні математичні функції та цифрове аудіо. Підручник Matlab Також бібліотеки рядків, включаючи регулярні вирази.

Програмування в Matlab. Цикли, умови, функції, створення багаторазових бібліотек, дескриптори функцій, типи даних (8, 16, 32-розрядні цілі числа, багатовимірні масиви, рядки, структури). MATLAB містить функції ООП, включаючи класи та об'єкти, перевантаження операторів, успадкування, хоча це досить добре приховано. Можна навіть викликати Java з Matlab. Однак Matalb не підтримує передачу аргументів за посиланням (вказівники).

Усі об'єкти незмінні – щоб «змінити» об'єкт, вам потрібно передати об'єкт як вхідний аргумент і повернути новий об'єкт зі зміненими даними. Приватні допоміжні методи реалізуються шляхом розміщення методів у підкаталозі під назвою private. Matlab підтримує перевантаження операторів. використовувати метод відображення як аналог toString(). Використовуйте subsref, щоб перевантажити посилання з індексом, плюс, мінус, mtimes, для додавання, віднімання та множення.

Функції функції. Дескриптори функцій є типами даних у Matlab. Може передавати функції іншим функціям (feval, eval, fzero, ezplot).

Структури даних. Немає підтримки посилань або покажчиків. Важко реалізувати пов'язані структури.

Інтеграція з Java. Matlab тісно інтегрований з Java - інтерпретатор Matlab написаний на Java. Може безпосередньо викликати код Java.

```
s = java.lang.String('привіт, світ')
s = s.replaceAll('l', 'abc')
```

Таким чином можна використовувати Matlab з його розвиненою системою програмування для побудови біблеотек моделей розрахунку квадрокоптерів.

РОЗДІЛ 3
ОПИС ПРОГРАМИ РОЗРАХУНКУ

3.1. Модуль обчислення дінаміки

Система для розрахунку складається з двох частин: бібліотеки та графічного інтерфейсу.

Головна функція для розрахунку балансувальних характеристик має наступний заголовок

```
function result = Calc_BalansQuadro(M_all, pos_eng, k_eng, Scx_fusel, Smx_fusel,  Scx_wing, Scy_wing, fy_wing, L_wing)
```

Вхідними даними для функції є

M_all – загальна маса квадрокоптера;

pos_eng – відстань від початку координат до осі двигуна;

k_eng – коефіцієнт перетворення для двигуна;

Scx_fusel – коефіцієнт опору планера;

Smx_fusel – коефіцієнт моменту планера;

Scx_wing – коефіцієнт опору крила;

Scy_wing – коефіцієнт підйомної сили крила;

L_wing – координата розташування центру консолі крила.

Приклад виклику функції наведено нижче

```
Calc_BalansQuadro(4, 0.2, 50, 0.1, 0.01, 0, 0, 0, [0;0;0])
```

Основні визначення констант

```
ro2 = 1.25/2; % плотность воздуха
g = 9.81;
```

Координати центрів гвинтів приймаються в кутах чотирикутника

```
% координаты винтов
%     +x
%   2    1
%-z        +z
%   3    4
%     -x
L_eng = pos_eng*[[1; 0; 1] [1; 0; -1] [-1; 0; -1] [-1; 0; 1]];
%L_eng = pos_eng*[[1; 0; 0] [0; 0; -1] [-1; 0; 0] [0; 0; 1]];
```

Формуємо структуру даних для розміщення результату

```
idx_max = 10;
idx=1;
result.Vg = zeros(idx_max,1);
result.tet = zeros(idx_max,1);
result.gam = zeros(idx_max,1);
result.c_all = zeros(idx_max,1);
result.c_roll = zeros(idx_max,1);
result.c_pitch = zeros(idx_max,1);
```

Виконуємо обчислення для заданого набору швидкостей

```
for Vgv  = 0:5:40
    Vg = [Vgv; 0; 0]; % Задана швидкість в земній системі
координат
```

Формування матриці повороту на кут встановлення крила

```
RotW = [[cos(fy_wing) -sin(fy_wing) 0];[sin(fy_wing) cos(fy_wing) 0];[0 0 1]];
```

Початкові наближення

```
tet = 0;
gam = 0;
c_all=0.2;
c_roll=0;
c_pitch =0;
cnt =0;
```

Створюємо цикл для обчислення положення органів керування методом послідовного наближення

```
while 1
     cnt = cnt+1;
```

Створюємо матрицю повороту по крену

```
RotE1 = [[1 0 0];[0 cos(gam) -sin(gam)];[0 sin(gam) cos(gam)]];
```

Створюємо матрицю повороту по тангажу

```
RotE2 = [[cos(tet) -sin(tet) 0];[sin(tet) cos(tet) 0];[0 0 1]];
```

Створюємо матрицю повороту Ейлера

```
RotE = RotE2*RotE1;
```

Перетворення швидкості у зв’язану систему координат

```
Vxyz = inv(RotE)*Vg;
```

Обчислення сумарних керуючих впливів на двигуни

```
P0_all   = [ c_all; c_all; c_all; c_all];
P0_roll  = [ -c_roll; c_roll; c_roll; -c_roll];
P0_pitch = [ c_pitch; c_pitch; -c_pitch; -c_pitch ];
P0_eng = k_eng*(P0_all + P0_roll + P0_pitch);
```

Обчислення сили тяги від несучих гвинтів

```
P_eng  =[[0;  P0_eng(1);  0]  [0;  P0_eng(2);  0]  [0;
P0_eng(3) ;0] [0; P0_eng(4); 0]];
F_eng = P_eng(:,1)+P_eng(:,2)+P_eng(:,3)+P_eng(:,4);
```

Розрахунок моментів від несучих гвинтів

```
M_eng                                                  =
cross(L_eng(:,1),P_eng(:,1))+cross(L_eng(:,2),P_eng(:,2))+cross(L_
eng(:,3),P_eng(:,3))+cross(L_eng(:,4),P_eng(:,4));
```

Розрахунок сил та моментів від планера

```
F_Fusel  =  [-Scx_fusel*Vxyz(1)*abs(Vxyz(1))*ro2;  0;  -
Scx_fusel*Vxyz(3)*abs(Vxyz(3))*ro2 ];
M_Fusel  =  [-Smx_fusel*Vxyz(1)*abs(Vxyz(1))*ro2;  0;  -
Smx_fusel*Vxyz(3)*abs(Vxyz(3))*ro2 ];
```

Розрахунок сил та моментів від крила

```
            V_wing=0;
            sign_x =0;
            sign_y = 0;
            F0_wing     =     [-Scx_wing*V_wing*V_wing*sign_x*ro2;
Scy_wing*V_wing*V_wing*sign_y*ro2; 0];
            F_wing = RotW * F0_wing;
            M_wing = cross(L_wing, F_wing);
```

Перетворення сили тяження в зв’язану систему координат

```
            G_all=[0; -M_all*g; 0 ];
            G_0 = inv(RotE)*G_all;
```

Визначення сумарних сил та моментів для системи в цілому

```
            F_sum = F_eng + F_Fusel + F_wing + G_0;
            M_sum = M_eng + M_Fusel + M_wing;
```

Реалізація методу послідовних наближень та збереження розрахованих значень

```
            if  (  (  norm(F_sum)<1e-2  &&  norm(M_sum)<1e-3  )  ||
cnt>50000 )
                result.Vg(idx) = Vgv;
                result.tet(idx) = tet;
                result.gam(idx) = gam;
                result.c_all(idx) = c_all;
                result.c_roll(idx) = c_roll;
                result.c_pitch(idx) = c_pitch;

                cnt;
                idx = idx+1;
                break
            end
```

Коректування керуючих сигналов за методом послідовних наближень

```
        tet = tet + F_sum(1)*1e-4;
        gam = gam - F_sum(3)*1e-4;
        c_all=c_all - F_sum(2)*1e-3;
        c_roll= c_roll - M_sum(1)*1e-3;
        c_pitch = c_pitch - M_sum(3)*1e-3;
    end
end
result
```

Таким чином можна отримати масив точок для різних умов та сполучень характеристик і побудувати відповідні графіки.

3.2. Модуль графічного інтерфейсу

Програма може мати одне головне вікно або кілька вікон, дозволяючи виводити графічну та текстову інформацію як у головне вікно програми, так і у окремі вікна. В MATLAB доступний набір функцій для створення стандартних діалогових вікон, таких як вікна відкриття та збереження файлів, друк, вибір шрифту, вікна для введення даних тощо, які можна використовувати у власних програмах.

Додаток із графічним інтерфейсом може бути написано без застосування середовища GUIDE. Як приклад можна навести програму bspligui, що входить до складу Spline ToolBox.

Давайте розглянемо створення програми в середовищі GUIDE, яка надає можливість створювати графіки функцій. Наприклад, розглянемо ситуацію, коли потрібно створити програму з графічним інтерфейсом, яку можна запустити з командного рядка

```
graphic;
```

Для створення програми запусткаємо середовище GUIDE за допомогою команди

```
guide;
```

В результаті запуститься майстер створення графічного інтерфейсу та з'явиться діалогове вікно GUIDE Quick Start (рис. 3.1).

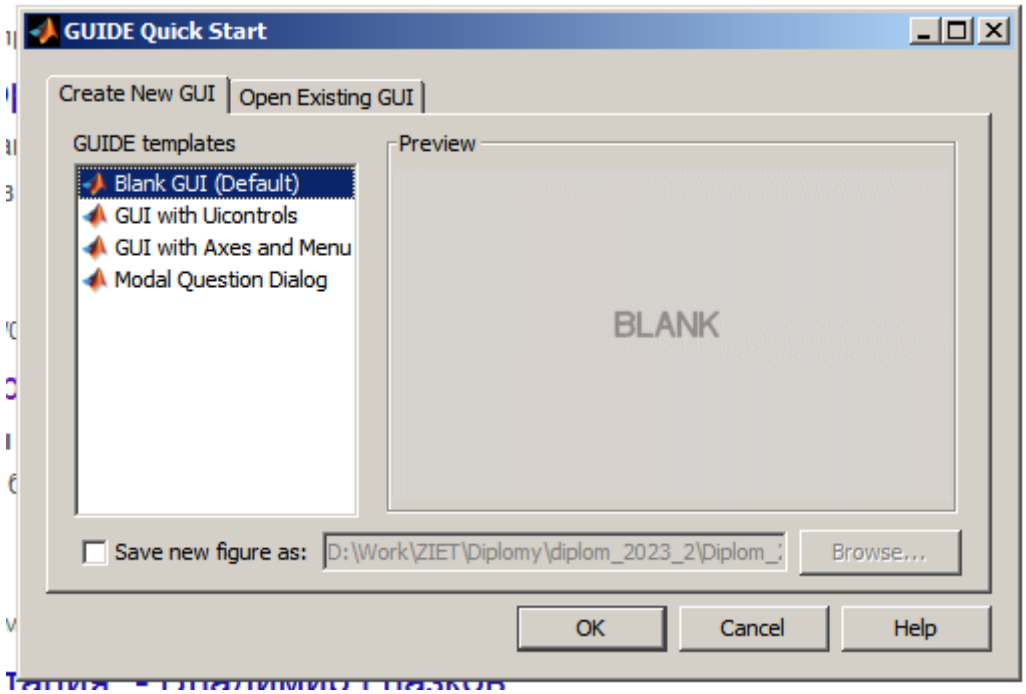

Рис. 3.1. Діалогове вікно GUIDE

Діалогове вікно GUIDE має дві вкладки:

Вкладка "Створення нового GUI" надає можливість створення нової програми з використанням різних варіантів. Тут доступні чотири типи шаблонів: "Порожній GUI" для створення пустого вікна програми, "GUI з елементами керування" з кнопками, перемикачами та полями введення, "GUI з віссю та меню" з осіми, меню, кнопкою та списком, а також "Модальне діалогове вікно" для створення вікна запитань.

Вкладка Open Existing GUI - відкриття існуючої програми.

Біля вкладки "Створення нового GUI" знаходиться опція, яка дозволяє призначити ім'я файлу, у якому буде збережено створену програму. Зрозуміло, що цей прапор не обов'язковий для встановлення, оскільки можна буде зберегти додаток під час його редагування.

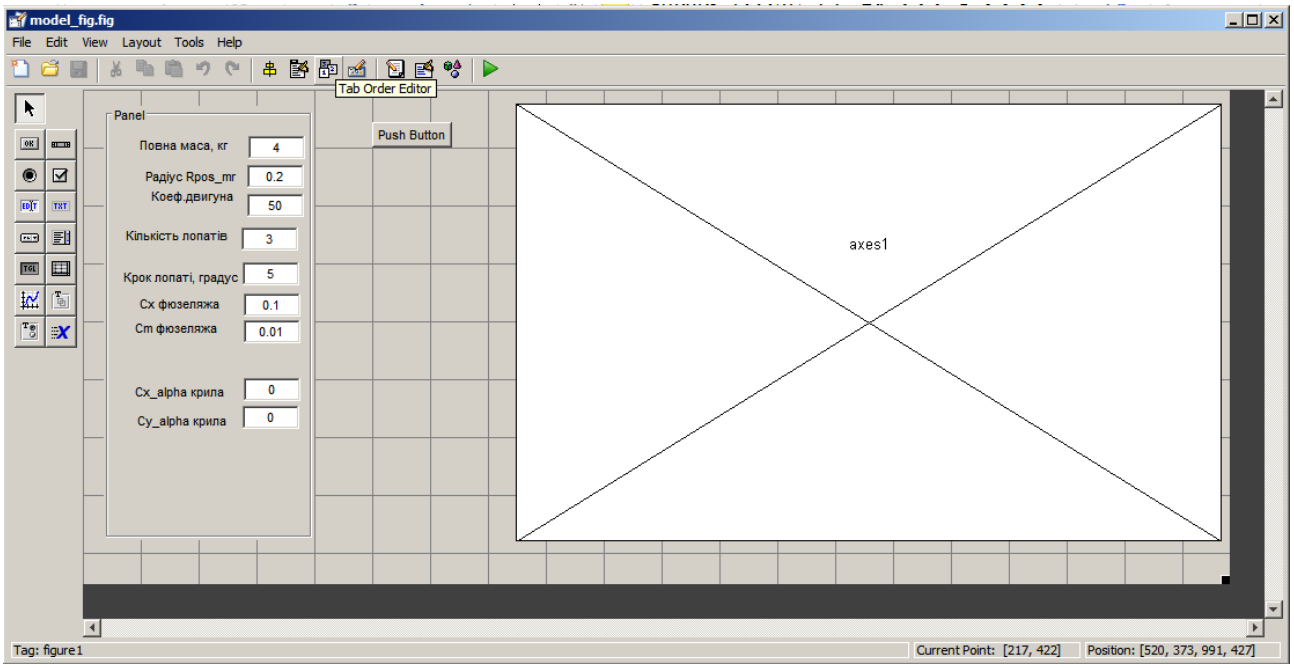

Рис.3.2 Проект діалогового вікна програми

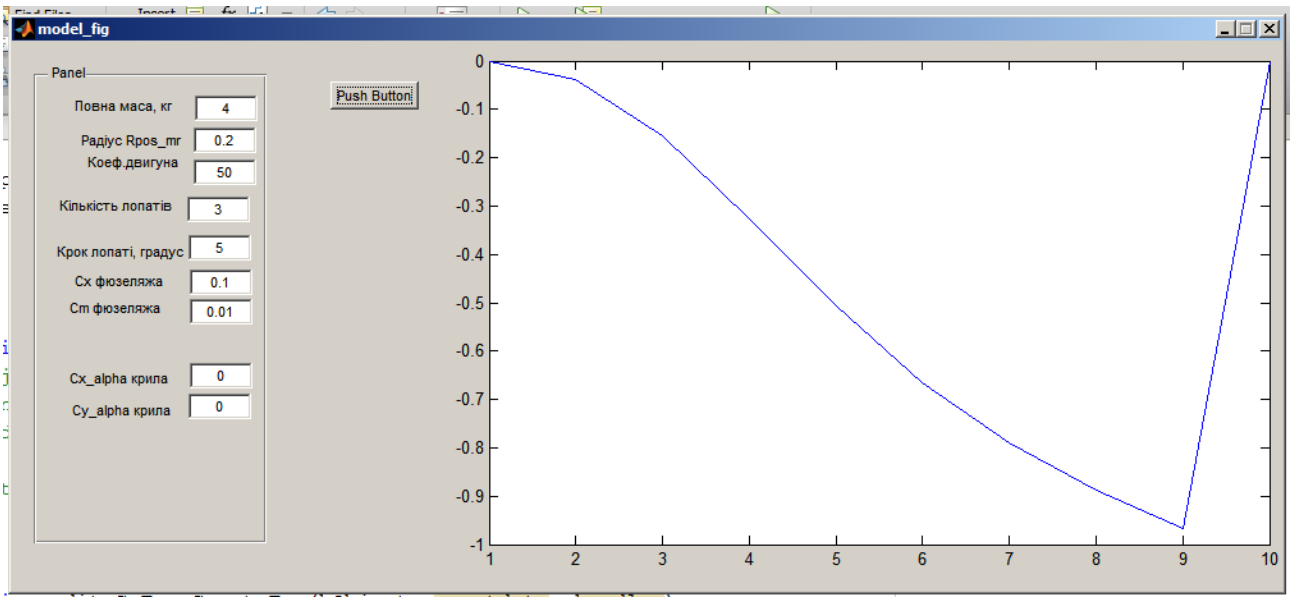

Рис. 3.3 Діалогове вікно програми

Функція для створення інтерфейсу програми має наступний вигляд

```
function varargout = model_fig(varargin)
```

Функція MODEL_FIG виконує ініціалізацію для інтерфейса model_fig.fig.

```
H = MODEL_FIG
```

Функція MODEL_FIG створює нову MODEL_FIG або повертає існуючий інтерфейс.

```
MODEL_FIG('CALLBACK',hObject,eventData,handles,...)
```

викликає локальну функцію під назвою CALLBACK у MODEL_FIG.M із заданими вхідними аргументами.

```
MODEL_FIG('Property','Value',...)
```

створює новий інтерфейс MODEL_FIG або модіфікує існуючий. Починаючи зліва, пари значеня-властивості застосовуються до GUI перед викликом model_fig_OpeningFcn. Нерозпізнане ім’я властивості або недійсне значення призводять до зупинки застосування властивості. Усі вхідні дані передаються в model_fig_OpeningFcn через varargin.

Початок ініціалізації графічного інтерфейсу має наступний вид

```
gui_Singleton = 1;
gui_var_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @model_fig_OpeningFcn,...
                   'gui_OutputFcn',  @model_fig_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(vararg_in{1})
    gui_var_State.gui_Callback = str2func(vararg_in{1});
end

if nargout
    [vararg_out{1:nargout}]    =    gui_mainfcn(gui_var_State,
varargin{:});
else
    gui_mainfcn(gui_var_State, vararg_in{:});
end
```

Функція model_fig_OpeningFcn виконується безпосередньо перед тим, як інтерфейс model_fig стає видимим.

```
function  model_fig_OpeningFcn(hObject,  eventdata,  handles,
varargin)
```

Вхідні аргументи наступні:

hObject - дескриптор для об’єкту;

eventdata – зарезервований параметр;

handles - структура із дескрипторами та даними користувача;

varargin – аргументи командного рядка varargin для model_fig.

Виконується встановлення командного рядка за замовчуванням для інтерфейсу model_fig

```
handles.output = hObject;
```

Виконується оновлення структури дескриптора

```
guidata(hObject, handles);
```

За необхідністю створюється для інтерфейсу model_fig цикл очикування для реакції пористувача

```
% uiwait(handles.figure1);
```

Визначення функції, що повертає значення для командного рядка. Вхідні аргументи цієї функції такі самі як і для функції model_fig_OpeningFcn

```
function varargout = model_fig_OutputFcn(hObject, eventdata, handles)
```

Отримати стандартний вихід командного рядка зі структури дескриптора

```
varargout{1} = handles.output;
```

Розглянемо функції зворотньюго виклику для елементів інтерфейсу.

Функція pushbutton_Calc_Callback виконується після натискання кнопки pushbutton_Calc.

```
function pushbutton_Calc_Callback(hObject, eventdata, handles)
```

Зчитування значень елементів для введення тексту

```
Mass = str2double( get(handles.edit_Mass,'String') );
EngPos = str2double( get(handles.edit_EngPos,'String') );
Keng = str2double( get(handles.edit_Keng,'String') );
Nlop = str2double( get(handles.edit_Nlop,'String') );
FiLop = str2double( get(handles.edit_FiLop,'String') );
CxFus = str2double( get(handles.edit_CxFus,'String') );
CmFus = str2double( get(handles.edit_CmFus,'String') );
CxWing = str2double( get(handles.edit_CxWing,'String') );
CyWing = str2double( get(handles.edit_CyWing,'String') );
```

Виклик функції розрахунку балансувальних характеристик квадрокоптеру

```
result_calc = Calc_BalansQuadro(Mass, EngPos, Keng, CxFus, CmFus, CxWing, CyWing, 0, [0;0;0]) ;
```

Побудова графіків балансувальних характеристик

```
axes(handles.axes1);
cla;
plot(result_calc.Vg, result_calc.c_roll, result_calc.Vg, result_calc.c_pitch, result_calc.Vg, result_calc.c_all);
```

Функція edit_FiLop_Callback виконується при змінах у полі вводу edit_FiLop.

```
function edit_FiLop_Callback(hObject, eventdata, handles)
% Приклад доступу до значення
str2double(get(hObject,'String'))
```

Функція edit_FiLop_CreateFcn виконується під час створення об'єкта, після встановлення всіх властивостей.

```
function edit_FiLop_CreateFcn(hObject, eventdata, handles)
% Елементи редагування мають білий фон у Windows.
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Функція edit_Nlop_Callback виконується при змінах у полі вводу edit_ Nlop.

```
function edit_Nlop_Callback(hObject, eventdata, handles)
```

Функція edit_Nlop_CreateFcn виконується під час створення об'єкта, після встановлення всіх властивостей.

```
function edit_Nlop_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Функція edit_Mass_Callback виконується при змінах у полі вводу edit_ Mass.

```
    function edit_Mass_Callback(hObject, eventdata, handles)
    function edit_Mass_CreateFcn(hObject, eventdata, handles)
    if     ispc     &&     isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

Далі пердставлені аналогічні функції для інших елементів інтерфейсу.

Функції для елементу редагування edit_CxFus

```
    function edit_CxFus_Callback(hObject, eventdata, handles)
    function edit_CxFus_CreateFcn(hObject, eventdata, handles)
    if     ispc     &&     isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

Функції для елементу редагування edit_CxWing

```
    function edit_CxWing_Callback(hObject, eventdata, handles)
    function edit_CxWing_CreateFcn(hObject, eventdata, handles)
    if     ispc     &&     isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

Функції для елементу редагування edit_CyWing

```
    function edit_CyWing_Callback(hObject, eventdata, handles)
    function edit_CyWing_CreateFcn(hObject, eventdata, handles)
    if     ispc     &&     isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
    end
```

Функції для елементу редагування edit_EngPos

```
function edit_EngPos_Callback(hObject, eventdata, handles)
function edit_EngPos_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Функції для елементу редагування edit_Keng

```
function edit_Keng_Callback(hObject, eventdata, handles)
function edit_Keng_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Функції для елементу редагування edit_CmFus

```
function edit_CmFus_Callback(hObject, eventdata, handles)
function edit_CmFus_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

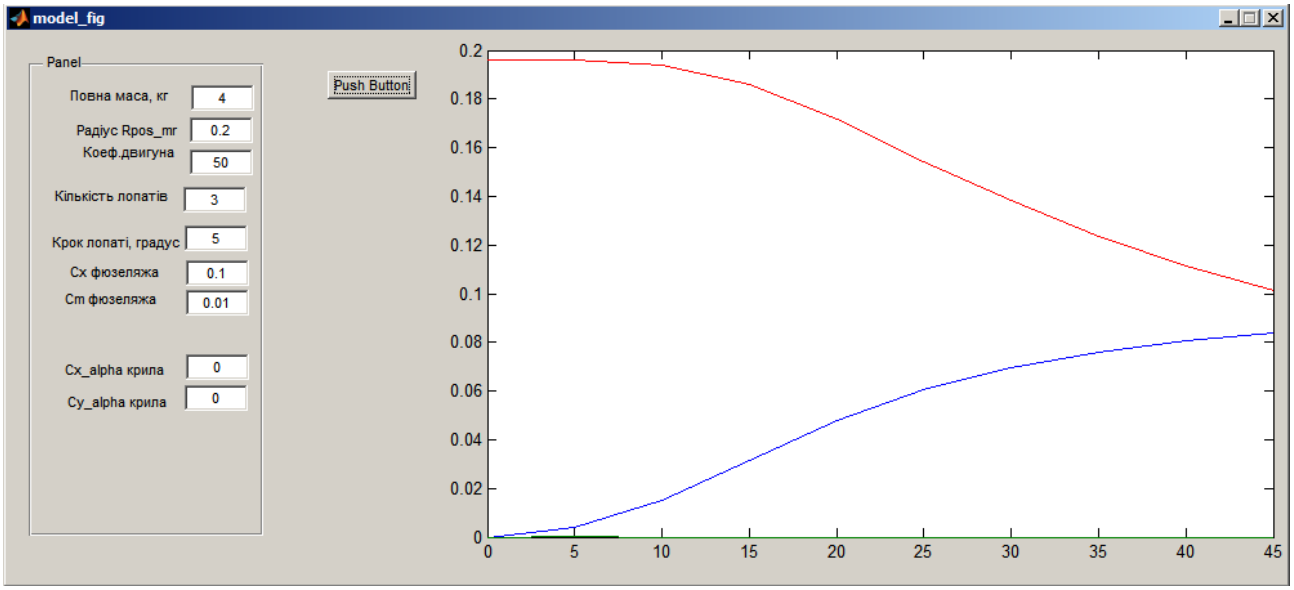

Рис. 3.4 Результат роботи моделі для набора m=4, Cx=0.1, Cm=0.01

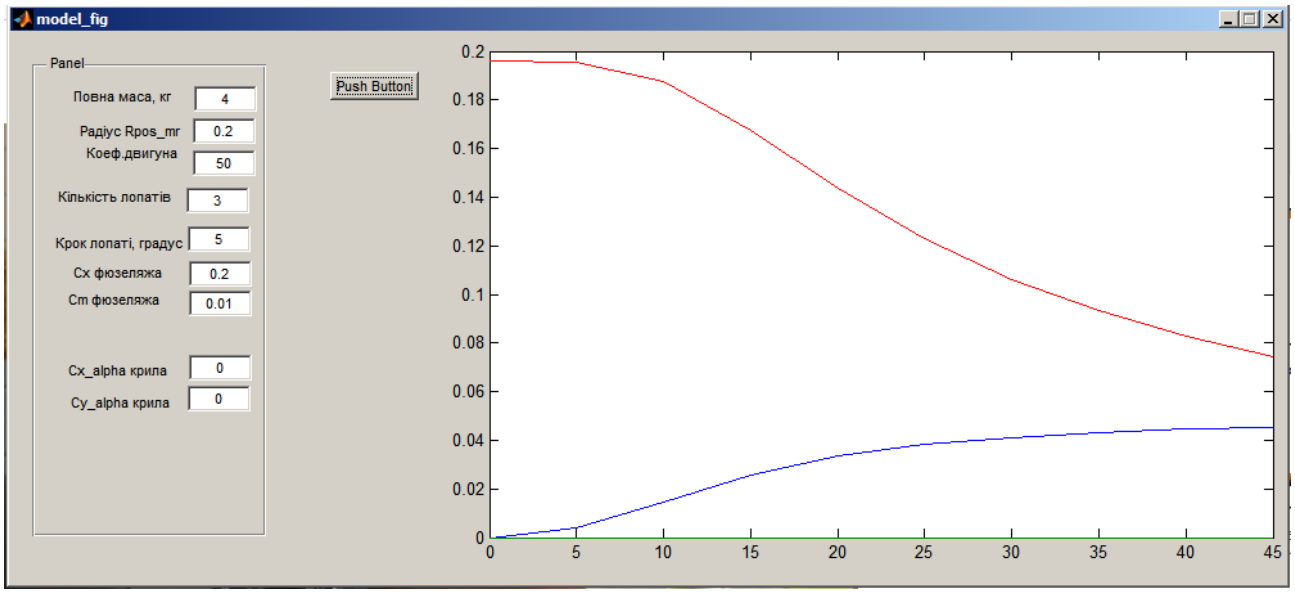

Рис. 3.5 Результат роботи моделі для набора m=4, Cx=0.2, Cm=0.01

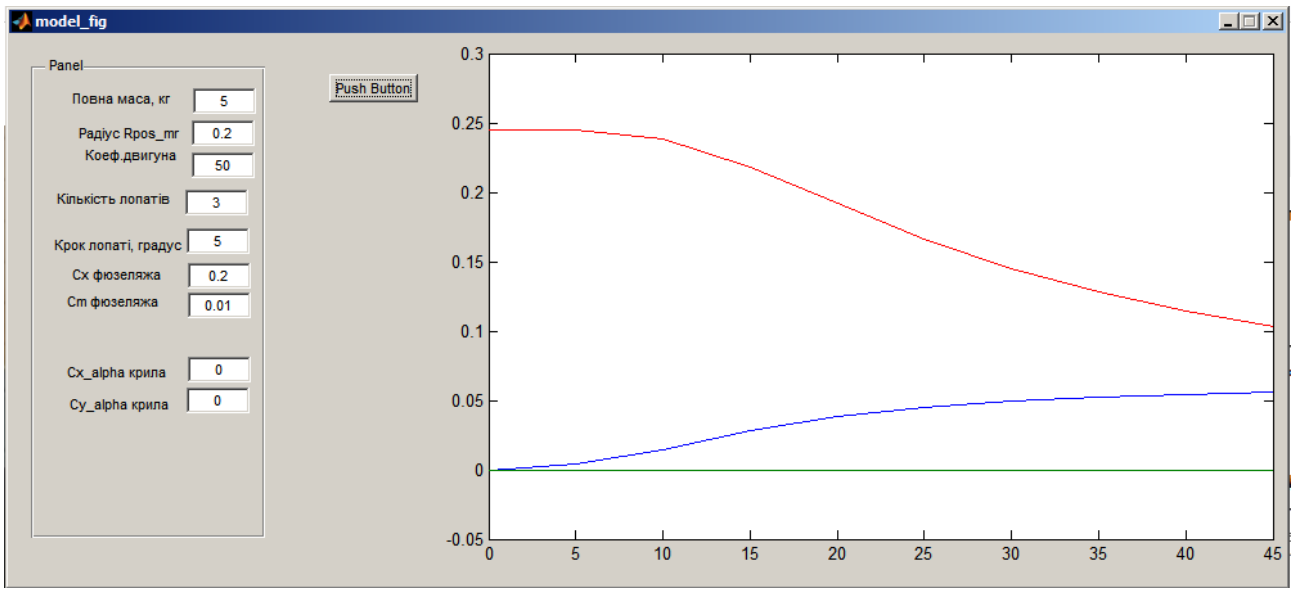

Рис. 3.6 Результат роботи моделі для набора m=5, Cx=0.1, Cm=0.01

ВИСНОВКИ

Основна увага була приділена дослідженню особливостей конструкції та функціональних можливостей квадрокоптерів, що є важливим для розуміння принципів їх роботи та можливостей управління.

Досліджено різноманітні схеми та системи управління квадрокоптерами, що відіграють важливу роль у забезпеченні стабільності та керованості літального апарата.

Розглянуті динамічні моделі є ключовими для математичного опису поведінки квадрокоптера у просторі та часі, що дозволяє проводити реалістичні обчислення та розрахунки.

Розроблений алгоритм є основою для обчислення різних характеристик та параметрів квадрокоптера, від інерції до стабілізації польоту.

Вивчено можливості та особливості програмування у середовищі Matlab, що є важливим для ефективної розробки та тестування алгоритмів.

Розроблений програмний модуль дозволяє проводити обчислення динамічних параметрів квадрокоптера на основі розглянутих моделей.

Створений графічний інтерфейс забезпечує зручну взаємодію з розробленими алгоритмами та модулями для виконання розрахунків.

Проведене тестування дозволило перевірити працездатність та ефективність розробленої бібліотеки, її здатність до точних розрахунків та адекватності результатів.

РЕКОМЕНДАЦІЇ

Удосконалення графічного інтерфейсу для спрощення взаємодії та роботи з алгоритмами, а також для підтримки практичних задач моделювання управління квадрокоптерами.

Вдосконалення алгоритмів динамічних моделей для забезпечення більш точних обчислень та збільшення стабільності та прогностичності квадрокоптерів у різних умовах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лазарєв Ю. Ф. Довідник з MATLAB / Електронний навчальний посібник з курсового і дипломного проектування. – К.: НТУУ "КПІ", 2013. – 132 с.
2. Інформаційні технології: Системи комп’ютерної математики [Електронний ресурс] : навч. посіб. для студ. спеціальності «Автоматизація та комп’ютерно-інтегровані технології» / І. В. Кравченко, В. І. Микитенко; КПІ ім. Ігоря Сікорського . – Електронні текстові дані (1 файл: 5,57 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 243с.
3. Моделювання об’єктів та систем керування засобами MatLab: навч. посіб. для студ. вищ. навч. закл. / М. В. Коржик. − Київ : НТУУ “КПІ”, 2016. − 174 с. : іл.
4. Халатов А.А, Немчін О.Ф., Шквар Є.О., Кузьмін А.В., Кобзар С.Г. Бойові малорозмірні безпілотні літальні апарати з реактивною тягою: монографія / під ред. Академіка НАН України, д.т.н., проф. А.А. Халатова. – Дніпро: Ліра, 2023. – 144 с.
5. Лебедев А. А., Чернобровкин Л. С. Динамика полета беспилотных летательных аппаратов. Учебное пособце для вузов. Изд. 2-е, переработанное и доп. М., «Машиностроение», 1973, 616 с.
6. Биард У. Рэндал, МакЛэйн У. Тимоти Малые беспилотные летательные аппараты: теория и практика / Рэндал У. Биард, Тимоти У. МакЛэйн — Киев: Изд. дом «СВАРОГ», 2023. — 312 с.

ДОДАТКИ

ДОДАТОК А

Вихідний код програми