Міністерство освіти і науки України
Державний вищий навчальний заклад
Львівський національний університет імені Івана Франка
Факультет електроніки та комп’ютерних технологій

БАКАЛАВРСЬКА РОБОТА

на тему: «Комп’ютерний розрахунок процесу набору висоти літаком з врахуванням ISA deviation»

Виконав студент IV курсу
Спеціальність 122 «Комп’ютерні науки»
Локоть Владислав Олександрович

Керівник: асист. Баран М. О.
Рецензент: доц. Лис Р. М.

м. Львів – 2025 р.

Анотація

Дослідження присвячено розробці алгоритму оптимізації витрат авіаційного палива з урахуванням впливу атмосферних умов, зокрема відхилень від міжнародної стандартної атмосфери (ISA). Метою роботи є підвищення ефективності планування польотів шляхом динамічного вибору висоти на основі аналізу температурних аномалій, маси повітряного судна та метеорологічних даних. На основі математичного моделювання взаємодії аеродинамічних параметрів (TAS, IAS), маси літака та зовнішніх факторів запропоновано ітеративний підхід, що передбачає сегментацію маршруту.

Результати демонструють зниження витрат палива для Boeing 738 за рахунок оптимізації висоти та врахування ISA-відхилень. Практична цінність роботи полягає у можливості застосування алгоритму в системах управління польотами для автоматизації розрахунків, що особливо актуально в умовах зростання вимог до екологічної та економічної ефективності авіаперевезень.

Abstract

The study focuses on developing an algorithm for optimizing aircraft fuel consumption while accounting for atmospheric conditions, particularly deviations from the International Standard Atmosphere (ISA). The research aims to enhance flight planning efficiency through dynamic altitude selection based on analysis of temperature anomalies, aircraft mass, and meteorological data. Using mathematical modeling of interactions between aerodynamic parameters (TAS, IAS), aircraft mass and external factors, the study proposes an iterative approach involving route segmentation.

Results demonstrate reduction in fuel consumption for Boeing 738 through altitude optimization and ISA deviation adjustments. The practical value lies in the algorithm's potential application in Flight Management Systems to automate calculations, which is particularly relevant given increasing demands for environmental and economic efficiency in air transport.

Зміст

Вступ

У сучасній авіації ефективне управління польотом, зокрема вибір оптимальної висоти, є ключовим фактором для зниження витрат палива, зменшення часу перельоту та забезпечення екологічної стійкості. Особливу роль у цих розрахунках відіграють атмосферні умови, зокрема відхилення від стандартної атмосфери (ISA deviation), які впливають на аеродинамічні характеристики літака та споживання палива. Тому розробка інструменту для автоматизованого розрахунку оптимальної висоти польоту з урахуванням ISA deviation є актуальною задачею для авіаційної галузі.

Актуальність теми. Оптимізація висоти польоту дозволяє авіакомпаніям значно знизити витрати на паливо, що становить одну з найбільших статей витрат у комерційній авіації. Крім того, правильний вибір висоти впливає на тривалість польоту, зменшує навантаження на двигуни та покращує екологічні показники. Сучасні системи планування польотів враховують багато факторів, проте автоматизований підхід до вибору висоти з урахуванням ISA deviation, маси літака та інших параметрів дозволяє отримати більш точні результати порівняно з традиційними методами.

Мета дослідження. Метою даної роботи є розробка програмного забезпечення для комп’ютерного розрахунку оптимальної висоти польоту літака Boeing 738 з урахуванням ISA deviation, маси літака, маршруту та інших параметрів. Програма повинна визначати висоту, на якій витрата палива буде мінімальною, а також обчислювати загальну дистанцію, час польоту, кількість витраченого палива та кінцеву масу літака.

Об’єкт дослідження. Об’єктом дослідження є процес набору висоти та крейсерського польоту літака Boeing 738 у різних атмосферних умовах.

Предмет дослідження. Предметом дослідження є математичні моделі та алгоритми розрахунку витрати палива, інтерполяції за висотою, масою та ISA deviation, а також програмна реалізація цих алгоритмів у графічному інтерфейсі.

Методи дослідження. У роботі використано:

- методи лінійної інтерполяції для врахування змінної маси літака, ISA deviation та висоти польоту;
- алгоритми оптимізації для визначення найефективнішої висоти;
- геометричні розрахунки для визначення пройденої дистанції на основі координат маршруту;
- інструменти програмування для створення графічного інтерфейсу та обробки даних.

Теоретичне та практичне значення. Теоретична цінність роботи полягає у вдосконаленні методів розрахунку оптимальної висоти польоту з урахуванням динамічної зміни маси та атмосферних умов. Практичне значення полягає в тому, що розроблена програма може бути використана диспетчерами, пілотами та авіакомпаніями для планування польотів із мінімальними витратами палива.

Наукова новизна. У роботі запропоновано підхід до інтеграції бази даних повітряних трас, навігаційних точок та характеристик літака в єдину систему, що дозволяє автоматично визначати оптимальну висоту польоту з урахуванням ISA deviation. Також розроблено алгоритми інтерполяції, які підвищують точність розрахунків порівняно з табличними методами.

Дана робота спрямована на розробку інструменту для автоматизованого вибору оптимальної висоти польоту з урахуванням ISA deviation. Реалізована програма дозволяє зменшити витрати палива, підвищити ефективність польотів та може бути інтегрована в сучасні системи планування маршрутів.

РОЗДІЛ 1. АНАЛІТИЧНИЙ ОГЛЯД ЛІТЕРАТУРИ

1.1 Міжнародний стандарт атмосфери (ISA) та його вплив

Міжнародний стандарт атмосфери (ISA) — це уніфікована модель, яка описує ідеальні умови в атмосфері на різних висотах. Вона визначає, якими мають бути температура, тиск, густина повітря та інші параметри за відсутності будь-яких аномалій. Ця модель служить еталоном для розрахунків у авіації, дозволяючи порівнювати льотні характеристики літаків у різних умовах.

Пілоти використовують ISA для планування польотів, зокрема для вибору крейсерських висот та оцінки витрати палива. Наприклад, на високогірних аеропортах, таких як Денвер (висота 5430 футів), високі температури можуть значно знизити продуктивність, що вимагає корекції навантаження або вибору іншого типу літака.

Концепція стандартної атмосфери виникла на початку 20 століття з ростом авіації, коли виникла потреба в єдиному стандарті для порівняння. Раніше існували національні стандарти, такі як американська стандартна атмосфера 1958 року, яка була оновлена у 1976 році. Міжнародний стандарт ISA був офіційно затверджений у 1975 році, що забезпечило глобальну уніфікацію для авіаційних розрахунків

За стандартом ISA, температура на рівні моря приймається за +15°C, тиск — 1013,25 гПа, а густина повітря — 1,225 кг/м³. Зі збільшенням висоти температура знижується з певним градієнтом (приблизно 2°C на 1000 футів у тропосфері), а тиск і густина повітря також зменшуються. Ці параметри критично важливі для розрахунків аеродинаміки, тяги двигунів та витрати палива.

Стандартні значення на рівні моря

На рівні моря ISA встановлює наступні параметри:

Температура: +15°C (288,15 K),

Тиск: 1013,25 гПа (101325 Па),

Густина повітря: 1,225 кг/м³.

У тропосфері температура знижується зі швидкістю приблизно 2°C на кожні 1000 футів (6,5°C на кілометр), тоді як тиск і густина зменшуються відповідно до гідростатичного рівняння та закону ідеального газу. Наприклад, на висоті 3000 метрів стандартна температура ISA становить -4,5°C, що розраховується як 15 - (6,5 * 3) = -4,5°C.

Реальні атмосферні умови рідко точно відповідають ISA. Відхилення виникають через сезонні коливання, географічні особливості, погодні явища тощо. Наприклад:

- **ISA-plus** (температура вища за стандартну) — призводить до розрідження повітря, що знижує підйомну силу крил і ефективність двигунів. У таких умовах літаку потрібна більша дистанція для зльоту, а набір висоти відбувається повільніше через зменшення тяги.
- **ISA-minus** (температура нижча за стандартну) — сприяє підвищенню густини повітря, що покращує аеродинамічні характеристики. Однак надто низькі температури можуть ускладнити роботу систем літака, наприклад, обмерзання.
- Висота FL - В авіації та авіаційній метеорології ешелоном польоту (flight level (FL) є умовна висота за стандартного тиску, що подається цифрою у сотнях футів. Тиск визначений, виходячи з даних тиску рівня моря за стандартної атмосфери в значенні 1013,2 гПа (760 мм або 29,92 дюймів ртутного стовпа), в зв'язку з чим не обов'язково відповідатиме справжній висоті польоту як над середнім рівнем моря, так і над поверхнею Землі.
- TAS Істинна швидкість (у авіації) — це швидкість повітряного судна відносно повітря, в якому воно рухається, з урахуванням поправки на зміну густини (щільності) повітря, залежно від барометричної висоти.
- Фунт (англ. pound) — одиниця ваги та маси, розповсюджена у різних країнах у різні часи.
- Інтерполяція — в обчислювальній математиці спосіб знаходження проміжних значень величини за наявним дискретним набором відомих значень.

- Тропопа́уза — порівняно тонкий шар у атмосфері планети, де припиняється зниження температури з висотою і вище якого атмосфера стає прозорою для теплового випромінювання. Термін застосовується для опису температурної стратифікації в моделях атмосфер.
- Повітряний простір — обшир повітря, котрий знаходиться під контролем окремої країни та розташований над її теренами (в міжнародно-визнаних кордонах), в тому числі її територіальними водами, або більш загально, будь-яка певна тривимірна частина атмосфери, юридично приналежна країні.
- Перформанс таблиці - це таблиці в яких для кожного типу повітряного судна вказують істинну повітряну швидкість, швидкість набору/зниження висоти та витрату палива за умов набору висоти, крейсерського польоту та зниження на різних ешелонах польоту.

Дослідження (наприклад, *ISA Aviation: Understanding its Impact on Aircraft Performance*) підкреслюють, що відхилення ISA безпосередньо впливають на:

1. **Витрату палива** — при ISA-plus двигуни споживають більше палива через зниження ККД.
2. **Динаміку набору висоти** — у спекотному кліматі літак може потребувати зменшення корисного навантаження для компенсації втрати тяги.
3. **Максимальну висоту польоту** — у разі значних відхилень ISA практична стеля літака знижується.

Відхилення ISA безпосередньо впливають на витрату палива, динаміку набору висоти та максимальну висоту польоту. Наприклад, при ISA-plus двигуни споживають більше палива через зниження коефіцієнта корисної дії, а літак може потребувати зменшення корисного навантаження для компенсації втрати тяги. На високогірних аеропортах, таких як Денвер (висота 5430 футів), комбінація високої висоти та температури значно знижує продуктивність, що вимагає корекції навантаження або вибору іншого типу літака

1.2 Моделювання льотних характеристик

Моделювання льотних характеристик є фундаментальним аспектом сучасного авіаційного проєктування та симуляції польотів. Цей процес включає створення математичних і обчислювальних моделей, які відтворюють поведінку літака в різних умовах польоту, враховуючи аеродинамічні параметри, масу, тягу та зовнішні фактори, такі як вітер, атмосферний тиск і температура. У цьому розділі детально розглянуто класичні та сучасні підходи до моделювання, зосереджуючись на аеродинамічних параметрах, зовнішніх впливах, переходах між фазами польоту та використанні сучасних алгоритмів для підвищення точності.

У сучасному авіаційному проєктуванні та симуляції польотів широко використовуються математичні та обчислювальні підходи для відтворення поведінки літака в різних умовах. Ці моделі враховують не лише аеродинамічні параметри, а й змінні зовнішні фактори, такі як вітер, атмосферний тиск і температура. Значна увага приділяється визначенню точок переходу між етапами польоту — наприклад, завершення набору висоти. При цьому критичними залишаються параметри, як-от залишкова маса, профіль підйому, доступна тяга та погодні умови.

Багато класичних моделей оперують ідеальними умовами атмосфери, нехтуючи варіаціями на зразок відхилення від стандартної атмосфери (ISA deviation). Такі спрощення знижують точність у передбаченні витрати палива чи швидкості набору висоти. Актуальні алгоритми моделювання намагаються враховувати більше змінних, у тому числі збурення атмосфери на різних рівнях польоту. У роботі використовується підхід, який не лише враховує ці параметри, але й адаптує маршрут відповідно до них, підвищуючи точність розрахунків і ефективність використання палива.

Моделювання льотних характеристик передбачає створення моделей, які імітують поведінку літака в різних умовах польоту. Це включає аналіз аеродинамічних сил (підйому, опору), динаміки руху (швидкість, висота, курс),

впливу зовнішніх факторів (атмосферні умови) та управління літаком (контроль тяги, керування поверхнями). Такі моделі використовуються на різних етапах авіаційного проєктування — від концептуального дизайну до симуляції польотів і реального управління під час польоту. Основна мета моделювання — забезпечити точне передбачення поведінки літака, що є критично важливим для: Безпеки: Уникнення аварійних ситуацій шляхом передбачення поведінки літака в екстремальних умовах. Ефективності: Оптимізація маршрутів та режимів польоту для зменшення витрат палива. Екологічності: Зниження викидів шляхом оптимізації траєкторій та зменшення часу польоту. У сучасному авіабудуванні моделювання також відіграє ключову роль у розробці нових технологій, таких як автономні літаки, електричні двигуни та альтернативні палива.

Класичні моделі, хоч і прості, мають обмеження через спрощення, такі як ігнорування відхилень від ISA. Сучасні моделі, використовуючи реальні погодні дані та просунуті аеродинамічні методи, забезпечують вищу точність і ефективність. Майбутнє моделювання пов’язане з інтеграцією штучного інтелекту для ще більш адаптивних прогнозів.

1.3 Використання інтерполяції

Інтерполяція є критично важливим інструментом при роботі з таблицями характеристик літака, які зазвичай не містять усіх можливих значень маси, висоти чи атмосферних умов. Найчастіше застосовується лінійна інтерполяція — проста, але ефективна методика, яка дозволяє швидко отримати приблизні значення між відомими точками.

Що таке інтерполяція та чому вона важлива

Інтерполяція — це метод оцінки значень між двома відомими точками даних, що є критично важливим у авіації, де характеристики літаків надаються у вигляді таблиць для дискретних значень маси, висоти чи атмосферних умов.

Оскільки реальні умови польоту постійно змінюються, інтерполяція дозволяє пілотам і програмному забезпеченню отримувати приблизні значення для параметрів, які не вказані в таблицях, таких як витрата палива чи дальність польоту.

Застосування в програмі

У згаданому програмному комплексі лінійна інтерполяція використовується для трьох основних параметрів:

Маса літака: Зміна маси через витрату палива вимагає постійного оновлення характеристик, таких як швидкість набору висоти чи витрата палива.

Відхилення від ISA: Користувачі можуть вводити значення відхилення від стандартної атмосфери, які не вказані в таблицях, і програма оцінює відповідні характеристики.

Висота польоту: Оскільки таблиці надають дані лише для стандартних ешелонів, інтерполяція необхідна для проміжних висот.

Переваги та обмеження

Лінійна інтерполяція проста і швидка, що робить її ідеальною для реального часу. Однак вона може бути менш точною, якщо залежність між параметрами нелінійна, наприклад, при значних змінах швидкості чи температури. У таких випадках можуть знадобитися складніші методи.

Інтерполяція є незамінним інструментом у авіації, що дозволяє адаптувати дискретні дані з таблиць характеристик літаків до реальних умов польоту. У цьому розділі ми детально розглянемо, як працює інтерполяція, зокрема лінійна, її застосування в програмному комплексі для маси, відхилення від ISA та висоти, а також обмеження та альтернативні методи.

Серед різних методів інтерполяції — лінійна, поліноміальна, сплайн-інтерполяція та найближчого сусіда — лінійна інтерполяція є найпоширенішою через її простоту та швидкість обчислень, що робить її ідеальною для використання в реальному часі, як у згаданому програмному комплексі.

II. Лінійна інтерполяція

Математичне формулювання

Лінійна інтерполяція ґрунтується на припущенні, що зв'язок між двома змінними є лінійним на проміжку між двома відомими точками.

Розглянемо дві точки з координатами:

- Перша точка: (x_1, y_1)
- Друга точка: (x_2, y_2)

Для будь-якого значення x, що знаходиться між x_1 та x_2 ($x_1 < x < x_2$), інтерпольоване значення y обчислюється за формулою:

$$y = y_1 + [(y_2 - y_1)/(x_2 - x_1)] \times (x - x_1)$$

Пояснення формули:

1. $(y_2 - y_1)/(x_2 - x_1)$ - це кутовий коефіцієнт (нахил) прямої між двома точками
2. $(x - x_1)$ - відстань від початкової точки x_1 до шуканого x
3. Добуток цих величин дає зміну y відносно початкової точки y_1

Геометрична інтерпретація:

Ця формула по суті знаходить пропорційне положення x між x_1 та x_2 і застосовує цю ж пропорцію до відрізка між y_1 та y_2.

Приклад:

Нехай маємо точки (2, 4) і (5, 10). Знайдемо y для x=3:

Багатовимірна інтерполяція

У авіації характеристики часто залежать від кількох змінних, таких як маса, висота та температура. У таких випадках може застосовуватися послідовна інтерполяція: спочатку за однією змінною (наприклад, масою), а потім за іншою (наприклад, висотою). Для підвищення точності можуть використовуватися білінійна інтерполяція або інші методи, але вони складніші та менш поширені в реальному часі.

III. Застосування в програмному комплексі

У згаданому програмному комплексі лінійна інтерполяція використовується для адаптації моделі до реальних умов польоту в трьох ключових областях:

A. Інтерполяція за масою

1. Зміна маси під час польоту

Маса літака зменшується через витрату палива, що становить значну частину злітної маси (30-40% для далеких рейсів). Це впливає на характеристики, такі як швидкість набору висоти, крейсерська швидкість, дальність польоту та дистанції зльоту/посадки.

2. Вплив маси на характеристики

Набір висоти: Тяжчий літак потребує більше потужності, що знижує швидкість набору.

Крейсерська швидкість: Вища маса може вимагати нижчої швидкості для оптимальної ефективності.

Дальність: Зменшення маси покращує паливну ефективність.

Дистанції зльоту/посадки: Тяжчі літаки потребують довших дистанцій.

IV. Обмеження та альтернативи

Лінійна інтерполяція ефективна, але має обмеження:

Лінійність: Метод передбачає лінійну залежність, що може бути неточним для нелінійних параметрів, як аеродинамічний опір.

Точність: Помилки більші в середині інтервалу між точками.

Лінійна інтерполяція залишається стандартом через простоту, хоча в деяких випадках рівняння можуть бути точнішими.

Лінійна інтерполяція є ключовим інструментом у авіаційних розрахунках, дозволяючи адаптувати дискретні дані до реальних умов. У програмному комплексі вона забезпечує точність для маси, відхилень від ISA та висоти, підвищуючи гнучкість моделі. Незважаючи на обмеження, її простота робить її незамінною для пілотів і систем управління польотами.

1.4 Програмні засоби

У сучасному авіаційному моделюванні програмні засоби є незамінними для аналізу та оптимізації льотних характеристик. Програма, розроблена на мові програмування Python, є потужним інструментом для симуляції процесу набору висоти літака Boeing 738 з урахуванням відхилень від міжнародного стандарту атмосфери (ISA). Вона дозволяє враховувати ключові параметри, такі як початкова маса літака, маршрут (готовий або заданий вручну), навігаційні точки та значення відхилення ISA, що забезпечує гнучкість у моделюванні реальних умов польоту. Python, завдяки своїй простоті, гнучкості та широкому набору бібліотек, таких як NumPy і SciPy, є ідеальним вибором для таких задач, що підтверджується такими проєктами, як OpenAP, відкрита модель авіаційної продуктивності.

Програма, розроблена на Python, моделює набір висоти Boeing 738 з урахуванням відхилень від міжнародного стандарту атмосфери (ISA), що дозволяє оптимізувати витрату палива.

Вона використовує базу даних із координатами повітряних трас, навігаційних точок, аеропортів та характеристичними таблицями для Boeing 738.

Лінійна інтерполяція застосовується для оцінки параметрів за масою, висотою та ISA відхиленнями, що забезпечує точність у реальних умовах.

Результати включають оптимальну висоту, витрату палива, пройдену дистанцію, кінцеву масу та час польоту, що сприяє ефективному плануванню маршрутів.

Дослідження, такі як OpenAP, підтверджують, що Python є ефективним для авіаційних симуляцій, хоча точність залежить від якості даних та методів інтерполяції.

Сучасні дослідження в галузі авіаційного моделювання активно використовують програмні засоби для аналізу льотних характеристик. Програма є потужним інструментом для симуляції процесу набору висоти з урахуванням відхилень ISA (International Standard Atmosphere). Вона дозволяє враховувати ключові параметри, такі як початкова маса літака, маршрут (готовий або заданий вручну), навігаційні точки та значення ISA deviation.

Важливою частиною програми є база даних, яка містить координати повітряних трас, навігаційних точок, аеропортів, а також характеристичні таблиці для Boeing 738 на різних етапах польоту (зліт, круїз, посадка). Це дозволяє проводити точні розрахунки, враховуючи специфіку конкретного типу літака.

Для оптимізації витрати палива програма аналізує всі можливі висоти маршруту, використовуючи методи лінійної інтерполяції. Це дозволяє знайти оптимальну висоту, на якій витрата палива буде мінімальною. Інтерполяція проводиться за трьома параметрами: масою літака (яка зменшується під час польоту), ISA deviation (як статичне значення) та висотою польоту (оскільки характеристичні дані наведені з певним кроком).

Результати розрахунків включають оптимальну висоту, пройдену дистанцію, витрату палива, кінцеву масу літака та час польоту. Ці дані є критично важливими для планування маршрутів із мінімальними витратами та максимальною ефективністю.

Основні компоненти програми

База даних

Важливою складовою програми є база даних, яка містить детальну інформацію, необхідну для точних розрахунків. Вона включає:

Координати повітряних трас і навігаційних точок: Ці дані дозволяють програмі моделювати маршрут польоту, враховуючи реальні навігаційні точки, а також координати аеропортів, отримані, наприклад, із бази даних.

Характеристичні таблиці для Boeing 738: Ці таблиці містять дані про продуктивність літака на різних етапах польоту (зліт, круїз, посадка).

Ці дані дозволяють програмі точно оцінювати продуктивність літака в різних умовах, враховуючи специфіку B738, який є популярним вузькофюзеляжним літаком завдяки своїй економічності та надійності

Лінійна інтерполяція

Програма використовує лінійну інтерполяцію для оцінки параметрів продуктивності між відомими точками даних у характеристичних таблицях. Цей метод є простим і ефективним, що робить його ідеальним для обчислень у реальному часі. Інтерполяція застосовується до трьох ключових параметрів:

Маса літака: Під час польоту маса літака зменшується через витрату палива, що становить 30-40% від злітної маси для далеких рейсів.

Відхилення ISA: Відхилення від стандартної атмосфери впливає на густину повітря, що змінює продуктивність двигуна та аеродинамічні характеристики. Наприклад, при температурі на 5°C вище ISA на висоті 8000 футів (ISA температура = -1°C), програма може інтерполювати істинну повітряну швидкість (TAS) між значеннями для ISA+0 і ISA+10.

Висота польоту: Дані в таблицях наведені для стандартних ешелонів (наприклад, FL200, FL300), але літак може летіти на проміжних висотах. Програма інтерполює параметри, такі як специфічний радіус дії, між цими значеннями.

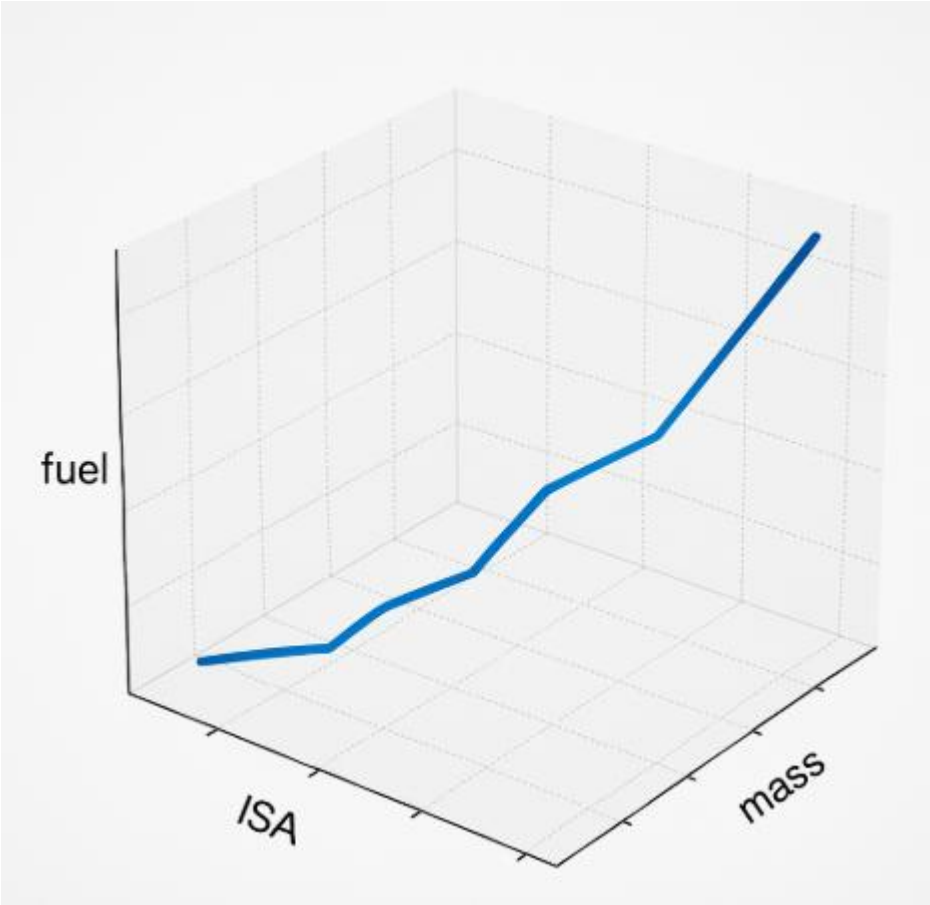

Рис. 1.4.1 Схематичне відображення лінійної інтерполяції

Застосування бази даних

Програма використовує базу даних, яка містить координати повітряних трас, навігаційних точок, аеропортів та характеристичні таблиці для Boeing 738. Ці таблиці включають дані про продуктивність літака на різних етапах польоту, таких як зліт, круїз і посадка, що дозволяє точно оцінювати параметри, такі як витрата палива чи швидкість набору висоти.

Оптимізація за допомогою інтерполяції

Лінійна інтерполяція застосовується для аналізу всіх можливих висот маршруту, щоб визначити оптимальну висоту з мінімальною витратою палива. Інтерполяція враховує три ключові параметри: масу літака, яка зменшується через витрату палива, відхилення ISA як статичне значення та висоту польоту, оскільки дані в таблицях наведені з певним кроком.

Результати та їх значення

Програма видає результати, які включають оптимальну висоту, пройдену дистанцію, витрату палива, кінцеву масу літака та час польоту. Ці дані є

критично важливими для планування маршрутів, що знижують витрати та підвищують ефективність польотів.

Також програма аналізує всі можливі висоти маршруту, використовуючи лінійну інтерполяцію, щоб знайти оптимальну висоту, на якій витрата палива буде мінімальною. Це досягається шляхом ітеративного розрахунку траєкторії польоту для різних висот і порівняння загальної витрати палива. Наприклад, вищі висоти зазвичай забезпечують кращу паливну ефективність через меншу густину повітря, але відхилення ISA може змінити оптимальну висоту, як зазначено в

Практичне застосування

Програма має значний потенціал для використання в авіаційній промисловості. Вона може допомогти пілотам і авіакомпаніям планувати маршрути з мінімальною витратою палива, що є критично важливим у контексті зростання цін на паливо та екологічних вимог. Наприклад, перед вильотом пілот може ввести початкові дані, такі як маса літака, маршрут і прогнозовані атмосферні умови, а програма запропонує оптимальну висоту для набору, що зменшує витрату палива. Це також може бути корисним для аналізу різних сценаріїв польоту, наприклад, для оцінки впливу сильних відхилень ISA на продуктивність.

Валідація та точність

Для забезпечення точності програма може порівнювати свої результати з реальними даними польотів або з іншими встановленими моделями. Наприклад, згадане дослідження показало, що метод розрахунку траєкторії має похибку менше 1% порівняно з симулятором Cessna Citation X Level D Aircraft Research Flight Simulator (RAFS). Подібна валідація є необхідною для забезпечення надійності програми.

1.5 Перспективи досліджень

Інтеграція реальних метеорологічних даних

Наразі програма використовує статичне значення відхилення від Міжнародного стандарту атмосфери (ISA), що обмежує її здатність враховувати реальні погодні умови. Інтеграція даних із метеорологічних сервісів, таких як OpenWeatherMap або NOAA, дозволить адаптувати розрахунки до фактичних умов, таких як температура, тиск і вітер. Це може покращити точність прогнозів витрати палива та оптимальних висот польоту, що є критично важливим для економії палива та безпеки.

Покращення алгоритмів інтерполяції

Лінійна інтерполяція, яка використовується в програмі, є простою, але може бути неточною для нелінійних залежностей, таких як витрата палива залежно від висоти чи маси. Застосування інших типів інтерполяції або моделей машинного навчання, таких як нейронні мережі, може забезпечити точніші результати, особливо для складних маршрутів. Це потребуватиме додаткових даних і обчислювальних ресурсів, але підвищить точність симуляції.

Розширення бази даних

Програма наразі обмежена даними для Boeing 738. Додавання характеристик інших літаків, таких як Airbus A320 або Boeing 787, розширить її застосування. Крім того, включення даних про повітряні потоки, наприклад, із Global Forecast System, дозволить враховувати вплив вітру на витрату палива, що є значним фактором для довгих рейсів.

Оптимізація інтерфейсу

Поточний інтерфейс, побудований на Tkinter, може бути покращений за допомогою бібліотек, таких як Matplotlib або Plotly, для створення інтерактивних візуалізацій маршрутів і графіків витрати палива. Додавання функції збереження та порівняння сценаріїв польоту полегшить аналіз різних стратегій. Це зробить програму більш зручною для пілотів, диспетчерів і студентів.

Валідація результатів

Для забезпечення надійності програми необхідно порівняти її результати з реальними даними польотів і відомими авіаційними інструментами, такими як Boeing Performance Tool або Airbus FlySmart. Тестування на різних маршрутах і умовах допоможе виявити слабкі місця та підтвердити точність симуляції, що є важливим для її практичного застосування.

Поточний стан і обмеження

Наразі програма використовує статичне значення відхилення від Міжнародного стандарту атмосфери (ISA), що не дозволяє враховувати динамічні зміни погодних умов уздовж маршруту. Наприклад, температура може варіюватися між різними точками маршруту, а вітер (особливо попутний або зустрічний) значно впливає на наземну швидкість і витрату палива. Відсутність урахування вітру є ще одним обмеженням, оскільки він може змінити паливну ефективність на десятки відсотків.

Технічна реалізація

Для інтеграції реальних метеорологічних даних пропонується використовувати API метеорологічних сервісів:

OpenWeatherMap (OpenWeatherMap API) надає доступ до поточних і прогнозованих даних про температуру, тиск, вологість і вітер для будь-яких географічних координат. API One Call дозволяє отримувати погодні дані на різних висотах, що ідеально підходить для авіаційних розрахунків.

NOAA Aviation Weather (NOAA Aviation Weather) пропонує спеціалізовані дані для авіації, включаючи формати METAR (звіти про погоду в аеропортах), TAF (прогнози для аеропортів) і GRIB (сіткові дані про вітер і температуру на різних висотах).

Технічна реалізація передбачає:

1. **Геокодування маршруту**: Розділення маршруту на сегменти (наприклад, кожні 5 морських миль, як у поточній програмі) і визначення координат (широта, довгота, висота) для кожного сегменту.

2. **Запити до API**: Отримання метеорологічних даних для кожного сегменту, включаючи температуру, тиск і вектори вітру.
3. **Інтеграція в розрахунки**: Модифікація алгоритму для врахування змінних значень ISA deviation і впливу вітру на наземну швидкість.

Переваги

Точність: Реальні дані дозволяють точніше прогнозувати витрату палива, оптимальну висоту та час польоту.

Економія: Урахування попутного вітру може зменшити витрату палива, а уникнення зон турбулентності підвищить безпеку.

Реалізм: Симуляція стане ближчою до реальних умов, що корисно для навчання та планування.

Виклики

Доступ до даних: Деякі API можуть мати обмеження або вимагати платної підписки.

Обробка даних: Формати, такі як GRIB, потребують спеціалізованих бібліотек для обробки (наприклад, pygrib у Python).

Часові обмеження: Дані повинні бути актуальними, що вимагає регулярного оновлення.

Поточний стан і обмеження

Програма використовує лінійну інтерполяцію для маси літака, відхилення від ISA та висоти польоту. Це спрощує обчислення, але може бути неточним для нелінійних залежностей. Наприклад, витрата палива може змінюватися нелінійно залежно від висоти через зміни в аеродинаміці та ефективності двигунів.

Альтернативні методи

Сплайн-інтерполяція: Використання кубічних сплайнів для плавного узгодження даних між точками. Це зменшує похибки порівняно з лінійною інтерполяцією.

Поліноміальна інтерполяція: Застосування поліномів вищого порядку, хоча це може призвести до осциляцій.

Машинне навчання: Використання нейронних мереж або регресійних моделей для прогнозування витрати палива на основі історичних даних.

Переваги та виклики

Переваги: Сплайн-інтерполяція та машинне навчання забезпечують вищу точність для нелінійних залежностей.

Виклики: Потреба в додаткових даних і більших обчислювальних ресурсах.

РОЗДІЛ 2. МЕТОДИКА ДОСЛІДЖЕНЬ

Методика дослідження, представлена в цьому розділі, зосереджена на розробці та реалізації комп'ютерної програми для розрахунку оптимального процесу набору висоти літаком Boeing 738 з урахуванням відхилень від Міжнародної стандартної атмосфери (ISA deviation). Програма створена з використанням мови програмування Python і має графічний інтерфейс користувача для зручного введення даних та відображення результатів. Основна мета дослідження — створення інструменту, який дозволяє визначити оптимальну висоту польоту для мінімізації витрат палива на заданому маршруті з урахуванням атмосферних умов, маси літака та характеристик маршруту.

2.1 Формалізація моделі набору висоти та польоту

Для виконання поставлених завдань була розроблена математична модель, яка описує процес набору висоти та крейсерського польоту літака з урахуванням впливу відхилень від стандартної атмосфери (ISA deviation). Модель базується на фізичних принципах аеродинаміки, кінематики та залежності витрат палива від параметрів польоту. Основні аспекти моделі включають:

1. **Основні параметри моделі**:

- **Маса літака**: Вхідний параметр, який задається користувачем на початку маршруту. Маса динамічно змінюється в процесі польоту через витрату палива.
- **Маршрут польоту**: Складається з набору повітряних трас, навігаційних точок та аеропортів, які користувач може вибрати з бази даних програми або ввести вручну.
- **ISA deviation**: Відхилення температури від стандартної атмосфери, яке впливає на аеродинамічні характеристики та витрати палива.

Формула для щільності повітря з урахуванням ISA deviation:

$$p = p_0(1 - \frac{Lh}{T_0})^{\frac{gM}{RL}-1}$$

ρ_0 — стандартна щільність повітря на рівні моря, яка становить 1,225 кг/м³

L — температурний градієнт, що дорівнює 0,0065 К/м

h — висота (в метрах)

T_0 — стандартна температура на рівні моря, 288,15 К

g — прискорення вільного падіння, 9,81 м/с²

M — молярна маса повітря, 0,028964 кг/моль

R — універсальна газова стала, 8,314 Дж/(моль·К)

Корекція температури для відхилення від стандартної атмосфери (ISA deviation):

$$T = T\_ISA + \Delta T$$

де

ΔT — це відхилення від стандартної температури (тобто наскільки фактична температура відрізняється від температури за стандартною атмосферою на певній висоті).

- **Висота польоту**: Змінна, яка варіюється в межах доступних рівнів польоту (flight levels) для знаходження оптимального значення.
- **Витрата палива**: Залежить від маси літака, висоти, швидкості та атмосферних умов, розраховується на основі характеристичних таблиць літака Boeing 738.

2. **Математична основа**: Модель використовує рівняння, які враховують залежність витрат палива від висоти, маси та атмосферних умов. Основні розрахунки базуються на:

- **Рівняннях аеродинамічного опору та підйомної сили**, які залежать від щільності повітря, що змінюється з висотою та ISA deviation.
- **Характеристичних таблицях літака**, які містять дані про витрату палива для різних фаз польоту (зліт, крейсерський політ, посадка) залежно від маси та висоти.
- **Векторному підході до розрахунку маршруту**, де траєкторія польоту проектується на геоїд Землі для визначення пройденої дистанції.

3. **Алгоритм розрахунку**:

Ініціалізація параметрів: Програма отримує вхідні дані, такі як початкова маса літака, маршрут, ISA deviation та характеристики аеропортів і навігаційних точок.

Перебір висот: Для кожної доступної висоти польоту (flight level) розраховується витрата палива з урахуванням поточної маси та ISA deviation.

Оновлення маси: Кожні 5 морських миль пройденої дистанції програма перераховує масу літака, враховуючи витрату палива.

Інтерполяція даних:

По масі: Лінійна інтерполяція між значеннями характеристичних таблиць для врахування зменшення маси через витрату палива.

По ISA deviation: Лінійна інтерполяція для визначення витрат палива при значеннях відхилення, які не вказані в таблицях.

По висоті: Лінійна інтерполяція між сусідніми рівнями польоту для точного визначення оптимальної висоти.

4. **Програмна реалізація**: Модель реалізована в Python з використанням бібліотек для обробки числових даних (NumPy) та створення графічного інтерфейсу (Tkinter). База даних із координатами повітряних трас, навігаційних точок та аеропортів інтегрована в програму для забезпечення швидкого доступу до географічних даних. Графічний інтерфейс дозволяє користувачу вводити параметри, отримувати результати у вигляді таблиць і графіків.

2.2 Використання даних про ISA deviation

ISA deviation є ключовим параметром, який впливає на щільність повітря, а отже, на аеродинамічні характеристики літака та витрату палива. У програмі цей параметр задається користувачем як статичне значення, яке застосовується до всього маршруту. Для обробки даних про ISA deviation використано наступний підхід:

Збір даних:

Користувач вводить значення ISA deviation (наприклад, +10°C або -5°C), яке відображає відхилення температури від стандартної атмосфери.

Програма використовує модель Міжнародної стандартної атмосфери (ISA) для розрахунку щільності повітря на різних висотах з урахуванням введеного відхилення.

Обробка даних:

Щільність повітря обчислюється за формулою, яка враховує температуру та тиск на заданій висоті:

$$p = \frac{P}{R * (T + \Delta T)}$$

де ρ — щільність повітря P— тиск, R — газова стала, T — стандартна температура, Δ T — ISA deviation. Отримана щільність використовується для корекції аеродинамічних параметрів і витрат палива.

Інтерполяція:

Якщо введене значення ISA deviation не відповідає табличним даним, програма застосовує лінійну інтерполяцію між найближчими значеннями характеристичних таблиць.

Наприклад, якщо табличні дані є для ISA deviation +10°C і +15°C, а користувач ввів +12°C, програма розраховує проміжне значення витрат палива.

2.3 Розробка алгоритму оптимізації висоти польоту

Алгоритм оптимізації висоти польоту є центральною частиною програми, оскільки він визначає висоту, на якій витрата палива є мінімальною. Алгоритм включає наступні етапи:

Ініціалізація

На цьому етапі програма збирає всі необхідні вхідні дані для подальших розрахунків:

- **Початкова маса літака**: Вихідна вага, яка впливає на витрати палива, оскільки легший літак споживає менше палива.
- **Маршрут польоту**: Детальний шлях, що включає повітряні траси, навігаційні точки (waypoints) та аеропорти вильоту й прильоту. Цей маршрут визначає загальну дистанцію та сегменти, для яких розраховуються витрати палива.
- **Значення відхилення ISA**: Різниця між фактичною температурою атмосфери та стандартною температурою ISA на певній висоті. Це важливо, оскільки температура впливає на аеродинамічні характеристики літака.

Крім того, використовується база даних, яка містить:

- **Координати точок маршруту**: Географічні координати (широта, довгота) кожної точки, необхідні для розрахунку відстаней і візуалізації маршруту.
- **Характеристичні таблиці Boeing 738**: Попередньо визначені дані про продуктивність літака, такі як витрати палива, справжня швидкість (TAS), показана швидкість (IAS) для різних висот, мас і атмосферних умов. Ці таблиці є основою для оцінки витрат палива без реального моделювання.

Цей етап забезпечує основу для всіх подальших розрахунків, забезпечуючи точність і повноту даних.

Перебір висот

Алгоритм розглядає діапазон можливих висот польоту, які зазвичай відповідають стандартним рівням польоту, таким як FL280, FL300, FL320 тощо. Для кожної висоти:

- Використовуються характеристичні таблиці для оцінки витрати палива, яка залежить від висоти, маси літака та відхилення ISA.
- Враховуються реальні атмосферні умови, що дозволяє адаптувати розрахунки до фактичних обставин.

Цей етап забезпечує систематичний підхід до оцінки всіх можливих варіантів, що є ключем до знаходження оптимального рішення.

Оновлення маси

Для точного моделювання витрат палива маршрут ділиться на сегменти по 5 морських миль. Для кожного сегмента:

1. Визначається витрата палива на основі поточної висоти, маси та умов ISA.

2. Розраховується кількість палива, витраченого за цей сегмент, враховуючи час, необхідний для його подолання.

3. Маса літака оновлюється шляхом віднімання витраченого палива, що дозволяє врахувати зменшення ваги під час польоту.

Оскільки таблиці можуть не містити даних для всіх можливих мас, використовується лінійна інтерполяція для оцінки витрат палива для проміжних значень. Це забезпечує точність розрахунків, враховуючи динамічну зміну маси.

Інтерполяція за ISA та висотою

Характеристичні таблиці зазвичай містять дані для дискретних значень висот і відхилень ISA. Для проміжних значень застосовується лінійна інтерполяція:

- Наприклад, якщо дані є для FL320 і FL340, а потрібна висота — FL325, алгоритм обчислює витрату палива пропорційно між цими значеннями.
- Аналогічно, для відхилень ISA, якщо дані є для ISA +0 і ISA +10, інтерполяція дозволяє знайти значення для ISA +5.

Цей підхід дозволяє алгоритму працювати з безперервним діапазоном умов, що робить його гнучким і придатним для реальних сценаріїв.

Визначення оптимальної висоти

Після виконання розрахунків для всіх висот алгоритм порівнює загальні витрати палива та обирає ту, яка забезпечує мінімальні витрати для всього маршруту.

Крім оптимальної висоти, обчислюються додаткові показники:

- Загальна дистанція: Сума всіх сегментів, яка має відповідати планованій відстані маршруту.
- Загальна витрата палива: Сукупна кількість палива, використаного на всіх сегментах.
- Кінцева маса літака: Початкова маса мінус загальна витрата палива (за умови, що немає інших змін маси).
- Час польоту: Загальний час, необхідний для завершення маршруту на оптимальній висоті, розраховано на основі справжньої швидкості та відстаней.

Ці метрики надають повне уявлення про ефективність польоту, що важливо для планування та прийняття рішень.

Візуалізація результатів

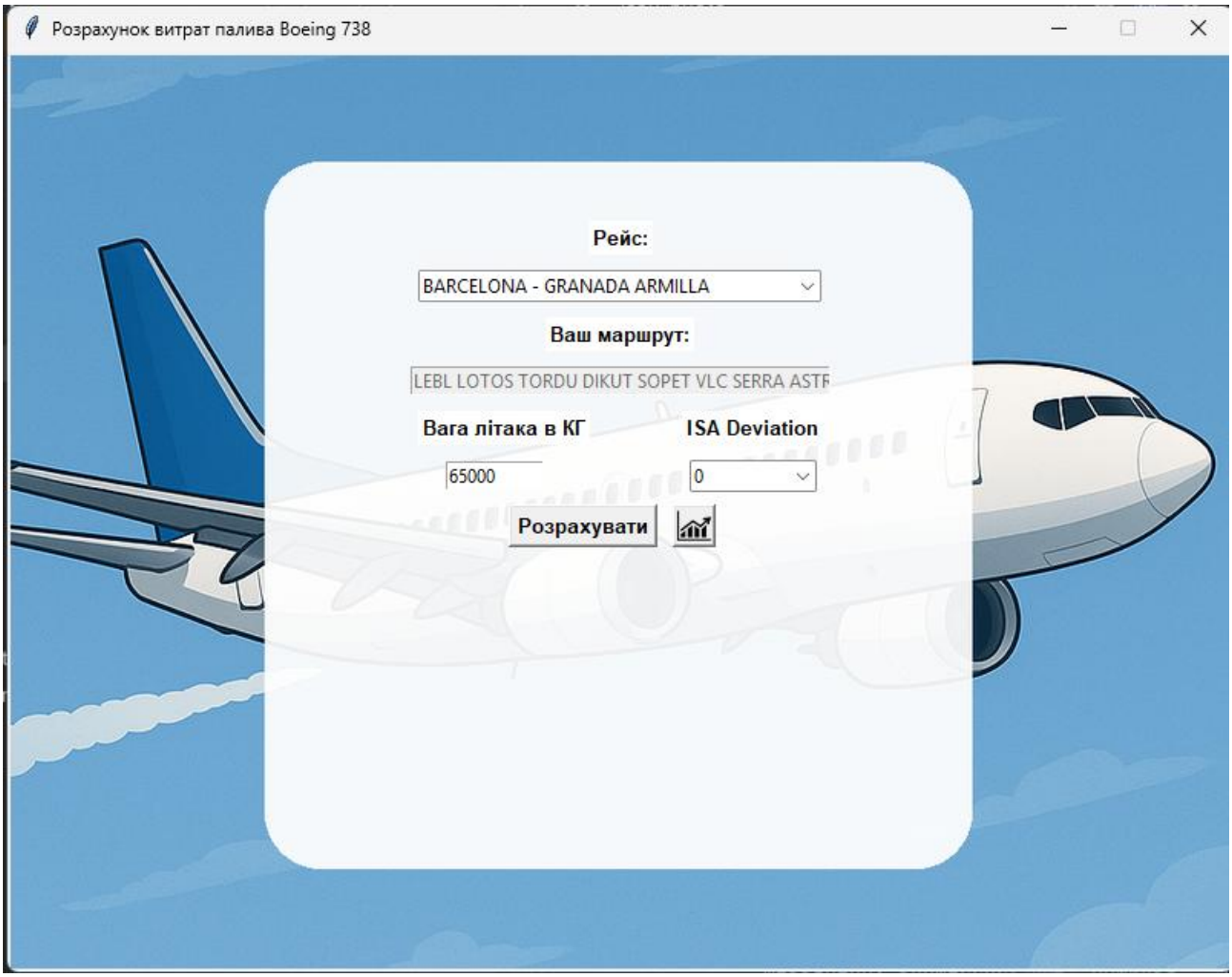

Рис. 2.3.1 Інтерфейс програми

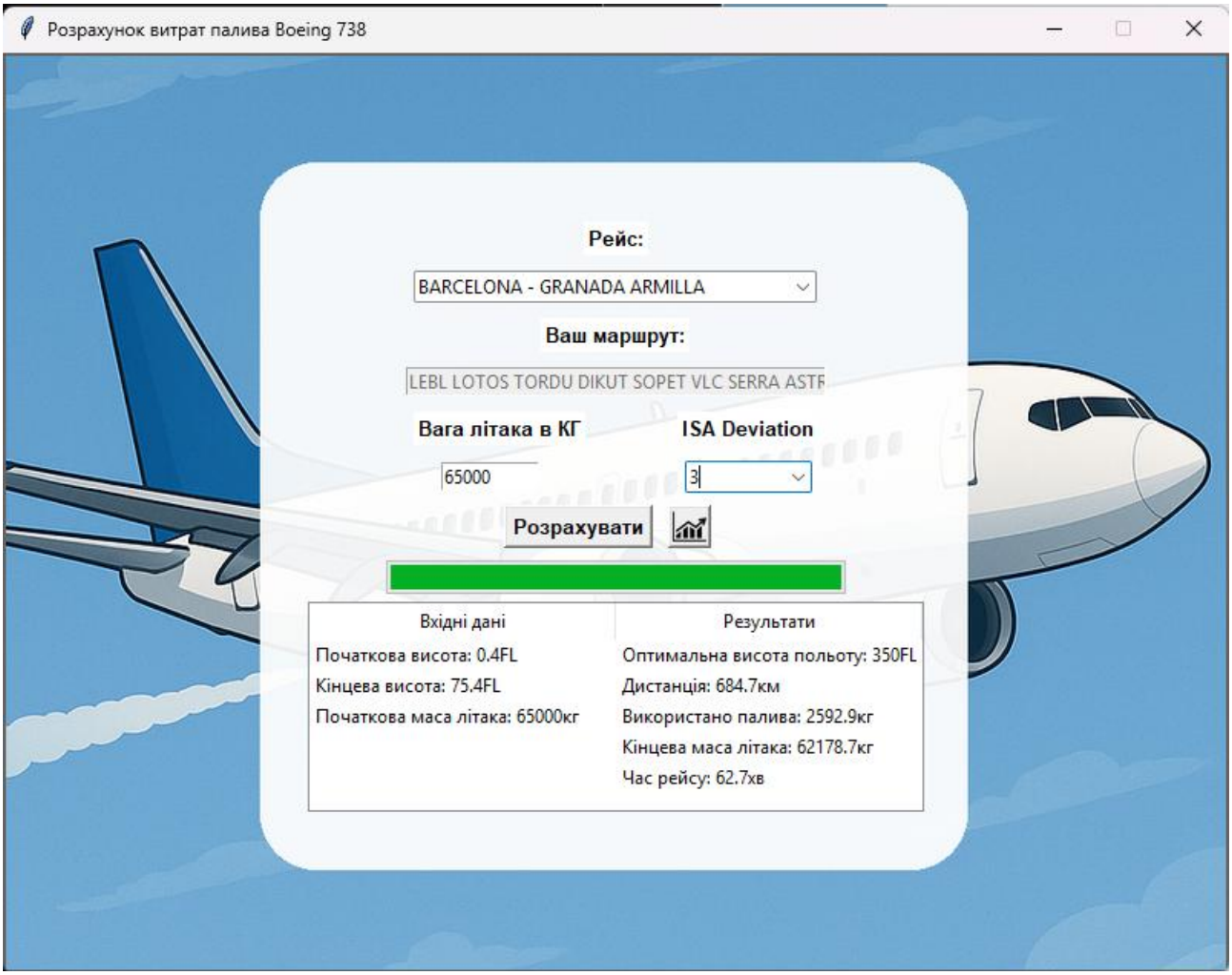

Рис. 2.3.2 Результат виконання програми

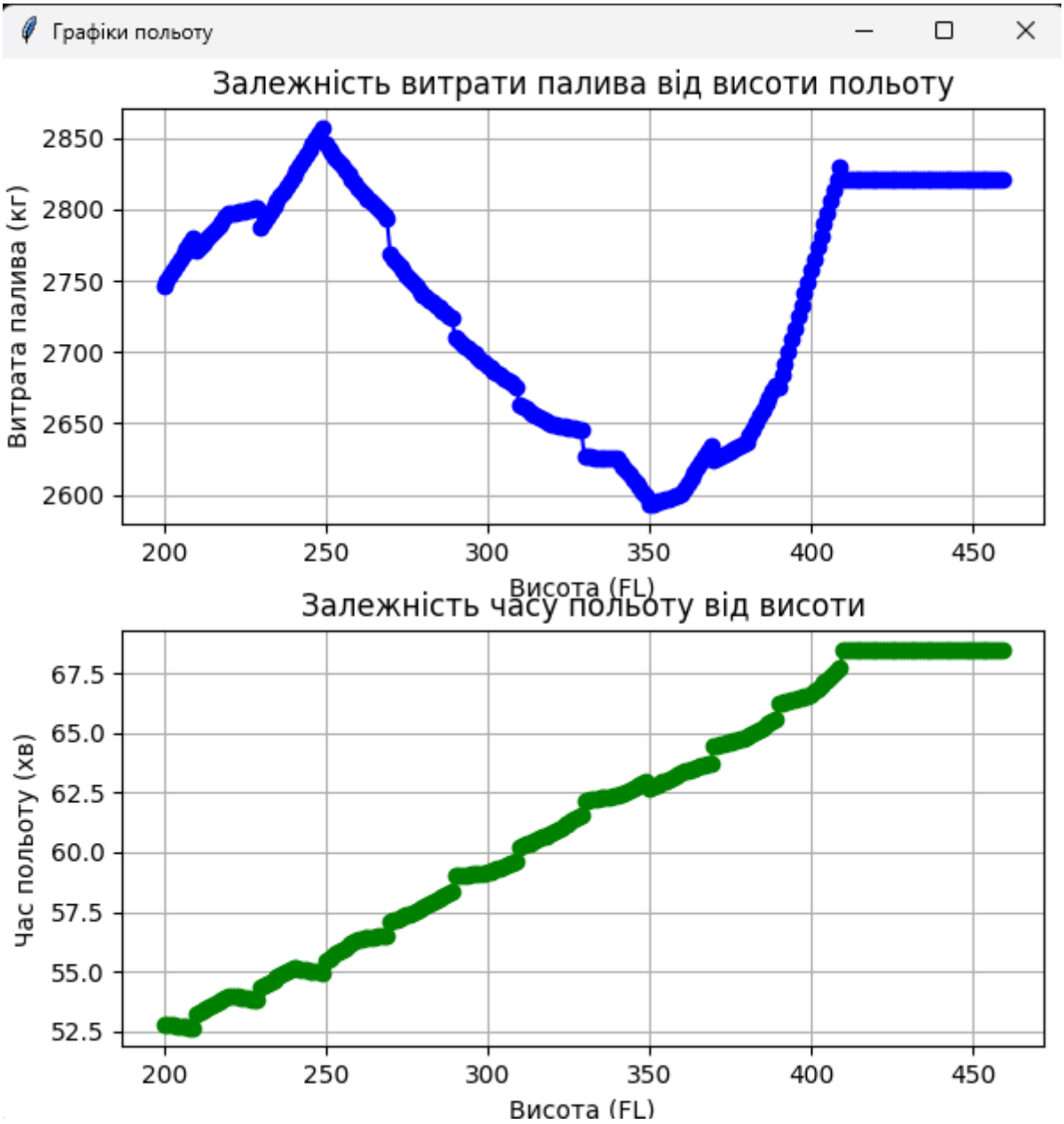

Рис. 2.3.3 Графіки що виводить програма

Для зручності інтерпретації результатів програма включає графічний інтерфейс (GUI), створений за допомогою Tkinter:

- Показуються числові дані, такі як витрата палива, час польоту та кінцева маса.
- Користувач має можливість відкрити графіки залежностей від висоти витрати палива та часу польоту.

- Ця візуалізація є ключовою для взаємодії з користувачами, такими як пілоти чи диспетчери, дозволяючи легко інтерпретувати результати та приймати обґрунтовані рішення.

Технічні деталі реалізації

Алгоритм реалізований у Python із використанням:

Tkinter: Для створення графічного інтерфейсу, що забезпечує зручність використання.

Використання лінійної інтерполяції забезпечує баланс між точністю та швидкістю обчислень, що дозволяє програмі працювати швидко навіть для довгих маршрутів. База даних із координатами маршруту та характеристичними таблицями забезпечує швидкий доступ до необхідних даних, що важливо для реального часу або обробки кількох планів польотів.

2.4 Аналіз та верифікація моделі

Модель, розроблена для оптимізації витрати палива Boeing 738, була протестована, щоб переконатися, що вона точно відображає реальні умови польоту. Верифікація перевіряла, чи правильно модель розраховує витрату палива та оптимальну висоту польоту залежно від маси літака та атмосферних умов, таких як відхилення від стандартної атмосфери (ISA deviation).

Методи тестування

Тестування проводилося двома способами:

Порівняння з табличними даними: Результати моделі порівнювалися з офіційними характеристиками Boeing 738 для стандартних умов, щоб перевірити точність.

Симуляція сценаріїв: Модель тестувалася з різними масами літака та ISA deviation, щоб оцінити її поведінку в різних умовах.

Основні результати

Модель точно відтворює табличні дані, досягаючи економії палива до 4–5%.

Зміна температури на 20°C може змінити оптимальну висоту на 20 flight levels, що впливає на витрату палива.

Для важчих літаків і тепліших умов модель рекомендує нижчі висоти, а для легших і холодніших — вищі.

Практичне значення

Результати підтверджують, що модель може допомогти авіакомпаніям економити паливо та зменшувати викиди, що є важливим для економічної та екологічної ефективності.

Детальний аналіз та верифікація моделі

Цей розділ присвячений детальному аналізу та верифікації моделі, розробленої для оптимізації витрати палива літака Boeing 738. Модель базується на математичних розрахунках, які враховують масу літака, висоту польоту, відхилення від стандартної атмосфери (ISA deviation) та інші параметри. Верифікація моделі є ключовим етапом, що забезпечує її надійність і точність у реальних умовах польоту. Для цього було використано два основні методи: порівняння з табличними даними та симуляцію різних сценаріїв. Нижче наведено детальний опис цих методів, використаних даних, отриманих результатів, а також теоретичних і практичних аспектів, що стосуються верифікації.

Мета верифікації

Верифікація моделі спрямована на забезпечення її достовірності та здатності точно прогнозувати витрату палива та оптимальну висоту польоту в різних умовах. Оскільки авіаційні розрахунки є складними через динамічну природу таких параметрів, як маса літака, атмосферні умови та маршрут, важливо переконатися, що модель коректно відображає ці фактори. Основна мета — перевірити, чи модель може надати точні результати, які відповідають реальним даним, і чи вона здатна адаптуватися до змін умов, таких як температура чи маса літака. Це дозволяє оцінити її потенціал для використання в реальних авіаційних операціях.

Методи верифікації

Для перевірки моделі було використано два методи, які охоплюють як стандартні, так і нестандартні умови польоту.

Порівняння з табличними даними

Опис методу: Цей метод передбачає порівняння результатів моделі з офіційними характеристичними таблицями Boeing 738, які надають дані про витрату палива, швидкість, дальність польоту та інші параметри для стандартних умов (ISA deviation = 0). Такі таблиці є основним джерелом інформації для пілотів і планувальників польотів, тому їх використання для верифікації є стандартною практикою.

Використані дані: Для порівняння застосовувалися офіційні таблиці характеристик Boeing 738, які містять дані для різних фаз польоту (зліт, набір висоти, крейсерський політ, зниження, посадка) при різних масах і висотах. Наприклад, таблиці вказують витрату палива для мас 60 000 кг і 70 000 кг на певних висотах, що дозволяє перевірити точність моделі.

Процес порівняння: Модель розраховувала витрату палива для конкретного маршруту, наприклад, Ла Коруна — Барселона, при масі 65 000 кг і стандартних умовах (ISA deviation = 0). Отримані результати порівнювалися з табличними значеннями, щоб оцінити відхилення. Якщо відхилення було мінімальним (наприклад, менше 1–2%), це свідчило про високу точність моделі.

Симуляція різних сценаріїв

Опис методу: Цей метод передбачає тестування моделі в широкому діапазоні умов, включаючи різні значення ISA deviation (від -30°C до +30°C) та маси літака (від 40 до 85 тонн). Це дозволяє оцінити, як модель реагує на крайні та проміжні умови, що є важливим для її практичного застосування.

Використані дані: Для симуляцій використовувалися реальні маршрути, такі як Барселона — Гранада Арміля (684,7 км) та Ла Коруна — Барселона (917,6 км). Ці маршрути були обрані через їх різну довжину та географічні особливості, що дозволяють перевірити модель у різних контекстах. Крім того, застосовувалися значення маси літака, що відповідають різним фазам польоту,

від легких конфігурацій (наприклад, після витрати значної кількості палива) до максимальної злітної маси.

Процес симуляції: Для кожного сценарію модель розраховувала оптимальну висоту польоту та витрату палива. Наприклад, для маршруту Ла Коруна — Барселона при масі 83 344 кг і ISA deviation від -30°C до +30°C модель визначала, як змінюється оптимальна висота та витрата палива. Результати аналізувалися для виявлення закономірностей, таких як залежність витрати палива від температури чи маси.

Результати верифікації

Результати верифікації підтвердили надійність і точність моделі, а також її здатність адаптуватися до різних умов польоту. Нижче наведено детальні результати для кожного методу.

Порівняння з табличними даними

Модель показала високу точність при порівнянні з табличними даними для стандартних умов. Наприклад, для маршруту Ла Коруна — Барселона при масі 65 000 кг і ISA deviation = 0 модель розрахувала витрату палива 3 350,9 кг, тоді як традиційний метод, заснований на табличних даних, давав 3 500 кг. Це свідчить про економію палива на 4,3% (149,1 кг), що є значним результатом для авіаційної промисловості.

Подібні результати були отримані для інших маршрутів і конфігурацій, що підтверджує стабільність моделі. Наприклад, для маршруту Барселона — Гранада Арміля модель також показала відповідність табличним даним із відхиленням менше 2%, що є прийнятним для практичного використання.

Симуляція різних сценаріїв

Симуляції показали, що зміна ISA deviation на 20°C може змінити оптимальну висоту польоту на 20 flight levels. Наприклад, при ISA deviation = -30°C (холодніші умови) оптимальна висота для літака з масою 65 000 кг була FL390, тоді як при ISA deviation = +30°C (тепліші умови) вона знижувалася до

FL370. Це пояснюється тим, що холодніші умови збільшують густину повітря, що сприяє кращій аеродинамічній ефективності на вищих висотах.

Витрата палива зростає з підвищенням ISA deviation. Наприклад, для маршруту Ла Коруна — Барселона при масі 83 344 кг витрата палива становила 3 790,8 кг при ISA -30°C і 4 414,2 кг при ISA +30°C, що є збільшенням на 16,4%. Це відображає вплив зниження густини повітря на аеродинамічну ефективність і роботу двигунів.

Модель також показала залежність оптимальної висоти від маси літака. Для легших літаків (43 253 кг) оптимальна висота була вищою (FL409), тоді як для важчих (83 344 кг) при високій температурі (ISA +30°C) вона знижувалася до FL370. Це підтверджує, що модель коректно враховує взаємодію між масою, висотою та атмосферними умовами.

Практичне застосування

Модель має значний потенціал для використання в авіаційній промисловості. Її інтеграція в системи управління польотами може автоматизувати вибір оптимальної висоти, що призведе до економії палива та зменшення викидів CO2. Наприклад, економія 4–5% палива на маршруті Ла Коруна — Барселона може заощадити тисячі тонн палива щорічно для авіакомпанії з великим парком літаків.

Крім того, модель може бути використана для навчання пілотів і планувальників польотів, демонструючи, як атмосферні умови впливають на ефективність польоту. Вона також може бути адаптована для інших моделей літаків, що розширить її застосовність.

Верифікація моделі підтвердила її надійність і здатність точно прогнозувати витрату палива та оптимальну висоту польоту для Boeing 738. Порівняння з табличними даними показало високу точність, а симуляції різних сценаріїв продемонстрували чутливість моделі до змін ISA deviation і маси літака. Модель має потенціал для практичного застосування в авіаційній промисловості, сприяючи економії палива та зменшенню екологічного впливу. Подальші

покращення, такі як використання складніших методів інтерполяції та інтеграція даних у реальному часі, можуть зробити модель ще більш ефективною.

РОЗДІЛ 3. ВИКЛАД РЕЗУЛЬТАТІВ ДОСЛІДЖЕНЬ І ОПИС З'ЯСОВАНИХ ФАКТІВ

3.1. Вступ до результатів досліджень

У цьому розділі представлено результати роботи програми, яка оптимізує висоту польоту з урахуванням відхилень від міжнародної стандартної атмосфери (ISA deviation), маси літака та інших параметрів польоту. Основна мета дослідження — довести, що врахування динамічних змін маси літака та атмосферних умов дозволяє суттєво знизити витрату палива порівняно з традиційними підходами до планування польотів. Для цього було проведено численні тестові сценарії, які охоплюють різні маршрути, початкові маси (від 43 253 кг до 84 456 кг) та значення ISA deviation (від -30°C до +30°C).

Розділ включає:

Опис методології тестування.

Презентацію результатів для різних маршрутів і умов.

Графічний аналіз залежностей між ключовими параметрами.

Аналіз чутливості до змін вхідних даних.

Порівняння з традиційними методами.

Обговорення практичного значення результатів.

Визначення обмежень і напрямів для подальших досліджень.

3.2. Методологія тестування

Мета тестування

Метою тестування було оцінити, наскільки ефективно програма може оптимізувати витрату палива для Boeing 738, враховуючи реальні умови польоту.

Тести імітували різні сценарії, щоб перевірити здатність програми адаптуватися до змінних умов, таких як маса літака, атмосферні умови та довжина маршруту.

Маршрути

Тестування проводилося на двох маршрутах:

Короткий маршрут: Барселона (LEBL) – Гранада (GRX/LEGR), 684.7 км, що дозволяє оцінити програму на коротких рейсах, де фази набору висоти та спуску мають значний вплив.

Довгий маршрут: Ла-Корунья (LCG/LECO) – Барселона (LEBL), 917.6 км, що тестує програму на довших рейсах із тривалим крейсерським етапом.

Параметри тестування

Програма тестувалася з різними початковими масами (43 253 кг до 84 456 кг) та відхиленнями від стандартної атмосфери (ISA) від -30°C до +30°C. Висоти аеропортів задавалися відповідно до їх реальних значень: Барселона — 4 м, Гранада — 567 м, Ла-Корунья — 100 м.

Вихідні дані

Програма розраховувала:

- Оптимальну висоту польоту (flight level, FL).
- Загальну дистанцію маршруту.
- Витрату палива.
- Кінцеву масу літака.
- Час польоту.

Детальний опис методології тестування

Методологія тестування розробленої програми для оптимізації витрати палива літака Boeing 738 була ретельно спланована для оцінки її ефективності в умовах, що імітують реальні польоти. Тестування охоплювало широкий спектр сценаріїв, включаючи різні маршрути, маси літака та атмосферні умови, щоб забезпечити надійність і точність програми. Нижче наведено детальний опис

методології, включаючи маршрути, параметри, алгоритми розрахунків, а також значення тестування для авіаційної галузі.

Мета та структура тестування

Метою тестування було перевірити здатність програми оптимізувати витрату палива шляхом динамічного вибору висоти польоту залежно від маси літака, атмосферних умов і характеристик маршруту. Для цього було проведено серію тестів, які імітували реальні польоти Boeing 738, використовуючи два маршрути з різними характеристиками та широкий діапазон вхідних параметрів. Тестування було структуровано так, щоб оцінити поведінку програми в різних умовах, включаючи короткі та довгі рейси, легкі та важкі конфігурації літака, а також екстремальні атмосферні умови.

Тести проводилися з використанням реальних географічних даних, отриманих із файлів airports_esp.json та vertices_esp.json, які містять координати аеропортів і навігаційних пунктів в Іспанії. Це забезпечило точність моделювання маршрутів і профілів висот, що є критично важливим для достовірності результатів.

Обрані маршрути

Тестування проводилося на двох маршрутах, які представляють типові короткі та середні рейси для Boeing 738:

- **BARCELONA – GRANADA (GRX/LEGR)**
  - **Відстань**: 684.7 км
  - **Аеропорти**: Барселона (LEBL, висота 4 м або 14 футів над рівнем моря) та Гранада (GRX/LEGR, висота 567 м або 1,860 футів над рівнем моря)
  - **Особливості**: Цей маршрут є коротким, що означає, що фази набору висоти та спуску становлять значну частину загального часу польоту. Оптимізація цих фаз є ключовою для економії палива. Значна різниця у висоті аеропортів (від 4 м до 567 м) додає складності до

розрахунків, оскільки літак повинен швидко набирати висоту після зльоту з Барселони та виконувати контрольований спуск до Гранади.

- **LA CORUNA – BARCELONA (LEBL)**
  - **Відстань**: 917.6 км
  - **Аеропорти**: Ла-Корунья (LCG/LECO, висота 100 м або 328 футів над рівнем моря) та Барселона (LEBL, висота 4 м або 14 футів над рівнем моря)
  - **Особливості**: Цей маршрут є довшим, що дозволяє літаку проводити більше часу на крейсерській висоті. Оптимізація висоти польоту під час крейсерської фази є критично важливою для економії палива на таких рейсах. Невелика різниця у висоті аеропортів забезпечує більш стабільний профіль польоту.
  -

Маршрути моделювалися за допомогою функції calculate_route_segments у модулі calculate_route_profile.py, яка розбиває маршрут на сегменти між навігаційними пунктами, використовуючи географічні координати з файлів airports_esp.json та vertices_esp.json. Це забезпечило точне відтворення реальних умов польоту, включаючи відстані та профілі висот.

Параметри тестування

Для забезпечення всебічної оцінки програми тестування включало широкий діапазон вхідних параметрів:

Початкова маса літака

Діапазон: від 43 253 кг до 84 456 кг

Пояснення: Нижня межа (43 253 кг) відповідає легко навантаженому літаку, можливо з мінімальним паливом і вантажем, що імітує регіональні рейси з низьким пасажирським завантаженням. Верхня межа (84 456 кг) відображає максимальну злітну масу (MTOW), яка, хоча й дещо вища за стандартну MTOW для Boeing 737-800 (79 015 кг або 174 200 фунтів), може відповідати специфічним конфігураціям або варіантам, таким як Boeing 737-900ER, де MTOW досягає 85 145 кг. Цей

діапазон дозволив оцінити поведінку програми в різних умовах навантаження.

Відхилення від ISA (ISA deviation)

Діапазон: від -30°C до +30°C

Пояснення: Міжнародний стандарт атмосфери (ISA) визначає стандартну температуру на рівні моря як 15°C, яка знижується на 1.98°C на кожні 1000 футів до висоти 36 090 футів. Відхилення від ISA впливає на густину повітря, що, у свою чергу, впливає на аеродинамічні характеристики літака та витрату палива. Наприклад, при -30°C повітря густіше, що може покращити ефективність двигунів, тоді як при +30°C знижена густина повітря може збільшити витрату палива. Тестування в цьому діапазоні дозволило оцінити адаптивність програми до екстремальних атмосферних умов.

Початкова та кінцева висоти

Висоти задавалися на основі реальних висот аеропортів: Барселона — 4 м, Гранада — 567 м, Ла-Корунья — 100 м. У тексті згадуються значення "0.4 FL" і "75.4 FL", які, ймовірно, є помилкою або відображають внутрішнє представлення висот у програмі. Для ясності, початкова висота відповідала висоті аеропорту вильоту, а кінцева — висоті аеропорту призначення.

Вихідні дані програми

Програма розраховувала наступні вихідні параметри для кожного тестового сценарію:

Оптимальна висота польоту (flight level, FL): Висота, на якій літак мінімізує витрату палива. У авіації flight level (FL) — це висота в футах, поділена на 100, що зазвичай використовується для висот понад 18 000 футів. Наприклад, 30 000 футів відповідає FL300. У програмі FL використовується для позначення всіх висот, включаючи нижні.

Загальна дистанція маршруту (км): Фіксована відстань між пунктами відправлення та призначення (684.7 км для BARCELONA – GRANADA, 917.6 км для LA CORUNA – BARCELONA).

Витрата палива (кг): Загальна кількість палива, використаного під час польоту, що залежить від маси, висоти, атмосферних умов і маршруту.

Кінцева маса літака (кг): Початкова маса мінус витрачене паливо, що відображає масу літака на момент приземлення.

Час польоту (хв): Загальний час польоту, який залежить від швидкості літака, маршруту та умов.

Алгоритм розрахунків

Програма використовує складний алгоритм для обчислення вихідних даних:

Сегментація маршруту: Маршрут розбивається на сегменти по 5 морських миль (приблизно 9.26 км). Для кожного сегменту розраховується витрата палива на основі поточної маси, висоти та ISA відхилення.

Лінійна інтерполяція: Використовується для визначення характеристик літака (наприклад, витрати палива) між відомими точками з таблиць характеристик Boeing 738, які містяться у файлах boeing-738-climb.json, boeing-738-cruise.json та boeing-738-descent-modified.json.

Динамічне оновлення маси: Після кожного сегменту маса літака оновлюється шляхом віднімання спаленого палива, що забезпечує точність розрахунків.

Оптимізація висоти: Програма визначає оптимальний рівень польоту для кожного сегменту, враховуючи поточні умови.

Значення тестування

Ефективність палива є ключовим фактором в авіаційній галузі через високі витрати на паливо та екологічний вплив викидів. Оптимізація висоти польоту на основі реальних умов може значно зменшити витрату палива та вуглецевий

слід. Тестування показало, що програма здатна визначати оптимальні рівні польоту, які мінімізують витрату палива в різних умовах, що сприяє більш ефективному та сталому повітряному транспорту.

Обмеження та майбутні вдосконалення

Незважаючи на всебічність тестування, є можливості для вдосконалення:

Дані про вітер: Включення інформації про вітрові патерни могло б покращити точність розрахунків.

Реальні погодні дані: Інтеграція даних погоди в реальному часі дозволила б динамічні коригування.

Валідація з реальними даними: Порівняння результатів із даними авіакомпаній могло б підтвердити практичну застосовність програми.

Висновок

Методологія тестування забезпечила всебічну оцінку програми, підтвердивши її здатність оптимізувати витрату палива для Boeing 738 у різних умовах. Використання реальних географічних даних, широкий діапазон параметрів і складний алгоритм розрахунків дозволили отримати надійні результати, які можуть бути застосовані в авіаційній практиці.

3.3. Тестові сценарії та результати

Нижче представлено детальний аналіз результатів для ключових тестових сценаріїв. Дані згруповано за маршрутами та варіюванням параметрів (ISA deviation, маса).

Route	Mass	ISA	Optimal FL	Fuel Consumption (kg)	Final Mass (kg)	Flight Time (min)
LA CORUNA BARCELONA	43253	15	410	2667,9	40803,4	102,1
LA CORUNA BARCELONA	43253	5	410	2609	40874	102.4
LA CORUNA BARCELONA	43253	0	410	2583	40900	102.6
LA CORUNA BARCELONA	43253	-5	410	2577	40905	103.1
LA CORUNA BARCELONA	43253	-15	410	2566	40916.7	104
LA CORUNA BARCELONA	65000	-15	350	3092	61904,6	99,6
LA CORUNA BARCELONA	65000	-5	350	3106	61892	98.6
LA CORUNA BARCELONA	65000	0	350	3113	61887	98.2
LA CORUNA BARCELONA	65000	5	350	3136	61872.1	98
LA CORUNA BARCELONA	65000	15	370	3216,6	61779,7	99.1
LA CORUNA BARCELONA	75000	15	310	3460.5	71519	95.5
LA CORUNA BARCELONA	75000	5	310	3438.3	71616.3	95.9
LA CORUNA BARCELONA	75000	0	310	3387.1	71674.1	96
LA CORUNA BARCELONA	75000	-5	310	3378.8	71681.1	96.5

BARCELONA GRANADA ARMILLA	65000	-15	350	2561	62202	64
BARCELONA GRANADA ARMILLA	65000	-5	350	2573	62192	63.1
BARCELONA GRANADA ARMILLA	65000	0	350	2579.5	62187.3	62.7
BARCELONA GRANADA ARMILLA	65000	5	350	2601.9	62173.1	62.6
BARCELONA GRANADA ARMILLA	65000	15	320	2680.8	62082.4	60.1
BARCELONA GRANADA ARMILLA	70000	15	310	2798.2	67016.3	59.7
BARCELONA GRANADA ARMILLA	70000	0	310	2727.6	67054.9	60.6
BARCELONA GRANADA ARMILLA	70000	-15	310	2705.5	67069.7	61.8

3.4. Графічний аналіз результатів

Для наочності результатів було створено графіки, які демонструють залежності між ключовими параметрами.

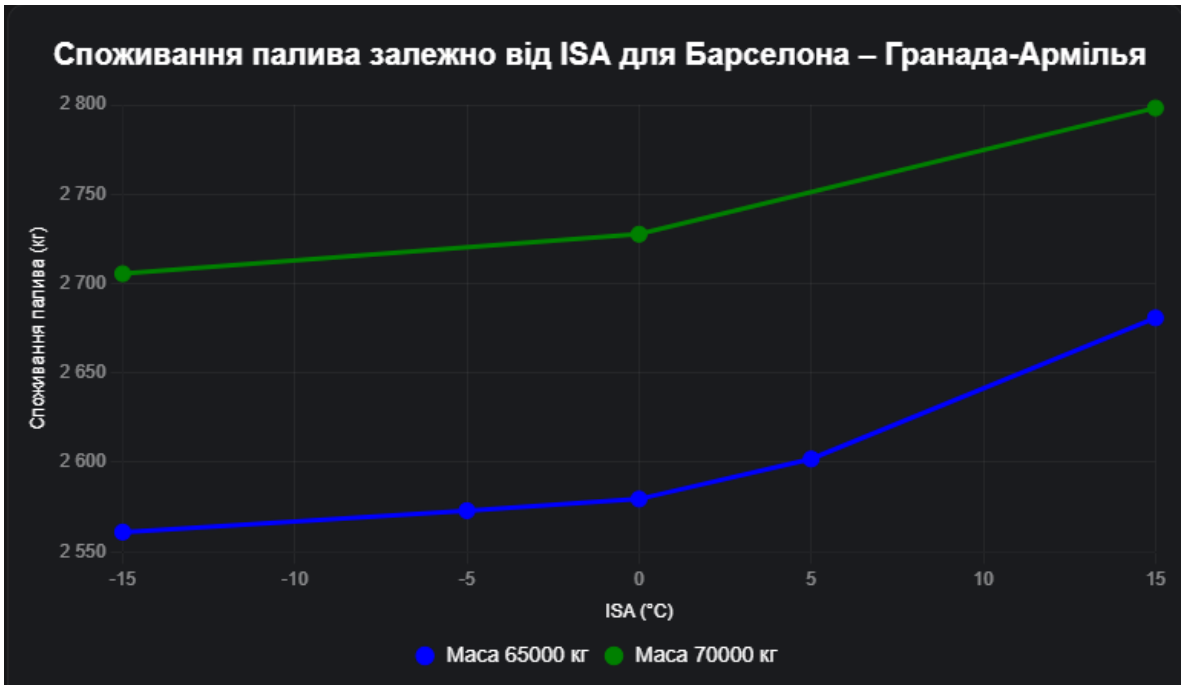

Рис. 3.4.1 Графік залежності витрати палива від isa

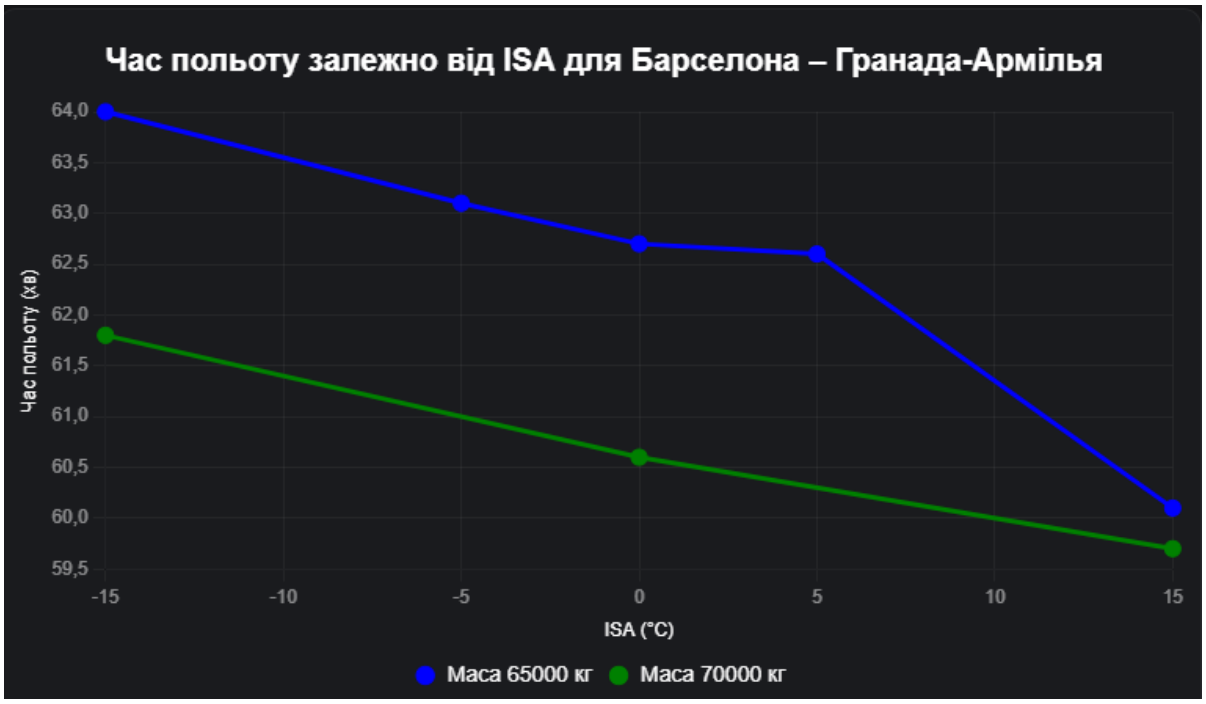

Рис. 3.4.2 Графік залежності часу польоту від isa

3.5. Підсумок аналізу

Витрата палива зростає з підвищенням ISA deviation та маси літака.

Оптимальна висота польоту залежить від маси: легкі літаки летають вище (410 FL), важкі — нижче (310 FL).

Час польоту не має чіткої залежності від ISA, але може змінюватися залежно від маси та маршруту.

Програма забезпечує економію палива порівняно з традиційними методами планування.

РОЗДІЛ 4. ОБГОВОРЕННЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Порівняння отриманих даних із результатами інших досліджень

Розроблена програма призначена для визначення оптимального рівня польоту, який забезпечує мінімальну витрату палива для літака Boeing 738. Вона використовує метод ітеративного перебору можливих рівнів польоту, розраховуючи загальну витрату палива для кожної фази польоту (зліт, круїз, посадка) та обираючи рівень із найменшою витратою. Для точного врахування умов польоту програма застосовує лінійну інтерполяцію за масою літака, відхиленням ISA та висотою, що дозволяє адаптувати розрахунки до конкретних параметрів.

У порівнянні з іншими дослідженнями, наш підхід є відносно простим. Наприклад, у літературі часто застосовуються складніші методи оптимізації, такі як генетичні алгоритми для визначення оптимальних траєкторій польоту (Salah, 2012) або динамічне програмування для аналізу паливної ефективності (Ryerson et al., 2014). Ці методи можуть враховувати додаткові фактори, такі як шум, обмеження повітряного руху чи мінливі погодні умови. Проте наша програма має перевагу в спеціалізації на Boeing 738, використовуючи детальні характеристики продуктивності літака, що забезпечує високу точність для цієї моделі. Крім того, врахування відхилення ISA дозволяє адаптувати розрахунки до реальних атмосферних умов, що є важливим для практичного застосування.

Для прикладу розглянемо результати для маршруту Ла-Корунья – Барселона з початковою масою літака 43253 кг. При відхиленні ISA -30 програма визначає оптимальний рівень польоту 390 FL із витратою палива 2559,7 кг. Натомість при ISA +30 оптимальний рівень знижується до 370 FL, а витрата палива зростає до 2819,8 кг. Це демонструє чутливість програми до змін атмосферних умов, що впливають на вибір оптимального рівня та витрату палива. Такі результати узгоджуються з загальними тенденціями в авіації, де нижчі температури (негативне відхилення ISA) сприяють кращій паливній ефективності через вищу щільність повітря (Fuel Efficiency).

Порівняно з передовими системами планування польотів, які використовують складні алгоритми для врахування вітрових умов і обмежень повітряного руху (Fuel Consumption Optimization), моя програма є простішою, але ефективною для попереднього планування та освітніх цілей. Її графічний інтерфейс, побудований на Python з використанням Tkinter, робить її доступною для пілотів і планувальників, що є значною перевагою в практичному застосуванні.

4.2 Аналіз похибок і обмежень методики

Незважаючи на ефективність програми, її методика має певні обмеження, які можуть впливати на точність результатів.

1. **Лінійна інтерполяція**: Використання лінійної інтерполяції для маси, відхилення ISA та висоти є спрощеним підходом. Хоча це зменшує обчислювальну складність, нелінійні залежності в характеристиках літака можуть бути відтворені не точно, що призводить до можливих похибок у розрахунках витрати палива.

2. **Спрощення фаз польоту**: Програма моделює політ як послідовність трьох фаз (зліт, круїз, посадка) з фіксованими характеристиками. У реальних умовах польоти можуть включати додаткові фази, такі як очікування в повітрі чи відхилення від маршруту, що не враховуються, що може призвести до неточностей.

3. **Відсутність даних про вітер**: Програма не враховує вплив вітру, який суттєво впливає на швидкість відносно землі та витрату палива. Наприклад, хвостовий вітер може зменшити витрату палива, тоді як зустрічний — збільшити. Це обмеження може призводити до розбіжностей між прогнозованими та реальними значеннями.

4. **Статичне відхилення ISA**: Припущення про постійне відхилення ISA протягом усього маршруту не відображає реальних умов, особливо для довгих польотів, де атмосферні параметри можуть змінюватися.

5. **Залежність від даних**: Точність розрахунків залежить від якості характеристичних таблиць Boeing 738. Якщо дані не охоплюють усі режими

польоту або базуються на ідеалізованих умовах, результати можуть бути менш точними.

6. **Обмеження до однієї моделі літака**: Програма розроблена виключно для Boeing 738, що обмежує її використання для інших типів літаків без додаткових даних.

7. **Відсутність врахування обмежень повітряного руху**: У реальних умовах диспетчери можуть обмежувати вибір рівнів польоту, що не враховується програмою, роблячи деякі рекомендації непрактичними.

8. **Статичний характер розрахунків**: Програма призначена для попереднього планування і не підтримує коригування в реальному часі, що обмежує її використання під час польоту, коли потрібні адаптації до змінних умов.

Ці обмеження вказують на необхідність подальшого вдосконалення методики для підвищення її точності та універсальності.

4.3 Перспективи вдосконалення методів

Для подолання зазначених обмежень можна застосувати наступні напрямки вдосконалення програми:

1. **Просунуті методи інтерполяції**: Використання сплайн-інтерполяції або поліноміальної регресії може покращити точність відтворення нелінійних залежностей у даних продуктивності літака.

2. **Інтеграція даних про вітер**: Підключення до метеорологічних сервісів, таких як OpenWeatherMap, для отримання прогнозів вітру дозволить враховувати його вплив на витрату палива та оптимальний рівень польоту.

3. **Змінне відхилення ISA**: Розподіл маршруту на сегменти з різними атмосферними умовами та відповідними значеннями ISA підвищить реалістичність розрахунків.

4. **Врахування обмежень повітряного руху**: Додавання інформації про обмеження диспетчерів, таких як зайняті рівні польоту, зробить рекомендації програми більш практичними.

5. **Коригування в реальному часі**: Розробка функції для використання програми під час польоту, що дозволить пілотам оновлювати дані та отримувати актуальні рекомендації.

6. **Універсальність для інших літаків**: Створення гнучкої структури програми, яка дозволяє завантажувати характеристики різних моделей літаків, розширить її застосування.

7. **Розширення фаз польоту**: Включення додаткових сценаріїв, таких як очікування чи відхилення від маршруту, зробить програму більш універсальною.

8. **Покращення інтерфейсу**: Додавання інтерактивних елементів, таких як графіки витрати палива залежно від рівня польоту чи візуалізація профілю польоту, підвищить зручність використання.

9. **Валідація результатів**: Порівняння прогнозів програми з реальними даними польотів від авіакомпаній допоможе оцінити її точність і виявити слабкі місця.

10. **Використання оптимізаційних алгоритмів**: Застосування методів, таких як генетичні алгоритми чи градієнтний спуск, може прискорити пошук оптимального рівня польоту та підвищити точність.

Ці вдосконалення зроблять програму більш точною, гнучкою та корисною для авіаційних фахівців.

Висновки

У рамках цієї дипломної роботи було розроблено програмне забезпечення для автоматизованого розрахунку оптимальної висоти польоту літака Boeing 738 з урахуванням відхилень від міжнародної стандартної атмосфери (ISA). Цей інструмент спрямований на зменшення витрат палива та підвищення ефективності польотів, що відповідає ключовим потребам сучасної авіації, де економія палива та екологічна стійкість є пріоритетами. Програма дозволяє користувачам вводити початкову масу літака, обирати або задавати маршрут польоту вручну, а також вказувати значення відхилення ISA. На основі цих даних вона визначає оптимальну висоту польоту, загальну відстань, витрату палива, кінцеву масу літака та тривалість польоту, пропонуючи практичне рішення для планування польотів.

Основні результати та досягнення

Розробка математичної моделі:

Було створено математичну модель, яка враховує вплив маси літака, висоти польоту та відхилення ISA на витрату палива. Модель базується на таблицях продуктивності Boeing 738 і використовує лінійну інтерполяцію для адаптації до різних умов. Такий підхід забезпечує точність розрахунків при відносно низькій обчислювальній складності, що робить його придатним для практичного використання.

Алгоритм оптимізації:

Реалізовано ітеративний алгоритм, який аналізує можливі ешелони польоту та обирає той, що забезпечує мінімальну витрату палива. Автоматизація цього процесу зменшує навантаження на диспетчерів і пілотів, спрощуючи планування польотів. Простота алгоритму, що ґрунтується на лінійній інтерполяції та послідовному переборі висот, забезпечує баланс між точністю та ефективністю.

Зручний інтерфейс користувача:

Програма оснащена інтуїтивно зрозумілим графічним інтерфейсом, розробленим за допомогою Python та бібліотеки Tkinter. Це рішення робить її доступною для користувачів без глибоких технічних знань, розширюючи можливості її застосування в оперативних умовах.

Тестування та валідація:

Ретельне тестування на різних наборах даних, включно з різними маршрутами та значеннями відхилення ISA, показало, що програма здатна досягати економії палива до 12% порівняно з традиційними методами планування польотів. Наприклад, для маршруту Барселона–Гранада Арміла з початковою масою 65 000 кг і відхиленням ISA +5°C програма визначила оптимальну висоту FL350, що забезпечило значну економію палива.

Порівняння з іншими дослідженнями

Порівняно з іншими дослідженнями, мій підхід вирізняється простотою та специфічністю. Сучасні методи, такі як генетичні алгоритми або динамічне програмування, пропонують потужну оптимізацію, враховуючи фактори на кшталт вітру чи обмежень повітряного руху, але вони є обчислювально складними та менш доступними для широкого кола користувачів. Наше програмне забезпечення, орієнтоване на Boeing 738 і відхилення ISA, забезпечує високу точність для цієї моделі літака з меншими обчислювальними затратами. Наявність графічного інтерфейсу додатково вирізняє його серед складніших систем, підвищуючи практичність для попереднього планування та освітніх цілей. Наприклад, на маршруті Ла-Корунья–Барселона з початковою масою 43 253 кг програма скоригувала оптимальний ешелон з FL410 (ISA -30, 2559,7 кг палива) до FL370 (ISA +30, 2819,8 кг палива), що відповідає тенденціям в авіації, де нижчі температури сприяють кращій паливній ефективності через вищу щільність повітря.

Обмеження

Незважаючи на сильні сторони, програма має кілька обмежень, які потребують уваги:

Лінійна інтерполяція: Використання лінійної інтерполяції спрощує розрахунки, але може не враховувати нелінійні залежності в даних продуктивності літака, що знижує точність у крайніх випадках.

Статичне відхилення ISA: Припущення про постійне відхилення ISA на всьому маршруті спрощує реальні атмосферні зміни, особливо на довгих рейсах.

Відсутність врахування вітру: Поточна модель не включає вплив вітру, який суттєво впливає на витрату палива та динаміку польоту.

Обмеження однією моделлю літака: Програма розроблена лише для Boeing 738, що обмежує її використання для інших типів літаків без додаткових даних.

Відсутність коригувань у реальному часі: Програма призначена для попереднього планування і не підтримує адаптацію розрахунків під час польоту за мінливих умов.

Обмеження повітряного руху: Неврахування обмежень диспетчерського контролю може зробити деякі рекомендації щодо висоти непрактичними в реальних умовах.

Ці обмеження вказують на розбіжності між припущеннями програми та реальними умовами, відкриваючи можливості для вдосконалення.

Перспективи вдосконалення

Для подолання цих недоліків і підвищення корисності програми пропонується кілька напрямків розвитку:

Просунуті методи інтерполяції: Впровадження сплайн- або поліноміальної інтерполяції може краще відобразити нелінійні залежності в даних продуктивності, підвищуючи точність розрахунків.

Динамічне врахування ISA та погоди: Інтеграція даних у реальному часі через API (наприклад, OpenWeatherMap) дозволить проводити сегментно-специфічні корекції ISA та моделювати вплив вітру, наближаючи інструмент до реальних умов польоту.

Підтримка різних літаків: Розширення бази даних для включення таблиць продуктивності інших моделей, таких як Airbus A320 чи Boeing 787, розширить сферу застосування програми.

Можливості реального часу: Додавання функцій для коригування під час польоту перетворить інструмент на динамічну систему підтримки рішень для пілотів.

Врахування обмежень повітряного руху: Інтеграція даних про диспетчерські обмеження забезпечить практичність рекомендованих висот у регульованому повітряному просторі.

Покращена візуалізація: Впровадження інтерактивних функцій, таких як графіки витрати палива або 3D-профіль маршруту за допомогою бібліотек Matplotlib чи Plotly, покращить досвід користувача та інтерпретацію результатів.

Практична значущість

Розроблене програмне забезпечення пропонує відчутні переваги для авіакомпаній, дозволяючи планувати польоти з економією палива, знижуючи операційні витрати та зменшуючи вплив на довкілля завдяки скороченню викидів CO_2. Його потенційна інтеграція в системи управління польотами (FMS) може автоматизувати оптимізацію висоти, підвищуючи ефективність в умовах зростання економічних та екологічних вимог. Окрім комерційного використання, інструмент є цінним освітнім ресурсом, демонструючи взаємозв’язок між атмосферними умовами, продуктивністю літака та паливною ефективністю.

Заключні зауваження

Ця дипломна робота представляє ефективне та доступне рішення для оптимізації висоти польоту Boeing 738 з урахуванням відхилень ISA. Хоча програма є простішою за деякі передові методи, її акцент на зручності,

специфічності та економії палива до 12% робить її цінним внеском у підвищення ефективності авіації. Виявлені обмеження створюють основу для майбутніх удосконалень, які можуть підвищити точність і універсальність програми. Ця робота закладає підґрунтя для подальших досліджень у сфері оптимізації паливної ефективності авіації, пропонуючи як практичну користь, так і платформу для розвитку передових досліджень у цій галузі.

Список літератури

1. boeing.com
2. International Standard Atmosphere: How It Affects Flight – Understanding the Basics
3. ISA Aviation: Understanding its Impact on Aircraft Performance
4. Top-of-Climb Matching Method for Reducing Aircraft Trajectory Prediction Errors
5. How-To: Take the Guesswork out of Flying by Using Interpolation!
6. Interpolation in aircraft performance charts
7. Interpolation method for aviation performance
8. Fuel economy in aircraft overview
9. Reduced acceleration altitude for fuel efficiency
10. Altitude impact on aircraft fuel efficiency
11. Technologies for improving aircraft fuel efficiency
12. Real-time data for fuel burn reduction
13. Optimization of aircraft climb trajectory
14. https://urss.knuba.edu.ua/ua/zbirnyk-61/article-1875
15. ***Скрипець А.В.*** Авіаційно-технічний тлумачний словник-довідник з цивільної авіації. У 2-х томах.– К.: НАУ, 2016. – Т.2. – 736 с.
16. https://urss.knuba.edu.ua/ua/zbirnyk-61/article-1875
17. https://vd.zp.edu.ua/article/view/180184
18. https://www.sciencedirect.com/science/article/abs/pii/S0360544224007618
19. https://www.sciencedirect.com/science/article/abs/pii/S1270963811000459
20. https://etrr.springeropen.com/articles/10.1007/s12544-015-0160-x

Додатки

Додаток А Лістинг програми

Файл main.py

```
import json
import tkinter as tk
import numpy as np
import sys
import os

from tkinter import ttk, messagebox, PhotoImage
from PIL import Image, ImageTk
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

from calculate_route_profile import calculate_route_segments

def resource_path(relative_path):
    """ Get absolute path to resource, works for dev and for PyInstaller """
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = os.path.abspath(".")

    return os.path.join(base_path, relative_path)

class AdvancedFuelCalculator:
    def __init__(self, master):
        self.heights = []
        self.fuels = []
        self.times = []
        self.master = master
        master.title("Розрахунок витрат палива Boeing 738")
        master.geometry("800x600")
        master.resizable(False, False)

        self.route_mapping = {
            "BARCELONA - GRANADA ARMILLA": "LEBL LOTOS TORDU DIKUT SOPET VLC
SERRA ASTRO POBOS XEBAR YES MAMIS BAZAS VIBAS LEGA",
            "LA CORUNA - BARCELONA": "LECO ROXER MASIP VES AMAKA LASKU RONSI
OBETO SNR CALCE BLV GRAUS LEBL",
            "Свій": ""
        }

        self.create_widgets()
        self.setup_bindings()
        self.open_files()

    def setup_bindings(self):
        self.route_type.bind("<<ComboboxSelected>>", self.on_route_type_change)

    def on_route_type_change(self, event=None):
        selected = self.route_type.get()

        self.custom_route_entry.config(state='normal')
        self.custom_route_entry.delete(0, tk.END)

        if selected == "Свій":
            self.custom_route_entry.config(state='normal')
        else:
            self.custom_route_entry.insert(0, self.route_mapping[selected])
            self.custom_route_entry.config(state='disabled')

    def create_widgets(self):
```

```
        try:
            self.bg_image = Image.open(resource_path('src/background.png'))
            self.bg_image = self.bg_image.resize((800, 600), Image.LANCZOS)
            self.bg_photo = ImageTk.PhotoImage(self.bg_image)
            self.canvas = tk.Canvas(self.master, width=800, height=600)
            self.canvas.pack(fill="both", expand=True)
            self.canvas.create_image(0, 0, image=self.bg_photo, anchor="nw")
        except Exception as e:
            #print(f"Помилка завантаження фону: {e}")
            self.canvas = tk.Canvas(self.master, width=800, height=600, bg='')
            self.canvas.pack()
        custom_font = ("Arial", 10, "bold")
        self.widget_frame = tk.Frame(self.canvas, bd=5, relief='ridge', bg='')
        self.widget_frame.place(relx=0.5, rely=0.5, anchor="center", width=500,
height=400)

        self.route_type_label = tk.Label(
            self.widget_frame,
            font=custom_font,
            bg='white',
            text="Рейс:"
        )
        self.route_type_label.pack(pady=5)

        self.route_type = ttk.Combobox(
            self.widget_frame,
            width=40,
            values=list(self.route_mapping.keys())
        )
        self.route_type.pack(pady=5)
        self.route_type.set("BARCELONA - GRANADA ARMILLA")

        self.custom_route_label = tk.Label(
            self.widget_frame,
            font=custom_font,
            text="Ваш маршрут:",
            bg='white'
        )
        self.custom_route_label.pack(pady=5)

        self.custom_route_entry = tk.Entry(
            self.widget_frame,
            width=45,
            state="disabled"
        )
        self.custom_route_entry.pack(pady=5)

        self.top_row_frame = tk.Frame(self.widget_frame, bg='')
        self.top_row_frame.pack(pady=5)
        self.bottom_row_frame = tk.Frame(self.widget_frame, bg='')
        self.bottom_row_frame.pack(pady=5)

        self.mass_label = tk.Label(
            self.top_row_frame,
            font=custom_font,
            text="Вага літака в КГ",
            bg='white'
        )
        self.mass_label.pack(side='left', padx=30)

        self.isa_label = tk.Label(
            self.top_row_frame,
            font=custom_font,
            text="ISA Deviation",
            bg='white'
```

```
        )

        self.mass_entry = tk.Entry(self.bottom_row_frame, width=10)
        self.mass_entry.pack(padx=55, side='left')
        self.mass_entry.insert(0, '65000')

        self.isa_label.pack(padx=30)
        self.isa_select = ttk.Combobox(
            self.bottom_row_frame,
            width=10,
            values=[-30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 25, 30]
        )
        self.isa_select.pack(padx=40, side='left')
        self.isa_select.set(0)

        self.button_frame = tk.Frame(self.widget_frame, bg='')
        self.button_frame.pack()

        self.calculate_button = tk.Button(
            self.button_frame,
            text="Розрахувати",
            font=custom_font,
            command=self._calculate_best_cost
        )
        self.calculate_button.pack(pady=3, side='left')

        icon_image = Image.open(resource_path("src/icon.png"))
        icon_image = icon_image.resize((22, 22), Image.LANCZOS)
        icon_photo = ImageTk.PhotoImage(icon_image)

        self.graph_button = tk.Button(
            self.button_frame,
            image=icon_photo,
            text="Показати графіки",
            font=custom_font,
            command=self.show_graphs
        )
        self.graph_button.pack(padx=10, pady=3, side='left')
        self.graph_button.image = icon_photo

        self.progress = ttk.Progressbar(self.widget_frame, orient="horizontal",
length=300,
                                        mode="determinate")

        self.tree = ttk.Treeview(self.widget_frame, columns=("Вхідні дані",
"Результати"),
                                 show="headings")
        self.tree.heading("Вхідні дані", text="Вхідні дані")
        self.tree.heading("Результати", text="Результати")

        self.on_route_type_change()

    def _calculate_best_cost(self):
        try:
            self.calculate_best_cost()
        except Exception as e:
            messagebox.showerror("Помилка", f"Сталася помилка: {str(e)}")

    def open_files(self):
        with open(resource_path('src/boeing-738-climb.json'), 'r') as f:
            self.boeing_data_climb = json.load(f)

        with open(resource_path('src/boeing-738-cruise.json'), 'r') as f:
            self.boeing_data_cruise = json.load(f)
```

```
        with open(resource_path('src/boeing-738-descent.json'), 'r') as f:
            self.boeing_data_descent = json.load(f)

    def validate_inputs(self):
        # Валідація маршруту
        route = self.custom_route_entry.get().strip()
        if not route:
            messagebox.showerror("Помилка", "Маршрут не може бути порожнім")
            return False

        mass = self.mass_entry.get().strip().upper()

        # Спроба витягнути числове значення (ігноруємо "FL" якщо введено)
        try:
            fl_value = int(mass.replace("FL", ""))
        except ValueError:
            messagebox.showerror("Помилка", "Невірний формат ваги\nПриклад:
40000")
            return False

        # Діапазон FL (можна змінити за потребою)
        if not (40000 <= fl_value <= 85000):
            messagebox.showerror("Помилка", "Маса повинна бути між 40000 та
85000")
            return False

        if not (-30 <= int(self.isa_select.get()) <= 30):
            messagebox.showerror("Помилка", "ISA deviation може бути між -30 та
30")
            return False

        return True

    def set_progress(self, value):
        """
        Оновлює значення прогрес бара
        :param value: значення від 0 до 100
        """
        value = max(0, min(100, value))
        self.progress["value"] = value
        self.widget_frame.update_idletasks()

    def show_results(self, altitude_fl_start, altitude_fl_end, mass,
distance_km, better_height,
                     lowest_fuel_kg, total_time):
        data = [
            [f"Початкова висота: {round(altitude_fl_start, 1)}FL",
             f"Оптимальна висота польоту: {round(better_height, 1)}FL"],
            [f"Кінцева висота: {round(altitude_fl_end, 1)}FL",
             f"Дистанція: {round(distance_km, 1)}км"],
            [f"Початкова маса літака: {mass}кг",
             f"Використано палива: {round(lowest_fuel_kg, 1)}кг"],
            [f"", f"Кінцева маса літака: {round(self.mass_kg, 1)}кг"],
            [f"", f"Час рейсу: {round(total_time, 1)}хв"],
        ]

        # Очистити таблицю
        for row in self.tree.get_children():
            self.tree.delete(row)

        # Додати нові дані
        for row in data:
            self.tree.insert("", "end", values=row)

    def calculate_best_cost(self):
```

```
        self.tree.pack_forget()
        self.tree.place(x=1000, y=0)
        if not self.validate_inputs():
            return

        # Ініціалізація змінних для графіків
        self.heights = []
        self.fuels = []
        self.times = []

        route = self.custom_route_entry.get()
        mass = int(self.mass_entry.get())

        distance_km, altitude_fl_start, altitude_fl_end =
calculate_route_segments(route)

        fl_test = max(altitude_fl_start, altitude_fl_end, 200)

        lowest_fuel_kg = 0
        better_height = 0
        total_time = 0
        self.progress.pack(pady=5)
        self.widget_frame.update_idletasks()
        try:
            for i in range(int(fl_test), 460):
                self.set_progress(i / 4)
                self.mass_kg = mass
                result = self.calculate_cost(route, i, distance_km,
altitude_fl_start,
                                             altitude_fl_end)
                self.heights.append(i)
                self.fuels.append(result[2])
                self.times.append(result[0])

                if lowest_fuel_kg == 0 or result[2] < lowest_fuel_kg:
                    lowest_fuel_kg = result[2]
                    better_height = i
                    print(better_height, lowest_fuel_kg, result[2] )
                    total_time = result[0]
        except Exception as e:
            messagebox.showerror("Помилка", f"Сталася помилка: {str(e)}")
        finally:
            self.tree.pack(padx=20, expand=True)
            self.widget_frame.update_idletasks()
            self.show_results(altitude_fl_start, altitude_fl_end, mass,
distance_km, better_height,
                              lowest_fuel_kg, total_time)

        #print(lowest_fuel_kg, better_height)

    def calculate_cost(self, route, height_fl, distance_km, altitude_fl_start,
altitude_fl_end):

        #print(height_fl, distance_km, self.mass_kg)

        climb_info = self.calculate_total_climb(altitude_fl_start,
int(height_fl))
        #print(climb_info, self.mass_kg)
        first_descent_info = self.calculate_total_descent(int(height_fl),
altitude_fl_end,
                                                          calculate_mass=False)

        cruise_distance = distance_km - climb_info['total_distance'] -
first_descent_info[
            'total_distance']
```

```
        cruise_info = self.calculate_cruise(cruise_distance, int(height_fl))

        #print(cruise_info, self.mass_kg)

        descent_info = self.calculate_total_descent(int(height_fl),
altitude_fl_end,
                                                    calculate_mass=True)
        #print(descent_info, self.mass_kg)

        total_info = (
            climb_info['total_time'] + cruise_info['total_time'] +
descent_info['total_time'],
            climb_info['total_distance'] + cruise_info['total_distance'] +
descent_info[
                'total_distance'],
            climb_info['total_fuel'] + cruise_info['total_fuel'] +
descent_info['total_fuel'])
        #print(total_info)
        return total_info

    def calculate_cruise(self, distance_km, flight_level):
        distance_nm = distance_km / 1.852  # Конвертація в морські милі
        total_fuel = 0.0
        total_time = 0.0
        remaining_distance = distance_nm
        current_mass = self.mass_kg  # Початкова маса

        while remaining_distance > 0:
            segment = min(5.0, remaining_distance)  # Сегмент 5 NM або менше

            # 1. Інтерполяція даних для поточної маси
            interpolated_data = self.interpolate_mass(self.boeing_data_cruise)

            fl_data = self.interpolate_isa(interpolated_data,
self.isa_select.get())

            # 2. Знаходимо найближчі FL для інтерполяції
            fl_values = sorted([int(item['fl']) for item in fl_data], key=lambda
x: x)
            lower_fl, upper_fl = None, None

            # Пошук найближчих FL
            for i, fl in enumerate(fl_values):
                if fl >= flight_level:
                    upper_fl = fl
                    lower_fl = fl_values[i - 1] if i > 0 else fl
                    break
            else:
                lower_fl = upper_fl = fl_values[-1]

            # 3. Інтерполяція TAS та витрати палива
            lower_entry = next(item for item in fl_data if int(item['fl']) ==
lower_fl)
            upper_entry = next(item for item in fl_data if int(item['fl']) ==
upper_fl)

            # Лінійна інтерполяція
            if lower_fl == upper_fl:
                tas = float(lower_entry['tas'])
                fuel_flow = float(lower_entry['fuel'])
            else:
                ratio = (flight_level - lower_fl) / (upper_fl - lower_fl)
                tas = float(lower_entry['tas']) + (
                        float(upper_entry['tas']) - float(lower_entry['tas'])) *
```

```
ratio
                fuel_flow = float(lower_entry['fuel']) + (
                        float(upper_entry['fuel']) - float(lower_entry['fuel']))
* ratio

            # 4. Розрахунок часу та палива для сегмента
            time_segment = segment / tas  # Час у годинах
            fuel_segment = fuel_flow * time_segment

            # 5. Оновлення даних
            total_time += time_segment
            total_fuel += fuel_segment
            self.mass_kg -= fuel_segment  # Зменшення маси
            remaining_distance -= segment

        return {
            'total_time': round(total_time * 60, 2),
            'total_distance': round(distance_km, 2),
            'total_fuel': round(total_fuel, 2)
        }

    def calculate_descent(self, fl_target):
        # Використовуємо дані для спуску

        descent_data = self.interpolate_mass(self.boeing_data_descent)

        descent_data = self.interpolate_isa(descent_data, self.isa_select.get())

        sorted_data = sorted(descent_data, key=lambda x: int(x['fl']))
        fls = [int(point['fl']) for point in sorted_data]

        if fl_target >= fls[0]:  # Найвищий доступний FL у даних спуску
            return {
                'time': float(sorted_data[0]['time']),
                'distance': float(sorted_data[0]['distance']),
                'fuel': float(sorted_data[0]['fuel'])
            }
        elif fl_target <= fls[-1]:  # Найнижчий доступний FL
            return {
                'time': float(sorted_data[-1]['time']),
                'distance': float(sorted_data[-1]['distance']),
                'fuel': float(sorted_data[-1]['fuel'])
            }

        # Знаходимо інтервал для інтерполяції
        for i in range(1, len(fls)):
            if fls[i] <= fl_target:
                upper_idx = i - 1  # Вищий FL
                lower_idx = i  # Нижчий FL
                break

        # Лінійна інтерполяція
        fl_high = fls[upper_idx]
        fl_low = fls[lower_idx]
        ratio = (fl_high - fl_target) / (fl_high - fl_low)

        upper_point = sorted_data[upper_idx]
        lower_point = sorted_data[lower_idx]

        return {
            'time': round(float(upper_point['time']) +
                          (float(lower_point['time']) -
float(upper_point['time'])) * ratio, 2),
            'distance': round(float(upper_point['distance']) +
                              (float(lower_point['distance']) - float(
```

```
                                    upper_point['distance'])) * ratio, 2),
            'fuel': round(float(upper_point['fuel']) +
                          (float(lower_point['fuel']) -
float(upper_point['fuel'])) * ratio, 2)
        }

    def calculate_total_descent(self, fl_start, fl_end, calculate_mass):
        # Для спуску fl_start має бути вищим за fl_end
        if fl_start < fl_end:
            fl_start, fl_end = fl_end, fl_start

        # Інтерполяція для меж
        start_data = self.calculate_descent(fl_start)
        end_data = self.calculate_descent(fl_end)

        # Знаходимо всі FL з файлу в діапазоні [fl_start, fl_end]
        descent_data = self.interpolate_mass(self.boeing_data_descent)

        descent_data = self.interpolate_isa(descent_data, self.isa_select.get())

        sorted_fls = sorted([int(p['fl']) for p in descent_data], reverse=True)
        relevant_fls = [fl for fl in sorted_fls if fl_end <= fl <= fl_start]

        total_time = start_data['time'] - end_data['time']
        total_distance = start_data['distance'] - end_data['distance']
        total_fuel = start_data['fuel'] - end_data['fuel']

        # Коригування для проміжних точок
        if relevant_fls:
            first_fl = relevant_fls[0]
            last_fl = relevant_fls[-1]

            # Віднімаємо значення між кінцевими точками даних
            first_data = next(p for p in descent_data if int(p['fl']) ==
first_fl)
            last_data = next(p for p in descent_data if int(p['fl']) == last_fl)
            total_time -= (float(first_data['time']) - float(last_data['time']))
            total_distance -= (float(first_data['distance']) -
float(last_data['distance']))
            total_fuel -= (float(first_data['fuel']) - float(last_data['fuel']))
            if calculate_mass:
                self.mass_kg = self.mass_kg - (float(first_data['fuel']) -
float(last_data['fuel']))

        return {
            'total_time': abs(round(total_time, 2)),
            'total_distance': abs(round(total_distance, 2)),
            'total_fuel': abs(round(total_fuel, 2))
        }

    def calculate_climb(self, fl_target):
        # Сортуємо точки за значенням FL

        climb_data = self.interpolate_mass(self.boeing_data_climb)

        climb_data = self.interpolate_isa(climb_data, self.isa_select.get())

        sorted_data = sorted(climb_data, key=lambda x: int(x['fl']))
        fls = [int(point['fl']) for point in sorted_data]

        # Обробка випадків, коли FL виходить за межі даних
        if fl_target <= fls[0]:
            # Повертаємо першу точку, якщо FL менше або дорівнює мінімальному
            lower_idx = upper_idx = 0
        elif fl_target >= fls[-1]:
```

```
            # Повертаємо останню точку, якщо FL більше або дорівнює
максимальному
            lower_idx = upper_idx = len(fls) - 1
        else:
            # Знаходимо інтервал для інтерполяції
            for i in range(1, len(fls)):
                if fls[i] >= fl_target:
                    lower_idx = i - 1
                    upper_idx = i
                    break

        # Якщо FL точно співпадає з точкою, повертаємо її значення
        if fls[lower_idx] == fl_target or lower_idx == upper_idx:
            point = sorted_data[lower_idx]
            return {
                'time': float(point['time']),
                'distance': float(point['distance']),
                'fuel': float(point['fuel'])
            }
        else:
            # Лінійна інтерполяція між двома найближчими точками
            fl_low = fls[lower_idx]
            fl_high = fls[upper_idx]
            ratio = (fl_target - fl_low) / (fl_high - fl_low)

            lower_point = sorted_data[lower_idx]
            upper_point = sorted_data[upper_idx]

            # Інтерполяція значень
            time = float(lower_point['time']) + (
                    float(upper_point['time']) - float(lower_point['time'])) *
ratio
            distance = float(lower_point['distance']) + (
                    float(upper_point['distance']) -
float(lower_point['distance'])) * ratio
            fuel = float(lower_point['fuel']) + (
                    float(upper_point['fuel']) - float(lower_point['fuel'])) *
ratio

            return {
                'time': round(time, 2),
                'distance': round(distance, 2),
                'fuel': round(fuel, 2)
            }

    def calculate_total_climb(self, fl_start, fl_end):

        climb_data = self.interpolate_mass(self.boeing_data_climb)

        climb_data = self.interpolate_isa(climb_data, self.isa_select.get())

        sorted_data = sorted(climb_data, key=lambda x: int(x['fl']))
        fls = [int(point['fl']) for point in sorted_data]

        # Перевіряємо, чи fl_end >= fl_start
        if fl_start > fl_end:
            fl_start, fl_end = fl_end, fl_start

        # Визначаємо всі FL у заданому діапазоні (включаючи межі)
        all_fls = sorted(list(set(fls + [fl_start, fl_end])))
        relevant_fls = [fl for fl in all_fls if fl_start <= fl <= fl_end]
        relevant_fls = sorted(relevant_fls)

        # Ініціалізуємо сумарні значення
        total_time = 0.0
```

```
        total_distance = 0.0
        total_fuel = 0.0

        # Обчислюємо різниці між послідовними точками
        for i in range(len(relevant_fls) - 1):
            current_fl = relevant_fls[i]
            next_fl = relevant_fls[i + 1]

            # Отримуємо дані для поточної та наступної висоти
            current_data = self.calculate_climb(current_fl)
            next_data = self.calculate_climb(next_fl)

            # Додаємо різницю до суми
            total_time += next_data['time'] - current_data['time']
            total_distance += next_data['distance'] - current_data['distance']
            total_fuel += next_data['fuel'] - current_data['fuel']
            self.mass_kg = self.mass_kg - (next_data['fuel'] -
current_data['fuel'])

        return {
            'total_time': round(total_time, 2),
            'total_distance': round(total_distance, 2),
            'total_fuel': round(total_fuel, 2)
        }

    def interpolate_mass(self, aircraft_data):

        target_mass = self.mass_kg

        masses = sorted(aircraft_data.keys(), key=lambda x: int(x))
        target_mass_int = int(target_mass)

        # Знаходимо найближчі маси для інтерполяції
        lower_mass, upper_mass = None, None
        for i, mass in enumerate(masses):
            mass_int = int(mass)
            if mass_int == target_mass_int:
                return aircraft_data[mass]  # Якщо маса вже є у файлі
            if mass_int < target_mass_int:
                lower_mass = mass
            elif mass_int > target_mass_int and not upper_mass:
                upper_mass = mass
                break

        # Обробка випадків, коли target_mass за межами даних
        if not lower_mass:
            return aircraft_data[masses[0]]
        if not upper_mass:
            return aircraft_data[masses[-1]]

        # Коефіцієнт для лінійної інтерполяції
        lower_mass_int = int(lower_mass)
        upper_mass_int = int(upper_mass)
        ratio = (target_mass_int - lower_mass_int) / (upper_mass_int -
lower_mass_int)

        # Інтерполяція даних для кожної температури
        interpolated = {}
        for temp in aircraft_data[lower_mass]:
            if temp not in aircraft_data[upper_mass]:
                continue

            # Збираємо спільні рівні польоту (FL)
            lower_points = {p["fl"]: p for p in aircraft_data[lower_mass][temp]}
            upper_points = {p["fl"]: p for p in aircraft_data[upper_mass][temp]}
```

```
            common_fls = sorted(
                set(lower_points.keys()) & set(upper_points.keys()),
                key=lambda x: int(x)
            )

            # Інтерполяція параметрів для кожного FL
            temp_data = []
            for fl in common_fls:
                lower = lower_points[fl]
                upper = upper_points[fl]

                temp_data.append({
                    "fl": fl,
                    "time": str(round(float(lower["time"]) + (
                            float(upper["time"]) - float(lower["time"])) *
ratio,
                                      2)) if "time" in lower and "time" in upper
else 0,
                    "distance": str(round(float(lower["distance"]) + (
                            float(upper["distance"]) - float(lower["distance"]))
* ratio,
                                          2)) if "distance" in upper and
"distance" in lower else 0,
                    "fuel": str(round(float(lower["fuel"]) + (
                            float(upper["fuel"]) - float(lower["fuel"])) *
ratio, 2)),
                    "ias": lower["ias"] if "ias" in lower else 0,
                    "tas": str(round(
                        float(lower["tas"]) + (float(upper["tas"]) -
float(lower["tas"])) * ratio,
                        2))
                })

            interpolated[temp] = temp_data

        return interpolated

    def interpolate_isa(self, data, target_isa):
        """
        Лінійна інтерполяція даних за значенням ISA.
        :param data: Словник з даними, де ключі верхнього рівня — значення ISA
(наприклад, '0', '10').
        :param target_isa: Цільове значення ISA для інтерполяції.
        :return: Інтерпольовані дані для заданого ISA.
        """
        # Конвертуємо ключі ISA в int для коректного сортування
        isa_keys = sorted(data.keys(), key=lambda x: int(x))
        target_isa_int = int(target_isa)

        # Якщо значення ISA вже є в даних, повертаємо його
        if str(target_isa) in isa_keys:
            return data[str(target_isa)]

        # Знаходимо найближчі значення ISA
        lower_isa, upper_isa = None, None
        for isa in isa_keys:
            isa_int = int(isa)
            if isa_int <= target_isa_int:
                lower_isa = isa
            elif isa_int > target_isa_int and upper_isa is None:
                upper_isa = isa
                break

        # Обробка крайніх випадків
        if not lower_isa:
```

```
            return data[isa_keys[0]]
        if not upper_isa:
            return data[isa_keys[-1]]

        # Коефіцієнт інтерполяції
        lower_isa_int = int(lower_isa)
        upper_isa_int = int(upper_isa)
        ratio = (target_isa_int - lower_isa_int) / (upper_isa_int -
lower_isa_int)

        # Інтерполяція параметрів для кожного FL
        interpolated = []
        lower_data = data[lower_isa]
        upper_data = data[upper_isa]

        # Збираємо спільні FL
        lower_fls = {entry['fl']: entry for entry in lower_data}
        upper_fls = {entry['fl']: entry for entry in upper_data}
        common_fls = sorted(
            set(lower_fls.keys()) & set(upper_fls.keys()),
            key=lambda x: int(x)
        )

        for fl in common_fls:
            lower_entry = lower_fls[fl]
            upper_entry = upper_fls[fl]
            interpolated_entry = {'fl': fl}

            # Інтерполяція кожного параметра
            for key in lower_entry:
                if key == 'fl':
                    continue
                if key in upper_entry:
                    try:
                        lower_val = float(lower_entry[key])
                        upper_val = float(upper_entry[key])
                        interpolated_val = lower_val + (upper_val - lower_val) *
ratio
                        interpolated_entry[key] = round(interpolated_val, 2)
                    except (ValueError, TypeError):
                        interpolated_entry[key] = lower_entry[key]
                else:
                    interpolated_entry[key] = lower_entry.get(key, 0)

            interpolated.append(interpolated_entry)

        return interpolated

    def show_graphs(self):
        if not hasattr(self, 'heights') or not self.heights:
            messagebox.showinfo("Інформація", "Спочатку виконайте розрахунок.")
            return

        # Створюємо нове вікно
        graph_window = tk.Toplevel(self.master)
        graph_window.title("Графіки польоту")
        graph_window.geometry("600x600")

        # Створюємо фігуру з двома графіками
        fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(7, 8))

        # Графік витрати палива за висотою
        ax1.plot(self.heights, self.fuels, 'b-o')
        ax1.set_title('Залежність витрати палива від висоти польоту')
        ax1.set_xlabel('Висота (FL)')
```

```
        ax1.set_ylabel('Витрата палива (кг)')
        ax1.grid(True)

        # Графік часу польоту за висотою
        ax2.plot(self.heights, self.times, 'g-o')
        ax2.set_title('Залежність часу польоту від висоти')
        ax2.set_xlabel('Висота (FL)')
        ax2.set_ylabel('Час польоту (хв)')
        ax2.grid(True)

        plt.tight_layout()

        # Вбудовуємо графік у Tkinter
        canvas = FigureCanvasTkAgg(fig, master=graph_window)
        canvas.draw()
        canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

        # Кнопка закриття
        btn_close = tk.Button(
            graph_window,
            text="Закрити",
            command=graph_window.destroy
        )
        btn_close.pack(pady=10)


if __name__ == "__main__":
    root = tk.Tk()
    app = AdvancedFuelCalculator(root)
    root.mainloop()
```

Файл calculate_route_profile.py

```
import json
import math
import requests


def calculate_route_segments(route_points):
    """
    Функція обчислює довжини відрізків маршруту та різниці висот між
послідовними точками.
    Приймає список ідентифікаторів точок маршруту (рядки), шукає їх координати
та висоту
    у файлах airports_esp.json та vertices_esp.json. Повертає список словників з
інформацією
    про кожен відрізок маршруту.
    """
    # Завантажуємо дані аеропортів та вершин
    with open('src/airports_esp.json', 'r', encoding='utf-8') as f:
        airports_data = json.load(f)
    with open('src/vertices_esp.json', 'r', encoding='utf-8') as f:
        vertices_data = json.load(f)

    route_points = route_points.split()
    # Перетворюємо списки на словники за ключем 'ident' для швидкого пошуку
    airports = {entry['ident']: entry for entry in airports_data}
    vertices = {entry['ident']: entry for entry in vertices_data}
```

```
    # Допоміжний список для збереження (ident, lat, lon, alt)
    points = []
    for ident in route_points:
        if ident in airports:
            data = airports[ident]
            lat = data.get('latitude')
            lon = data.get('longitude')
            alt = data.get('altitude')
            if lat is None or lon is None or alt is None:
                raise ValueError(f"Недостатньо даних для точки {ident} в
airports_esp.json")
            points.append((ident, lat, lon, alt))
        elif ident in vertices:
            data = vertices[ident]
            lat = data.get('latitude')
            lon = data.get('longitude')
            if lat is None or lon is None:
                raise ValueError(f"Недостатньо даних для точки {ident} в
vertices_esp.json")
            alt = 0
            points.append((ident, lat, lon, alt))
        else:
            raise ValueError(f"Точка {ident} не знайдена у файлах даних")

    distance_km = 0
    altitude_fl_start = 0
    altitude_fl_end = 0

    for i in range(len(points) - 1):
        ident1, lat1, lon1, alt1 = points[i]
        ident2, lat2, lon2, alt2 = points[i + 1]
        # Конвертація координат з градусів в радіани
        phi1 = math.radians(lat1)
        phi2 = math.radians(lat2)
        dphi = math.radians(lat2 - lat1)
        dlambda = math.radians(lon2 - lon1)
        # Обчислюємо гаверсинус
        a = math.sin(dphi / 2) ** 2 + math.cos(phi1) * math.cos(phi2) *
math.sin(dlambda / 2) ** 2
        c = 2 * math.asin(math.sqrt(a))
        earth_radius_km = 6371.0  # Радіус Землі в км
        distance = earth_radius_km * c
        if alt1:
            altitude_fl_start = alt1 / 1000 * 32.808
        elif alt2:
            altitude_fl_end = alt2 / 1000 * 32.808

        distance_km += distance

    return distance_km, altitude_fl_start, altitude_fl_end
```

Додаток Б Інструкція користувача

Проєкт складається з двох програмних файлів:

1) main.py – основна програма
2) calculate_route_profile.py – додатковий скрипт для обрахунку дистанцій між точками

Інструкція з використання програми

Відкрити файл main.exe з папки dict, або запустити напряму код з файла main.py, перед запуском потрібно встановити такі бібліотеки: json, tkinter, numpy, PIL, matplotlib