ЧИСЛОВИЙ АНАЛІЗ КЕРОВАНОГО РУХУ ДРОНА-КВАДРОКОПТЕРА

(Шифр - «Керований рух»)

ЗМІСТ

ВСТУП

В останні двадцять років тема теоретичного дослідження поведінки безпілотних літаючих апаратів (далі БПЛА) різної степені складності та призначення, в тому числі й дронів-квадрокоптерів набула не аби якого розмаху. В останні десять років БПЛА почали з'являтись у вигляді рішень прикладних задач, які представляють складнощі за безпосередньої участі людини чи іншої техніки із використанням пілота. В наш час БПЛА проникли в значну кількість сфер життєдіяльності й стали її невід'ємною частиною. В задачах фото- та відео- зйомки, тобто для задач фіксації БПЛА використовуються як звичайними людьми, що обирають для своїх потреб засоби, що є в вільному продажі, так і професіоналами, що обирають професійні рішення для певної спеціалізації що мають відповідати спеціальним потребам збору різноманітних даних з місцевості. Також БПЛА використовуються в розважальних цілях, наприклад, для гонок дронів, фрістайлу, аерофотозйомки під час екстремальних видів спорту, таких як серфінг або лижі. Деякі інноваційні компанії розробляють системи доставки товарів за допомогою БПЛА. Це може бути доставка харчових продуктів, медикаментів або невеликих посилок та вантажів безпосередньо до пункту призначення у віддалених або важкодоступних місцях. БПЛА використовуються також в сільському господарстві для моніторингу росту рослин, оцінки стану посівів, розподілу ресурсів у сільському господарстві. Вони дозволяють збирати дані про вологість, урожайність та стан посівів, що сприяє оптимізації сільськогосподарських процесів. Різноманітні БПЛА використовуються й в інших сферах, зокрема в військовій для різноманітних задач, як тактичні пристрої що досліджують місцевість та сили супротивника, чи як системи скиду або доставки боєкомплекту.

На даний момент найпопулярніший вид БПЛА – дрон-квадрокоптер, адже має досить просту конструкцію, та великий спектр задач, які можуть бути вирішені за його допомогою. Однак, керування рухом дрона-квадрокоптера є складною задачею, яка вимагає розробки ефективних алгоритмів та методів для

досягнення бажаних результатів. Чисельний аналіз керованого руху дрона-квадрокоптера стає важливим інструментом для дослідження його поведінки та оптимізації траєкторій його польоту. Тут на допомогу може прийти теорія керування та її непрямі методи, що спрямовані на розв'язок поставленої крайової задачі необхідних умов оптимальності принципу максимуму Л.С. Понтрягіна із врахуванням умов на керування та обмежень на степені сподоби дрона, що описують його поведінку в просторі, так звані фазові обмеження. Поряд із принципом максимуму в роботі застосовується метод штрафних функціоналів, оптимізаційний метод умовного градієнта та метод Рунге-Кутта четвертого порядку точності для розв'язання виникаючих допоміжних задач Коші. Після вдалої комбінації математичної постановки задачі керованого руху дрона-квадрокоптера та методів теорії керування та методів теорії оптимізації, очікується отримання дієвого підходу для розв'язання задачі знаходження оптимального керування для оптимальної траєкторії польоту.

Актуальність теми: розглянутий підхід числового аналізу руху дрона-квадрокоптера є істотним фундаментом для подальшого дослідження поведінки дронів-квадрокоптерів чи інших БПЛА та може використовуватись для розв'язання різноманітних прикладних задач керування БПЛА як більшої так і меншої складності.

Мета роботи: провести числовий аналіз запропонованої математичної моделі керованого руху дрона-квадрокоптера; запропонувати та застосувати інструменти теорії керування, методи оптимізації та чисельних методів для розв'язання задачі оптимізації траєкторії польоту дрона-квадрокоптера.

Завдання роботи:

1. Дослідити будову та фізичний зміст поведінки дрона-квадрокоптера;

2. Дослідити відомі математичні моделі руху дрона-квадрокоптера, запропонувати математичну модель;

3. Поставити задачу оптимального керування рухом дрона-квадрокоптера;

4. Розв’язати програмно поставлену задачу за допомогою інструментів теорії керування та методів оптимізації;

5. Проаналізувати отримані результати.

Об’єкт дослідження: математична модель керованого руху дрона-квадрокоптера.

Предмет дослідження: інструменти теорії керування та теорії оптимізації, що застосовуються до математичної моделі керованого руху дрона-квадрокоптера.

Методи дослідження: методи теорії керування – метод максимуму Понтрягіна, метод штрафних функціоналів; метод теорії оптимізації – метод умовного градієнта; метод розв’язання задачі Коші – метод Рунге-Кутта четвертого порядку точності.

Дана наукова робота складається зі вступу що розкриває актуальність досліджуваної проблеми, також із наступних трьох розділів: математичної моделі керованого руху дрона-квадрокоптера що досліджується протягом кваліфікаційної роботи, постановки задачі оптимального керування рухом дрона-квадрокоптера де детально освітлені усі аспекти постановки задачі оптимального керування досліджуваної математичної моделі, розробки програмного забезпечення розв’язання задачі керованого руху дрона-квадрокоптера де освітлено деталі реалізації програмного продукту та отримані за його допомогою результати. Також робота містить висновки, перелік використаних під час дослідження джерел та додатки.

1 МАТЕМАТИЧНА МОДЕЛЬ КЕРОВАНОГО РУХУ ДРОНА-КВАДРОКОПТЕРА

Математична модель в даній кваліфікаційній роботі розглянута відповідно до роздумів за джерелом [1].

1.1 Постановка задачі про керований рух дрона-квадрокоптера

Побудуємо математичну модель руху дрона у вигляді системи звичайних диференціальних рівнянь, відповідно до [2]:

$$\dot{X}(t) = f\big(X(t), u(t)\big), \quad t_0 \le t \le T; \tag{1.1}$$

де $X(t) = (x_1(t), x_2(t), \dots, x_n(t))^T$ – шукана векторна функція на $[t_0; T]$, яка характеризує процес керованого руху дрона і відображає всі ступені свободи дрона, а саме, положення його центру мас та орієнтацію в просторі в кожен момент часу t $(t_0 \le t \le T)$; $u(t) = (u_1(t), u_2(t), u_3(t), u_4(t))^T$ – шуканий вектор параметрів керування, що визначають хід процесу і являють собою зусилля кожного з чотирьох моторів дрона в проміжку часу $[t_0; T]$; $f(X(t), u(t)) = (f_1(X(t), u(t)),\ f_2(X(t), u(t)), \dots, f_n(X(t), u(t)))^T$ – відома векторна функція, яка описує внутрішній устрій дрона і враховує вплив зовнішніх факторів. Будемо вважати, що межі t_0 та T проміжку змінення часу фіксовані.

1.2 Рівняння режимів руху

Розглянутий дрон-квадрокоптер може літати лише в чотирьох режимах: зависання, крен, тангаж, рискання. Математична модель міститиме узагальнені рівняння руху відповідно до законів аеродинаміки. З точки зору аеродинаміки, нам знадобляться теорія моментів, теорія конструкції та дії гвинтів за джерелом [3]. Відповідно до теорії моментів, ротор моделюється як тонкий диск, оберт якого спричиняє постійну швидкість вздовж осі обертання без врахування тертя за джерелами [4], [5]. Усі аеродинамічні сили і моменти, що діють на роторі, визначаються за допомогою теорії дії гвинтів. Наведена аеродинамічна модель буде відповідно до джерела [6] допускати такі фактори: повітря є ідеальним

нестиглім газом; товщина диску – нескінченно мала величина; вертикальна швидкість повітря постійна навколо ротора; ротори – жорсткі; сила, паралельна валу ротора, визначається як тяга ротора $T \equiv T(t)$, а сила, перпендикулярна осі ротора, визначається як сила маточини $T_C \equiv T_C(t),\ t_0 \leq t \leq T$. Діючі моменти на роторі є гальмівними $M_T \equiv M_T(t)$ і рухомими $M_П \equiv M_П(t),\ t_0 \leq t \leq T$.

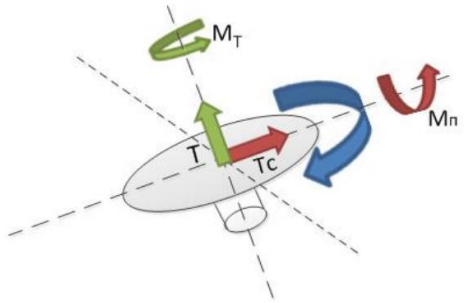


Рисунок 1.1 – Аеродинамічні сили й моменти, що діють на ротор

Як вже відомо, дрон-квадрокоптер представляє собою доволі легкий хрестоподібний симетричній каркас, який пов’язує чотири ротори – двигуни дрона, які значно важчі за сам каркас дрона. Ротори працюють за принципом поперечної конфігурації, кожен гвинт яких під’єднаний за допомогою редуктору; усі оберти гвинтів жорстко фіксовані, паралельні та мають фіксовані кроки обертання. Зрозуміло, що потоки повітря спрямовані до низу, аби дрон мав підйомну силу вгору.

Отже, маючи таку аеродинамічну модель, можемо дійти очевидного висновку, що керування дроном може здійснюватися лише за допомогою регулювання швидкості обертання гвинтів дрона. Для визначеності пронумеруємо ротори дрона відповідно до площини $O_bX_bY_b$:

- Ротор 1 – передній ротор, вздовж додатного напряму O_bX_b;
- Ротор 2 – лівий ротор, вздовж від’ємного напрямку O_bY_b;
- Ротор 3 – заний ротор, вздовж від’ємного напрямку O_bX_b;
- Ротор 4 – правий ротор, вздовж додатного напрямку O_bY_b.

Зазначимо, що передній-задній гвинти обертаються проти годинникової стрілки, лівий-правий гвинти обертаються за годинниковою стрілкою (рис. 1.2).

Така конфігурація з дозволяє дрону-квадрокоптеру досягнути рівноваги моментів інерції, через що дрон не потребує додаткового хвостового гвинта, як у випадку звичайного гелікоптера.

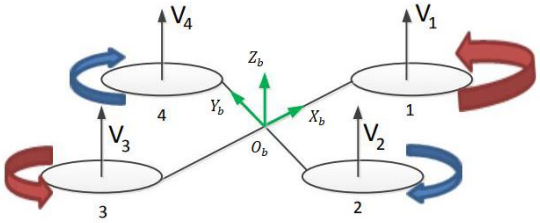


Рисунок 1.2 – Спрощена швидкісна схема квадрокоптера

Відповідно гвинт кожного з роторів може обертатись з деякою кутовою швидкістю $\Omega_i \equiv \Omega_i(t)$ (в рад $* \text{с}^{-1}$), $i = \overline{1,4}$; $t_0 \leq t \leq T$.

Запишемо режими поведінки дрона (зависання, крен, тангаж, рискання) виходячи із значень кутової швидкості гвинтів дрона:

<u>Зависання:</u>

$$U_1 = \sum_{i=1}^{4} (\Omega_i + \delta_A) * sign(\Omega_i), \quad t_0 \leq t \leq T, \tag{1.2}$$

де $U_1 \equiv U_1(t)$ – позначка режиму зависання; δ_A (в рад $* \text{с}^{-1}$) – додана додатна величина. Схематичне зображення режиму зависання подано на рис. 1.2.

Таблиця 1.1 – Схематичне зображення режиму зависання (відносні позначки швидкості обертання гвинтів Ω_i, $i = \overline{1,4}$: « ↑↑ », « ↑ », «0»)

Ω_1	Ω_2	Ω_3	Ω_4
↑↑	↑↑	↑↑	↑↑

<u>Крен:</u>

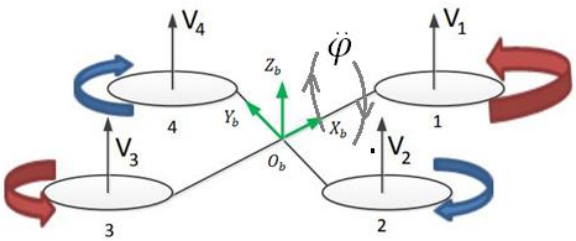


Рисунок 1.3 – Режим крену квадрокоптера

$$U_2 = \Omega_1 - (\Omega_2 + \delta_B) + \Omega_3 - (\Omega_4 + \delta_B), \quad t_0 \le t \le T, \qquad (1.3)$$

де $U_2 \equiv U_2(t)$ – позначка режиму крену; δ_B (в рад $* \text{с}^{-1}$) – додатна додана величина. Схематичне зображення режиму крену подано на рис. 1.3.

Таблиця 1.2 – Схематичне зображення режиму крену

Ω_1	Ω_2	Ω_3	Ω_4
↑↑	↑↑ або ↑	↑↑	↑ або ↑↑

<u>Тангаж:</u>

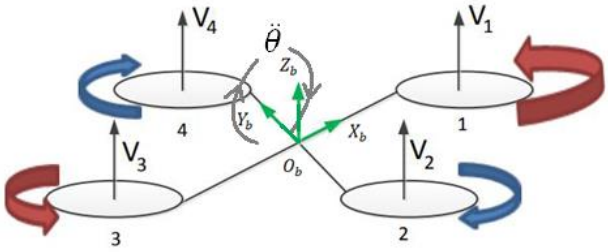


Рисунок 1.4 – Режим тангажу квадрокоптера

$$U_3 = (\Omega_1 - \delta_B) - \Omega_2 + (\Omega_3 + \delta_A) - \Omega_4, \quad t_0 \le t \le T, \qquad (1.4)$$

де $U_3 \equiv U_3(t)$ – позначка режиму тангажу. Схематичне зображення режиму тангажу подано на рис. 1.4.

Таблиця 1.3 – Схематичне зображення режиму крену

Ω_1	Ω_2	Ω_3	Ω_4
↑↑ або ↑	↑↑	↑ або ↑↑	↑↑

<u>Рискання:</u>

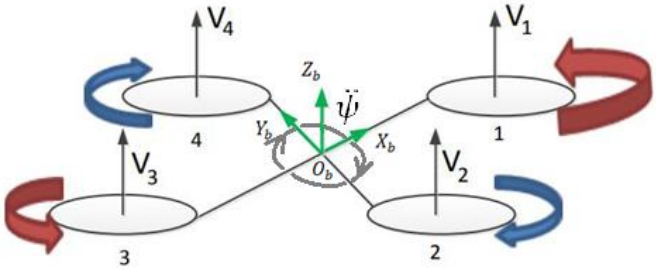


Рисунок 1.5 – Режим рискання квадрокоптера

$$U_4 = (\Omega_1 - \delta_B) - (\Omega_2 + \delta_\text{A}) + (\Omega_3 + \delta_\text{B}) - (\Omega_4 + \delta_\text{A}), \quad t_0 \le t \le T, \quad (2.5)$$

де $U_4 \equiv U_4(t)$ – позначка режиму тангажу. Схематичне зображення режиму тангажу подано на рис. 1.5.

Таблиця 1.4 – Схематичне зображення режиму рискання

Ω_1	Ω_2	Ω_3	Ω_4
↑↑ або ↑	↑ або ↑↑	↑↑ або ↑	↑ або ↑↑

Детально розглянуто вивід математичної моделі руху БПЛА типу дрон-квадракоптер буде в Додатку А. В ході подальших роздумів будемо посилатись на формули та рисунки, додаючи літеру додатку, тобто наприклад тактм чином: (1.1А) та рис. 1.1Б відповідно.

Наступним кроком застосуємо граничні умови та обмеження математичної моделі руху БПЛА для системи (1.40А).

1.3 Граничні умови та обмеження в математичній моделі руху БПЛА

Цілком логічно, що поставлена математична модель руху БПЛА для постановки задачі керування рухом дрона-квадрокоптера має містити певні обмеження для змінних стану та для керуючих змінних.

Для вектора керування $u = (u_1, u_2, u_3, u_4)^T$ оберемо такі обмеження, що відповідають мінімальній та максимальній потужності керування у кожен момент часу t ($t_0 \le t \le T$):

$$u_{min} \le u_i(t) \le u_{max}, \quad i = \overline{1,4}, \quad (1.6)$$

де $u_{min} \ge 0$ та $u_{max} > 0$ – задані сталі.

Для вектора положення дрона $P = (x, y, z)^T$ на часовому проміжку $(t_0;\ T)$ обмеження ставитись не будуть.

Для вектора поступальної швидкості дрона $V = (\dot{x}, \dot{y}, \dot{z})^T$ будуть вводитись такі обмеження:

$$\dot{x}(t) \ge 0, \quad (1.7)$$

$$\dot{y}(t) \ge 0, \quad (1.8)$$

$$\dot{z}(t) \geq 0, \quad t_0 \leq t \leq T. \tag{1.9}$$

Для вектора орієнтації дрона $E = (\varphi, \theta, \psi)^T$ будуть введені наступні обмеження:

$$-\frac{\pi}{2} \leq \varphi(t) \leq \frac{\pi}{2}; \tag{1.10}$$

$$-\frac{\pi}{2} \leq \theta(t) \leq \frac{\pi}{2}; \tag{1.11}$$

$$-\frac{\pi}{2} \leq \psi(t) \leq \frac{\pi}{2}, \quad t_0 \leq t \leq T. \tag{1.12}$$

Для вектора кутової швидкості дрона $W = (p, q, r)^T$ мають вводитись такі обмеження:

$$p \geq 0, \tag{1.13a}$$

$$q \geq 0, \tag{1.14a}$$

$$r \geq 0, \tag{1.15a}$$

на підставі зв'язку (1.33А), замість умов (1.13а) – (1.15а) висунемо такі умови:

$$\dot{\varphi}(t) \geq 0; \tag{1.13}$$

$$\dot{\theta}(t) \geq 0; \tag{1.14}$$

$$\dot{\psi}(t) \geq 0, \quad t_0 \leq t \leq T, \tag{1.15}$$

Вочевидь умови (1.7)-(1.15) формуватимуть множину фазових обмежень, детальніше про це – в розділі 1.4.

Тепер перейдемо до формування режимів на кінцях траєкторії в задачі керування рухом дрона-квадрокоптера. Умови мають визначати змінні стану дрона на початку та у кінці процесу керування дроном, тобто в моменти часу $t = t_0$ та $t = T$. Варто зазначити, що усі крайові умови будуть визначені виходячи з формальних роздумів: дрон на початку керованого руху знаходиться в деякій точці простору із координатами (x_0, y_0, z_0) в положенні рівноваги та в кінці керування має знаходитися в положенні рівноваги в деякій точці (x_1, y_1, z_1):

$$x(t_0) = x_0, \quad y(t_0) = y_0, \quad z(t_0) = z_0, \tag{1.16}$$

$$x(T) = x_1, \quad y(T) = y_1, \quad z(T) = z_1, \tag{1.17}$$

$$\dot{x}(t_0) = 0, \quad \dot{y}(t_0) = 0, \quad \dot{z}(t_0) = 0, \tag{1.18}$$

$$\dot{x}(T) = 0, \qquad \dot{y}(T) = 0, \qquad \dot{z}(T) = 0, \tag{1.19}$$

$$\varphi(t_0) = 0, \qquad \theta(t_0) = 0, \qquad \psi(t_0) = 0, \tag{1.20}$$

$$\varphi(T) = 0, \qquad \theta(T) = 0, \qquad \psi(T) = 0, \tag{1.21}$$

$$\dot{\varphi}(t_0) = 0, \qquad \dot{\theta}(t_0) = 0, \qquad \dot{\psi}(t_0) = 0, \tag{1.22}$$

$$\dot{\varphi}(T) = 0, \qquad \dot{\theta}(T) = 0, \qquad \dot{\psi}(T) = 0, \tag{1.23}$$

1.4 Формулювання оптимізаційної задачі для розглянутої моделі руху БПЛА

Для зручності запису здійснимо перехід до нової невідомої функції керування $\bar{u}(t) = \left(\bar{u}_1(t), \bar{u}_2(t), \bar{u}_3(t), \bar{u}_4(t)\right)$, $t_0 \le t \le T$ у математичній моделі руху БПЛА, поклавши

$$\bar{u}_i(t) = \frac{u_i(t) - u_{min}}{u_{max} - u_{min}}, \quad i = \overline{1,4}; \quad t_0 \le t \le T, \tag{1.24}$$

звідки, відповідно до (1.6) матимемо такі обмеження на нові керуючі змінні:

$$0 \le \bar{u}_i(t) \le 1, \quad i = \overline{1,4}; \quad t_0 \le t \le T.$$

Далі, користуючись формулами

$$u_i(t) = \bar{u}_i(t)(u_{max} - u_{min}) + u_{min}, \quad i = \overline{1,4}; \quad t_0 \le t \le T.$$

та позначеннями

$$\bar{A}_x = \frac{A_x}{m}, \quad \bar{A}_y = \frac{A_y}{m}, \quad \bar{A}_z = \frac{A_z}{m}; \quad K = \frac{k_L}{m}(u_{max} - u_{min}); \tag{1.25}$$

$$\bar{J}_x = \frac{k_L}{J_x}(u_{max} - u_{min}), \quad \bar{J}_y = \frac{k_L}{J_y}(u_{max} - u_{min}), \quad \bar{J}_z = \frac{b}{J_z}(u_{max} - u_{min});$$

$$c = \frac{4u_{min}}{u_{max} - u_{min}},$$

замість системи диференціальних рівнянь (1.5) отримаємо на $[t_0; T]$ систему

$$\begin{cases} \ddot{x} = g\theta - \bar{A}_x\dot{x}, \\ \ddot{y} = -g\varphi - \bar{A}_y\dot{y}, \\ \ddot{z} = K((\bar{u}_1 + \bar{u}_2 + \bar{u}_3 + \bar{u}_4) + c) - g - \bar{A}_z\dot{z}, \\ \ddot{\varphi} = \bar{J}_x(\bar{u}_4 - \bar{u}_2), \\ \ddot{\theta} = \bar{J}_y(\bar{u}_3 - \bar{u}_1), \\ \ddot{\psi} = \bar{J}_z(-\bar{u}_1 + \bar{u}_2 - \bar{u}_3 + \bar{u}_4). \end{cases} \tag{1.26}$$

Під час подальших розглядів іноді тимчасово опускатимемо позначку $\bar{\ }$ у змінних керування та величин, введених в (1.25), там, де це не буде викликати різночитань.

Для формулювання задачі керування рухом дрона-квадракоптера необхідно сформулювати цільовий функціонал, інтерпретація математичного змісту якого може сприйматись як критерій оптимальності керування. Відомо, що через конструкційні особливості каркас дрона-квадракоптера має бути значно легшим за ротори на кінцях каркасу, через що вага каркасу є дуже обмеженою. Відомо також, що дистанційний дрон-квадракоптер має електричне живлення електромоторів від акумулятора, який розміщено в каркасі дрона. Виходячи з цих роздумів, можемо дійти висновку, що якісний дрон має оптимально використовувати заряд акумулятора для здійснення польоту, переміщення, маневрів, аби подовжити час своєї роботи; в противному випадку довелось би збільшувати об'єм акумулятора, що звісно, виходячи з конструкційних особливостей, не є якісним рішенням.

Отже, сформулюємо цільовий функціонал, який характеризує загальну інтенсивність використання роторів із плином часу:

$$J(\bar{u}) = \int_{t_0}^{T} \sum_{i=1}^{4} \bar{u}_i^2(t)\, dt\,; \tag{1.27}$$

і поставимо задачу мінімізації функціонала (1.27):

$$J(\bar{u}) \to \min_{\bar{u} \in U} \quad ; \tag{1.28}$$

на множині допустимих керувань

$$U = \{\bar{u} \equiv \bar{u}(t) \in L_2^{(4)}[t_0;T]:\ 0 \le \bar{u}_i(t) \le 1,\ \ i = \overline{1,4};\ \ \ t_0 \le t \le T\}, \tag{1.29}$$

де $L_2^{(4)}[t_0;T] = \underbrace{L_2[t_0;T] \times \ldots \times L_2[t_0;T]}_{\text{4 множники}}$ – гільбертів простір векторних функцій $u \equiv u(t) = (u_1(t), u_2(t), u_3(t), u_4(t)),\ \ u_i(t) \in L_2[t_0;T],\ i = \overline{1,4},$ зі скалярним добутком

$$(u,v)_{L_2^{(4)}[t_0;T]} = \int_{t_0}^{T} \sum_{i=1}^{4} u_i(t) \cdot v_i(t)\, dt, \quad \forall u(t), v(t) \in L_2^{(4)}[t_0;T]; \tag{1.29a}$$

та нормою елемента

$$\|u\|_{L_2^{(4)}[t_0;T]} = \sqrt{(u,u)_{L_2^{(4)}[t_0;T]}} = \sqrt{\int_{t_0}^{T} \sum_{i=1}^{4} u_i^2(t)\, dt}, \quad \forall u(t) \in L_2^{(4)}[t_0;T]. \ (1.29b)$$

за наявності диференціальних зв'язків (1.26), крайових умов (1.16) – (1.23) і фазових обмежень (1.7) – (1.15). Отже, маємо постановку задачі про оптимальний керований рух дрона-квадрокоптера.

Зазначимо, що функціонал (1.27), вочевидь, є обмеженим знизу ($J^* = \min_{\bar{u} \in U} J(\bar{u}) = 0$), квадратичним (а, отже, сильно опуклим і неперервним) на опуклій замкненій обмеженій множині (1.29) з гільбертового простору $L_2^{(4)}[t_0;T]$. [9]

<u>Означення.</u> Функціонал $J(u)$, визначений на опуклій множині U банахового простору, називають сильно опуклим на U, якщо існує така стала $\chi > 0$, що виконується нерівність

$$J(\alpha u + (1-\alpha)v) \leq \alpha\, J(u) + (1-\alpha) J(v) - \ \chi \cdot \alpha(1-\alpha)\|u - v\|^2.$$

для всіх $u, v \ \in U$ та всіх α, $0 \leq \ \alpha \leq 1$.

Зазначені властивості цільового функціонала дозволяють стверджувати, на підставі узагальненої теореми Вейєрштрасса [9], що функціонал (1.27) досягає своєї нижньої грані на множині (1.29).

<u>Теорема Веєрштрасса (узагальнена)</u>. Опуклий неперервний функціонал на замкненій обмеженій опуклій множині з гільбертового простору досягає на цій множині своєї нижньої грані.

Отже, можемо стверджувати, що задача про оптимальний керований рух дрона-квадрокоптера має розв'язок, до того ж єдиний.

2 ПОСТАНОВКА ЗАДАЧІ ОПТИМАЛЬНОГО КЕРУВАННЯ РУХОМ ДРОНА-КВАДРОКОПТЕРА

2.1 Зведення постановки здачі про оптимальний керований рух дрона-квадрокоптера до канонічної (понтрягінської) форми

Уведемо до розгляду фазовий простір розмірності $n = 12$ і позначимо через $X(t) = (x_1(t), x_2(t), \dots, x_n(t))^T$, $t_0 \le t \le T$ векторну функцію змінних стану (фазових координат) керованого об'єкту, де

$$\begin{aligned} &x_1(t) = x(t), \quad x_2(t) = \dot{x}(t), \quad x_3(t) = y(t), \quad x_4(t) = \dot{y}(t), \\ &x_5(t) = z(t), \quad x_6(t) = \dot{z}(t), \quad x_7(t) = \varphi(t), \quad x_8(t) = \dot{\varphi}(t), \\ &x_9(t) = \theta(t), \quad x_{10}(t) = \dot{\theta}(t), \quad x_{11}(t) = \psi(t), \quad x_{12}(t) = \dot{\psi}(t). \end{aligned} \tag{2.1}$$

Тоді, враховуючи (1.26), отримаємо на $[t_0; T]$ наступну систему диференціальних рівнянь першого порядку:

$$\begin{cases} \dot{x}_1(t) = x_2(t), \\ \dot{x}_2(t) = g x_9(t) - \bar{A}_x x_2(t), \\ \dot{x}_3(t) = x_4(t), \\ \dot{x}_4(t) = -g x_7(t) - \bar{A}_y x_4(t), \\ \dot{x}_5(t) = x_6(t), \\ \dot{x}_6(t) = K((\bar{u}_1 + \bar{u}_2 + \bar{u}_3 + \bar{u}_4) + c) - g - \bar{A}_z x_6(t), \\ \dot{x}_7(t) = x_8(t), \\ \dot{x}_8(t) = \bar{J}_x(\bar{u}_4 - \bar{u}_2), \\ \dot{x}_9(t) = x_{10}(t), \\ \dot{x}_{10}(t) = \bar{J}_y(\bar{u}_3 - \bar{u}_1), \\ \dot{x}_{11}(t) = x_{12}(t), \\ \dot{x}_{12}(t) = \bar{J}_z(-\bar{u}_1 + \bar{u}_2 - \bar{u}_3 + \bar{u}_4), \end{cases} \tag{2.2}$$

яка з урахуванням позначень для векторної функції правих частин $f(X(t), \bar{u}(t)) = (f_1(X(t), \bar{u}(t)), \dots, f_n(X(t), \bar{u}(t)))^T$:

$$\begin{aligned} &f_1(X(t), \bar{u}(t)) = x_2(t); \quad f_2(X(t), \bar{u}(t)) = g x_9(t) - \bar{A}_x x_2(t); \\ &f_3(X(t), \bar{u}(t)) = x_4(t); \quad f_4(X(t), \bar{u}(t)) = -g x_7(t) - \bar{A}_y x_4(t); \\ &f_5(X(t), \bar{u}(t)) = x_6(t); \\ &f_6(X(t), \bar{u}(t)) = K((\bar{u}_1 + \bar{u}_2 + \bar{u}_3 + \bar{u}_4) + c) - g - \bar{A}_z x_6(t); \\ &f_7(X(t), \bar{u}(t)) = x_8(t); \quad f_8(X(t), \bar{u}(t)) = \bar{J}_x(\bar{u}_4 - \bar{u}_2); \\ &f_9(X(t), \bar{u}(t)) = x_{10}(t); \quad f_{10}(X(t), \bar{u}(t)) = \bar{J}_y(\bar{u}_3 - \bar{u}_1); \end{aligned} \tag{2.3}$$

$$f_{11}(X(t),\bar{u}(t)) = x_{12}(t); \quad f_{12}(X(t),\bar{u}(t)) = \bar{J}_z(-\bar{u}_1 + \bar{u}_2 - \bar{u}_3 + \bar{u}_4),$$

може бути записана у векторній формі (1.1).

До системи диференціальних рівнянь (2.2) приєднаємо крайові умови (1.16) – (1.23), фазові обмеження (1.7) – (1.15) і, з урахуванням введених позначень, сформулюємо наступну задачу оптимального керування.

Задача А. Знайти нижню грань функціонала

$$J(\bar{u}) = \int_{t_0}^{T} \sum_{i=1}^{4} \bar{u}_i^2(t)\, dt; \tag{2.4}$$

за умов

$$\dot{X}(t) = f(X(t),\bar{u}(t)), \quad t_0 \le t \le T; \tag{2.5}$$

$$X(t_0) = X^{(0)}; \tag{2.6}$$

$$X(T) = X^{(1)}; \tag{2.7}$$

$$x_{2i}(t) \ge 0, \quad i = \overline{1,6}; \quad t_0 \le t \le T; \tag{2.8}$$

$$-\frac{\pi}{2} \le x_i(t) \le \frac{\pi}{2}, \quad i = 7,9,11; \quad t_0 \le t \le T; \tag{2.9}$$

$$U = \{\bar{u} \equiv \bar{u}(t) \in L_2^{(4)}[t_0;T]:\ 0 \le \bar{u}_i(t) \le 1,\ i = \overline{1,4};\ t_0 \le t \le T\}, \tag{2.10}$$

де $f(X(t),\bar{u}(t)) = (f_1(X(t),\bar{u}(t)),\dots,f_n(X(t),\bar{u}(t)))^T$ – відома векторна функція, визначена в (2.3), яка, вочевидь, є неперервною за сукупністю аргументів разом із частинними похідними першого порядку за умов (2.6)-(2.10); $X(t) = (x_1(t),\dots,x_n(t))^T$ і $\bar{u}(t) = (\bar{u}_1(t),\dots,\bar{u}_4(t))$ – шукані векторні функції на $[t_0;T]$; t_0, T, g, c, K, $\bar{A}_x$, $\bar{A}_y$, $\bar{A}_z$, $\bar{J}_x$, $\bar{J}_y$, $\bar{J}_z$ – задані додатні сталі; $X^{(0)} \equiv (x_0,0,y_0,0,z_0,0,\dots,0)^T$ та $X^{(1)} \equiv (x_1,0,y_1,0,z_1,0,\dots,0)^T$ – відомі вектори з $\mathbb{R}^n$.

Задача А є автономною задачею оптимального керування з квадратичним функціоналом, лінійними диференціальними зв'язками, закріпленими кінцями траєкторії, фазовими обмеженнями, обмеженнями на керуючий вплив та фіксованим часом.

2.2 Застосування методу штрафних функціоналів до розв'язання задачі оптимального керування рухом дрона-квадрокоптера

Для розв'язання задачі А, сформульованої у підрозділі 2.1, будемо застосовувати метод штрафних функціоналів, який дозволяє звести її до задачі оптимального керування *з вільним правим кінцем* за рахунок введення «штрафів» на умови (2.7) на правому кінці та на фазові обмеження (2.8), (2.9). Слід зазначити, що для задач оптимального керування з вільним кінцем розроблені ефективні наближені способи їх розв'язання, які використовують важливу властивість цього класу задач: для отримання точного розв'язку задачі оптимального керування динамічною системою, якщо вона лінійна за фазовими змінними і на правий кінець не накладено жодних обмежень, достатньо розв'язати дві задачі Коші. Через це задачі оптимального керування з вільним кінцем, лінійні відносно фазових координат, відіграють роль основних елементів для побудови ітераційних схем розв'язання складних задач оптимального керування [12].

Для мінімізації функціонала (2.4) за умов (2.5)-(2.10) метод штрафних функціоналів полягає в тому, що замість задачі мінімізації основного функціонала $J(\bar{u})$ розв'язують задачі мінімізації послідовності $\{J^{(m)}(\bar{u})\}_{m=1}^{\infty}$ допоміжних функціоналів

$$J^{(m)}(\bar{u}) = J(\bar{u}) + P^{(m)}(\bar{u}), \quad \bar{u} \in U, \ \ m = 1,2,\ldots, \qquad (2.11)$$

де $P^{(m)}(\bar{u})$ $(m = 1,2,\ldots)$ – так звані штрафні функціонали. Задачі (2.11) конструюють так, щоб послідовність $\{\bar{u}^{*(m)}\}_{m=1}^{\infty}$ їх оптимальних розв'язків збігалася до оптимального розв'язку $\bar{u}^* \in U$ вихідної задачі при $m \to \infty$. Згідно з методом зовнішніх штрафів, штрафні функціонали вибирають так, щоб за межами допустимої множини елементи послідовно зі зростанням m ставали все більш «невигідними» і або $P^{(m)}(\bar{u}) = 0$ для всіх $\bar{u} \in U$, або $P^{(m)}(\bar{u}) \to 0$ при $m \to \infty$ для будь-яких $\bar{u} \in U$.

Для зняття обмеження (2.7), накладеного на правий кінець траєкторії в задачі А, можна використати штраф

$$P_1{}^{(m)}(\bar{u}) = m \cdot \left\| X(T;\bar{u}) - X^{(1)} \right\|^2 = m \cdot \sum_{i=1}^{n} (x_i(T;\bar{u}) - x_i{}^{(1)})^2, \qquad (2.12)$$

$$\bar{u} \in U; \quad m = 1,2,\ldots$$

За великих значень константи m мінімум функціонала (2.11) буде досягатися при $x_i(T;\bar{u}) - x_i{}^{(1)} \approx 0$, $i = \overline{1,n}$, оскільки через структуру функціонала $J^{(m)}(\bar{u})$ значна похибка у виконанні хоча б однієї з умов $x_i(T;\bar{u}) - x_i{}^{(1)} = 0$, $i = \overline{1,n}$ призведе до значного зростання функціонала $J^{(m)}(\bar{u})$.

Для зняття фазових обмежень (2.8) можна використати, наприклад, один з таких штрафів:

$$P_2{}^{(m)}(\bar{u}) = m \cdot \int_{t_0}^{T} \sum_{i=1}^{6} \max\{-x_{2i}(t;\bar{u});\ 0\}\, dt, \quad \bar{u} \in U;\ m = 1,2,\ldots$$

або

$$P_2{}^{(m)}(\bar{u}) = \frac{1}{m} \cdot \int_{t_0}^{T} \sum_{i=1}^{6} e^{-m \cdot x_{2i}(t;\bar{u})}\, dt, \quad \bar{u} \in U;\ m = 1,2,\ldots, \qquad (2.13)$$

а для зняття фазових обмежень (2.9) – такий штраф [17]:

$$P_3{}^{(m)}(\bar{u}) = \frac{1}{m} \cdot \int_{t_0}^{T} \sum_{i \in I} e^{m \cdot (x_i(t;\bar{u}) - \frac{\pi}{2})(x_i(t;\bar{u}) + \frac{\pi}{2})}\, dt, \quad \bar{u} \in U;\ m = 1,2,\ldots, \qquad (2.14)$$

де $I = \{7, 9, 11\}$.

Таким чином, замість задачі А (2.4)-(2.10) можна розглядати наступну задачу мінімізації вихідного функціонала зі штрафом.

Задача В. Знайти нижню грань функціонала

$$J^{(m)}(\bar{u}) = \int_{t_0}^{T} \sum_{i=1}^{4} \bar{u}_i^2(t)\, dt + P_1{}^{(m)}(\bar{u}) + P_2{}^{(m)}(\bar{u}) + P_3{}^{(m)}(\bar{u}); \qquad (2.15)$$

за умов

$$\dot{X}(t) = f(X(t), \bar{u}(t)), \quad t_0 \le t \le T; \qquad (2.16)$$

$$X(t_0) = X^{(0)}; \qquad (2.17)$$

$$U = \{\bar{u} \equiv \bar{u}(t) \in L_2^{(4)}[t_0;T]:\ 0 \le \bar{u}_i(t) \le 1,\ i = \overline{1,4};\ t_0 \le t \le T\}, \qquad (2.18)$$

де штрафні фунціонали $P_1^{(m)}(\bar{u})$, $P_2^{(m)}(\bar{u})$, $P_3^{(m)}(\bar{u})$ $(m = 1,2,\dots)$ надаються формулами (2.12)-(2.14), а інші позначення зберігають сенс, зазначений при формулюванні задачі А.

Задача В є автономною задачею оптимального керування з вільним правим кінцем, без фазових обмежень, з лінійними диференціальними зв'язками, обмеженнями на керуючий вплив та фіксованим часом.

Таку задачу можна розв'язати будь-яким відомим методом (як прямим, так і непрямим) теорії оптимального керування. Після цього необхідно перевірити виконання умов (2.7)-(2.9); якщо точність їх виконання недостатня, потрібно збільшити в декілька разів коефіцієнті штрафу m і знову розв'язати задачу В (2.15)-(2.18) і т. д. [12]. Розв'язок вихідної задачі оптимального керування рухом дрона-квадрокоптера вважатиметься знайденим, якщо буде досягнена задана точність виконання умов (2.7)-(2.9).

2.3 Застосування методу умовного градієнта до розв'язання задачі В оптимального керування з вільним правим кінцем

Запишемо цільовий функціонал задачі В у вигляді

$$J(\bar{u}) = \int_{t_0}^{T} \left(\sum_{i=1}^{4} \bar{u}_i^2(t) + \frac{1}{m} \sum_{i=1}^{6} e^{-m \cdot x_{2i}(t;\bar{u})} + \frac{1}{m} \sum_{i \in I} e^{m \cdot \left(x_i(t;\bar{u}) - \frac{\pi}{2} \right) \left(x_i(t;\bar{u}) + \frac{\pi}{2} \right)} \right) dt +$$

$$+ m \cdot \sum_{i=1}^{n} \left(x_i(T;\bar{u}) - x_i^{(1)} \right)^2 ; \qquad (2.19)$$

для фіксованого достатньо великого значення m $(m > 0)$ і будемо розглядати задачу його мінімізації за умов (2.16)-(2.18).

Введемо у розгляд допоміжну векторну функцію $\psi(t) = (\psi_1(t), \psi_2(t), \dots, \psi_n(t))$, $t_0 \le t \le T$ і скалярну величину ψ_0 та складемо функцію Гамільтона-Понтрягіна (гамільтоніан) задачі оптимального керування (2.16)-(2.19):

$$H \equiv H(X, \psi_0, \psi, \bar{u}) = \sum_{j=0}^{n} \psi_j(t) \cdot f_j(X(t), \bar{u}(t)), \qquad (2.20)$$

де $\psi_0(t) \equiv \psi_0$; $f_0(X(t), \bar{u}(t))$ – інтегрант функціонала (2.19):

$$f_0\big(X(t),\bar{u}(t)\big)=\sum_{i=1}^{4}\bar{u}_i^2(t)+\frac{1}{m}\sum_{i=1}^{6}e^{-m\cdot x_{2i}(t;\bar{u})}+\frac{1}{m}\sum_{i\in I}e^{m\cdot\left(x_i(t;\bar{u})-\frac{\pi}{2}\right)\left(x_i(t;\bar{u})+\frac{\pi}{2}\right)};$$

$f_i\big(X(t),\bar{u}(t)\big),\ i=\overline{1,n}$ – праві частини системи (2.16), визначені в (2.3).

Складемо систему спряжених диференціальних рівнянь

$$\dot{\psi}_i(t)=-\sum_{j=0}^{n}\psi_j(t)\frac{\partial f_j}{\partial x_i}(X(t),\bar{u}(t)),\quad i=\overline{1,n},\quad t_0\le t\le T,$$

або, поклавши $\psi_0=-1$, детальніше:

$$\begin{cases}\dot{\psi}_1(t)=0,\\ \dot{\psi}_2(t)=-e^{-m\cdot x_2(t;\bar{u})}-\psi_1(t)+\bar{A}_x\psi_2(t),\\ \dot{\psi}_3(t)=0,\\ \dot{\psi}_4(t)=-e^{-m\cdot x_4(t;\bar{u})}-\psi_3(t)+\bar{A}_y\psi_4(t),\\ \dot{\psi}_5(t)=0,\\ \dot{\psi}_6(t)=-e^{-m\cdot x_6(t;\bar{u})}-\psi_5(t)+\bar{A}_z\psi_6(t),\\ \dot{\psi}_7(t)=2x_7(t;\bar{u})e^{m\cdot\left(x_7(t;\bar{u})-\frac{\pi}{2}\right)\left(x_7(t;\bar{u})+\frac{\pi}{2}\right)}+g\psi_4(t),\\ \dot{\psi}_8(t)=-e^{-m\cdot x_8(t;\bar{u})}-\psi_7(t),\\ \dot{\psi}_9(t)=2x_9(t;\bar{u})e^{m\cdot\left(x_9(t;\bar{u})-\frac{\pi}{2}\right)\left(x_9(t;\bar{u})+\frac{\pi}{2}\right)}-g\psi_2(t),\\ \dot{\psi}_{10}(t)=-e^{-m\cdot x_{10}(t;\bar{u})}-\psi_9(t),\\ \dot{\psi}_{11}(t)=2x_{11}(t;\bar{u})e^{m\cdot\left(x_{11}(t;\bar{u})-\frac{\pi}{2}\right)\left(x_{11}(t;\bar{u})+\frac{\pi}{2}\right)},\\ \dot{\psi}_{12}(t)=-e^{-m\cdot x_{12}(t;\bar{u})}-\psi_{11}(t),\end{cases}\qquad t_0\le t\le T.\quad(2.21)$$

Систему рівнянь (2.21) доповнимо умовою трансверсальності на правому кінці

$$\psi_j(T)=-\frac{\partial\Phi}{\partial x_j(T)}(X(T)),\quad j=\overline{1,n},$$

де $\Phi(X(T))=m\cdot\sum_{i=1}^{n}(x_i(T;\bar{u})-x_i^{(1)})^2$ – термінальна частина функціонала (2.19). Отже,

$$\psi_j(T)=-2m\cdot(x_j(T;\bar{u})-x_j^{(1)}),\quad j=\overline{1,n}.\qquad(2.22)$$

Можна показати, на підставі відомої теореми про диференційованість функціонала задачі оптимального керування з вільним правим кінцем, що функціонал (2.19) за умов (2.16)-(2.18) неперервний і диференційований за $\bar{u}=$

$\bar{u}(t)$ в нормі $L_2^{(4)}[t_0;T]$, причому його градієнт $J'(\bar{u})$ у точці $\bar{u}=\bar{u}(t)\in U$ може бути поданий у вигляді

$$J'(\bar{u})=-\frac{\partial H}{\partial \bar{u}}\big(X(t),\psi(t),\bar{u}(t)\big),\quad t_0\le t\le T,$$

або у координатній формі:

$$J_i'(\bar{u})=-\frac{\partial H}{\partial \bar{u}_i}\big(X(t),\psi(t),\bar{u}(t)\big),\ \ i=\overline{1,4};\ \ t_0\le t\le T,$$

тобто, з урахуванням формул (2.20) і (2.3),

$$J'(\bar{u})=\begin{bmatrix}J_1'(\bar{u})\\J_2'(\bar{u})\\J_3'(\bar{u})\\J_4'(\bar{u})\end{bmatrix}=\begin{bmatrix}2\bar{u}_1-K\psi_6(t)+\bar{J}_y\psi_{10}(t)+\bar{J}_z\psi_{12}(t)\\2\bar{u}_2-K\psi_6(t)+\bar{J}_x\psi_8(t)-\bar{J}_z\psi_{12}(t)\\2\bar{u}_3-K\psi_6(t)-\bar{J}_y\psi_{10}(t)+\bar{J}_z\psi_{12}(t)\\2\bar{u}_4-K\psi_6(t)-\bar{J}_x\psi_8(t)-\bar{J}_z\psi_{12}(t)\end{bmatrix},\ \ t_0\le t\le T,\quad (2.23)$$

де $X(t)\equiv X(t;\bar{u}(t))$ – розв’язок задачі Коші (2.16), (2.17) (або, що те ж саме, системи диференціальних рівнянь (2.2) з початковими умовами (2.6)) при $\bar{u}=\bar{u}(t)\in U$, а вектор-функція $\psi(t)\equiv\psi(t;\bar{u}(t))$ є розв’язком задачі Коші (2.21), (2.22) на $[t_0;T]$. Таким чином, для обчислення градієнта $J'(\bar{u})$ функціонала (2.19) за умов (2.16)-(2.18) в будь-якій точці $\bar{u}=\bar{u}(t)\in U$ потрібно послідовно розв’язати дві задачі Коші: (2.2), (2.6) та (2.21), (2.22), і знайдені при цьому компоненти $\psi(t)$, $t_0\le t\le T$ підставити в (2.23). Під час розв’язування зазначених задач Коші можна використовувати відомі чисельні методи, наприклад метод Рунге-Кутта четвертого порядку точності.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ КЕРОВАНОГО РУХУ ДРОНА

3.1 Програмна реалізація

Програмна реалізація включає в себе реалізацію методу штрафних функціоналів із пункту 2.2 та методу умовного градієнта пункту 2.3 із виникаючими задачами Коші. Для реалізації було використано мову програмування Java в середовищі програмування IntelliJ IDEA Community Edition 2020.3.3. Детальна структура та код програми наведені в додатку В.

3.2 Алгоритм роботи програми

Опишемо алгоритм роботи програми у вигляді псевдокоду із застосуванням формальних роздумів щодо роботи програми. Перейдемо безпосередньо до опису тіла програми, що містить основну частину обчислень в класі Main в методі main, що є точкою входу програми:

1. Будуємо розбиття проміжку часу $[t_0,\ T]$, розділивши його на N частин;
2. Задаємо початкові значення крокового множника метода штрафних функціоналів m^0 та метода умовного градієнта $\alpha^{(0)}$, задаємо початкові значення вектора $u^{(0)}$;
3. Виконуємо тіло циклу методу штрафних функціоналів:
4. Отримаємо розв’язок задачі Коші для функцій вектора $X^{(k)}$;
5. Отримаємо значення функціонала за значенням керування $u^{(k)}$ та $X^{(k)}$;
6. Виконуємо тіло циклу методу умовного градієнту:
   i. Отримаємо значення функціонала за значенням керування $u^{(k)}$ та $X^{(k)}$;
   ii. Отримаємо розв’язок задачі Коші для функцій вектора $\psi^{(k)}$;
   iii. Отримаємо значення градієнта за значень $\psi^{(k)}$;
   iv. Обчислимо допоміжний вектор значень керування $\bar{u}^{(k)}$ в залежності від знаку градієнта $J'\left(u^{(k)}\right)$;
   v. За ітераційною формулою метода умовного градієнта отримаємо нові значення керування $u^{(k+1)}$;

vi. Отримаємо розв'язок задачі Коші для функцій вектора $X^{(k+1)}$;

vii. Отримаємо значення функціонала за значенням керування $u^{(k+1)}$ та $X^{(k+1)}$;

viii. За допомогою метода бінарного пошуку обчислимо нове значення ітераційного множника $\alpha^{(k+1)}$, обчислюючи нове значення аж допоки не буде виконана умова збіжності градієнтного метода;

ix. Копіюємо значення $u^{(k+1)}$ до змінних $u^{(k)}$, а значення $x^{(k+1)}$ до змінних $x^{(k)}$;

x. Додаємо одиницю до лічильника пройдених ітерацій й перевіряємо умову виходу з градієнтного метода або за максимальною кількістю ітерацій end_M. За виконання умов, виходимо з тіла циклу метода умовного градієнта за невиконання умов, переходимо до пункту 6;

7. Отримаємо значення функціонала за значенням керування $u^{(k^*)}$ та $X^{(k^*)}$;
8. Додаємо одиницю до лічильника пройдених ітерацій, переходимо до нового значення крокового множника m, додаючи деяку константу, перевіряємо умову виходу з градієнтного метода або за максимальною кількістю ітерацій end_M. За виконання умов, виходимо з тіла циклу метода умовного градієнта, за невиконання умов, переходимо до пункту 3;
9. Формуємо звітність про результати роботи програми, створюємо файли, що містять сіткові значення функцій керування u^*, вектора X^* та значення спряжених функцій ψ^*.

Отже, за описаним алгоритмом відбуваєтеся робота програми, деталі програмної реалізації модулів програми можна побачити в додатку А.

Зазначимо, що програмна реалізація являє собою щось схоже на інженерний калькулятор й не призначена для використання недосвідченим користувачем. Взагалі програма використовується для дослідження поведінки компонентів поставленої задачі В.

3.2 Чисельний експеримент

3.2.1 Числові дані для задачі В

Чисельний експеримент буде проведено відповідно до значень відомих додатних сталих, які були визначені в дослідженні [1], беручи до уваги значення, що були використані в дослідженнях [2], [15], [16], [17].

Початкові параметри математичної моделі(константи) наведено в додатку Б.

Дані параметри математичної моделі, для потреб поставленої задачі В будуть модифіковані програмно за формулами (1.25).

3.2.2 Опис чисельних експериментів

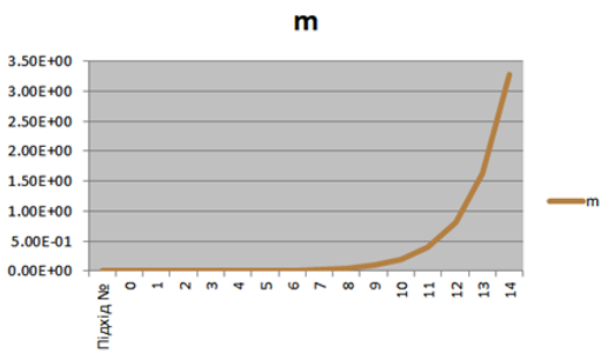
Проведено два числових експерименти. В першому досліджено залежність між такими компонентами, як значення цільового функціонала $J^{(m)}(\bar{u})$, значення коефіцієнту штрафу m, кількість необхідних ітерацій методу умовного градієнта k_g та норма сіткового градієнта $J'(\bar{u})$. А в другому дослідимо поведінку метода умовного градієнта за фіксованого оптимального m, що було обрано дослідним шляхом. Зазначимо, що експерименти були проведені для зазначених параметрів математичної задачі, що подані в таблиці 1Б та значень класу Constants програмної реалізації, що описаний в додатку В.

Перший експеримент

Для зручності, подамо результати розрахунків графічно, що видно на рис. 3.2, рис. 3.3, рис. 3.4, рис. 3.5. Зазначимо, що для значень функціонала та норми градієнта в табличці на рис. 3.1 було спеціально обрано інші значення в червоних клітинках, за для передбачення спотворення графіків.

Підхід №	m	k_g	J	Norm_grad
0	1.00E-04	9	179791	12245.9485
1	2.00E-04	3	89967.6	12248.1158
2	4.00E-04	3	45012.9	12252.4512
3	8.00E-04	3	22536	12261.095
4	0.0016	5	11298.4	12278.3373
5	0.0032	4	5681.54	12312.5153
6	0.0064	4	2877.17	12379.6685
7	0.0128	6	1484.45	12509.8842
8	0.0256	8	813.924	12756.2444
9	0.0512	10	578.197	13209.6342
10	0.1024	14	1286.34	14202.5667
11	0.2048	13	42702.2	19765.866
12	0.4096	6	1.00E+05	73408.299
13	0.8192	8	2.00E+05	120000
14	1.6384	1	3.00E+05	240000
15	3.2768	1	4.00E+05	360000

Рисунок 3.1 – Результати розрахунків за першим експериментом в табличці



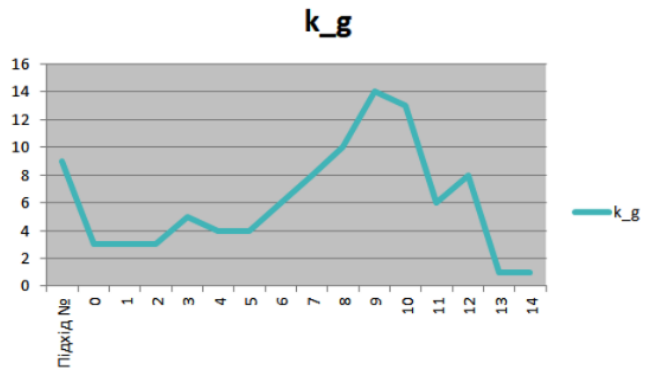
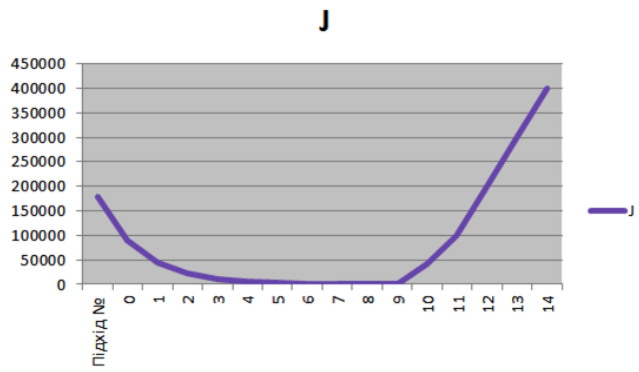


Рисунок 3.2, 3.3 – Значення m та k_g відповідно за результатами першого експерименту



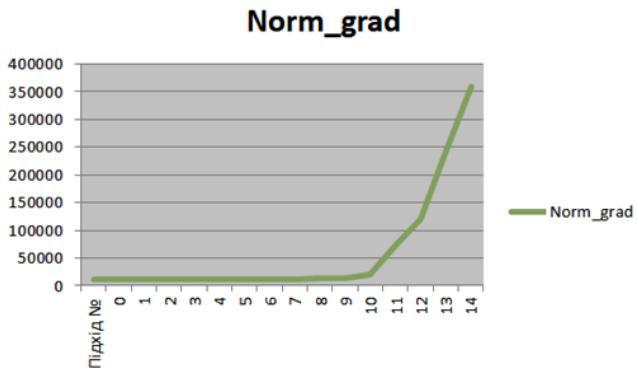


Рисунок 3.4, 3.5 – Значення J та $Norm\_grad$ відповідно за результатами першого експерименту

Вочевидь за рисунком 3.4 можна судити, що цільовий функціонал має деякий мінімум, за певного значення m. Шляхом проведення низки досліджень, було встановлено, що це значення $m = 0.25$.

Другий експеримент:

```
0.m = 0.25, k_g = 18, J = 50.70551841254992 Norma_Grad = 25181.21715859759
--------------------------------------------------------------------------------

Process finished with exit code 0
```

Рисунок 3.6 – Результати другого експерименту за фіксованого $m = 0.25$

В додатку Б продемонстровано графіки функцій $x_1, \dots, x_{12}$, $\psi_1, \dots, \psi_{12}$ на рис 1.1Б – рис 1.7Б, а на рис 3.7 зображено графіки функцій керування u_1, u_2, u_3, u_4

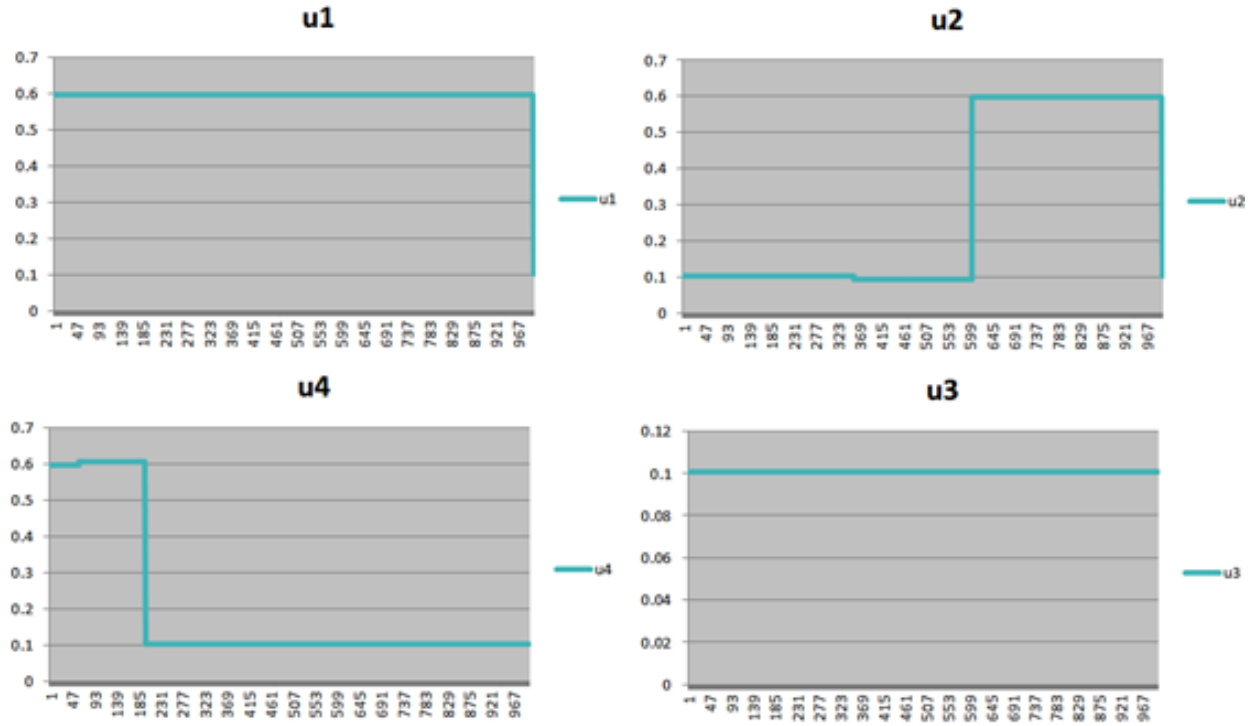


Рисунок 3.7 – Результати другого експерименту для $u_1,\ u_2,\ u_3,\ u_4$

3.3 Аналіз результатів

Результати експериментів демонструють певну нестабільну поведінку обчислювальних алгоритмів, що можна пов’язати як із властивостями розглянутої математичної моделі, в тому числі системи спряжених диференціальних рівнянь, яка в загальному випаду є нестійкою (тобто некоректною за Адамаром), так і особливостями сіткової реалізації побудованих числових схем методу штрафних функціоналів, які є досіть чутливими до процедур узгодження параметрів алгоритму. З огляду на це, необхідно детально проаналізувати отримані результати та поміркувати про можливі причини подібної поведінки моделі, запропонувати шляхи для подальших досліджень.

3.3.1 Аналіз результатів дослідження

Звернемось для початку до рис. 3.2. Одразу стає зрозумілим, що значення цільового функціонала із врахуванням штрафів за невиконання фазових обмежень, не може бути більше ніж $J(u^*) = 4$, враховуючи межі в яких

визначено керування та формулу цільового функціонала (2.10). Що також підтверджує велике значення норми градієнту. Виходячи з цього можемо дійти першого висновку, що поставлена задача оптимального керування рухом дрона-квадрокоптера не була розв'язана оптимальним чином, й що вочевидь результати дослідження буде важко або навіть неможливо проінтерпретувати. Звернувшись до програмної реалізації, можемо визначити, які саме компоненти цільового функціоналу приносять йому великих значень.

```
J(fo) = 0.15983999999999618
J(f0_P2) = 23.916872174728773
J(f0_P3) = 25.87880623782115
J(f0_P1) = 0.75
0.m = 0.25, k_g = 18, J = 50.70551841254992 Norma_Grad = 25181.21715859759
--------------------------------------------------------------------------------
```

Рисунок 3.8 – Детальні результати другого експерименту

За рис. (3.8) видно, що значення цільового функціонала дійсно менше за очікуване $J(u^*) = 4$, значення штрафу $P_1^{(0.25)}$ (2.12) за зняття умови (2.7) є й складає досить невелике значення, але це вже говорить, про невиконання умов задачі (2.2), (2.6). Наступні значення штрафу $P_2^{(0.25)}$(2.13) та $P_3^{(0.25)}$ (2.14) майже однакові й складають істотну частину цільового функціоналу, через що можемо зробити висновок, що є грубе порушення умов (2.8) та (2.9). Після цього, говорити про інтерпретацію отриманих результатів не є доцільним, адже ті фізичні обмеження, виходячи з яких були сформульовані обмеження ЗОК, не виконуються.

Проаналізуємо поведінку метода штрафних функціоналів та метода умовного градієнту за рис. 3.2 – 3.5. За рис. 3.4 та висновками розділу 1.4 про опуклість цільового функціонала, можна справді стверджувати, що цільовий функціонал в постановці задачі В має мінімум. Саме з цих роздумів було обрано фіксоване значення $m = 0.25$, й справді йому відповідає найменше з отриманих під час дослідження, значення функціоналу, як видно за рисунком 3.7. Також зазначимо очевидну залежність між ростом m та ростом кількості необхідних підходів метода умовного градієнта k_g, що радикально впливає на

значення цільового функціоналу та відповідно істотно збільшує норму градієнта.

Проаналізувавши значення за рис 1.1Б – 1.6Б, можемо зазначити, що є невиконання фазових обмежень на компоненти задачі В, але не зважаючи на це, є виконання граничних умов для $x_2, x_4, x_6, x_7, x_8, x_9, x_{10}, x_{11}, x_{12}$, що свідчить про правильність реалізації метода Рунге-Кутта для задачі Коші (2.2), (2.6). Що до умов для x_1, x_3, x_5, можемо сказати, що граничні умови не виконуються, що проаналізуємо пізніше. За рисунком 1.7Б видно, що деякі з $\psi_1, \dots, \psi_{12}$ не сходяться до нуля на правому кінці часового проміжку $[t_0,\ T]$, це значення ψ_1, ψ_3, ψ_5, які є спряженими до x_1, x_3, x_5, що наводить на думки про неповному складеної математичної моделі. Ще одне зауваження стосується значень u_1, u_2, u_3, u_4 за рис. 3.7, видно, що керування в парах u_1, u_3 та u_2, u_4 має деяку схожість в поведінці, що можна проінтерпретувати як відповідність будові дрона-квадрокоптера, де відповідні зазначені пари спряжені й обертаються в однаковому напрямку, що є добрим знаком.

Зазначимо, що аналіз точності та впливу різноманітних констант на результати дослідження є не доцільним, адже розв’язок поставленої задачі не є оптимальним й потребує доопрацювання.

3.3.2 Можливі причини незадовільної поведінки розв’язку та шляхи до їх подолання

Зазначимо ще раз, що немає сумнівів у правильності реалізації методів Рунге-Кутта для розв’язання виникаючих задач Коші. Було також проведено дослід, в якому значення керування задавалось на гвинти у вигляді взагалі-то кажучи сіткової функції, що нагадує параболу, тобто без використання методів штрафних функціоналів та умовного градієнта. При цьому встановлено, що система не може адекватно реагувати на надане керування, що знову ж таки відводить підозри про неправильність програмної або алгоритмічної реалізації на користь помилки в виведеній математичної моделі.

Враховуючи огляд літературних джерел із дослідженнями, що ставили собі на меті схожу задачу керування, можемо сказати, що підозри про неадекватність фізичної моделі скоріш за все можна відкинути. Найбільше питань викликають чисельні спрощення математичної моделі, які були проведені в роботі [1], скоріш за все саме в цьому полягає проблема й пошальше виведення математичної моделі та ЗОК, що не має адекватного зв'язку керування та степенів свободи дрона-квадрокомпера, що описують його рух.

Подальше дослідження за даною проблематикою має передбачати досліжедння математичної моделі керування рухом дрона-квадрокоптера на стійкість із залученням відповідних регуляризуючих алгоритмів.

ВИСНОВКИ

Числовий аналіз керованого руху дрона-квадрокоптера має велике прикладне значення та є актуальним як для сфери виготовлення БПЛА так і для сфер використання БПЛА. Ця наукова робота демонструє підхід до дослідження задачі керованого руху БПЛА, який відкриває багато супутніх питань, що є доречними та важливими для розвитку теорії керованого руху БПЛА зокрема дронів-квадрокоптерів.

Отже:

1. Було опрацьовано теоретичний матеріал що розглядає БПЛА як фізичні тіла в просторі;
2. Було досліджено існуючі математичні моделі руху БПЛА, обрано конкретне джерело, що містить цінні роздуми про побудову математичної моделі керованого руху дрона-квадрокоптера, на основі якого було виведено математичну модель керованого руху дрона-квадрокоптера.
3. Була сформульована ЗОК в канонічній формі відповідно до виведеної математичної моделі.
4. Було опрацьовано теоретичний матеріал теорії керування та теорії оптимізації, що дозволив сформулювати ЗОК в канонічній формі як крайову задачу принципу максимуму Понтрягіна із врахуванням обмежень на керування та фазовими обмеженнями.
5. Розроблено алгоритм та програмне забезпечення, що реалізує метод штрафних функціоналів, метод умовного градієнта та вирішує супутні задачі Коші за допомогою методу Рунге-Кутта четвертого порядку точності.
6. Проведено чисельний експеримент та проаналізовано результати. Зроблено відповідні висновки, особливо стосовно математичної моделі керованого руху дрона-квадрокоптера.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нагайко, Д. Ю. Система керування пошуково-рятувальним безпілотним літальним апаратом : магістерська дис. : 126 Системи керування. – Київ, 2018. – 158 с.
2. Luukkonen T. Modelling and Control of Quadcopter, Espoo., 2011. – 26p.
3. Jolliffe I. T. Principal Component Analysis, 2002. – 271p.
4. Tanner O. Modeling, Identification and Control of Autonomous Helicopters // ETHZ, 2003. – 191p.
5. Pounds P. Towards dynamically-favourable quad-rotor aerial robots // Proc. Australasian Conference on Robotics and Automation (ACRA'04), 2004. – 10p.
6. Fay G. Derivation of the aerodynamic forces for the mesicopter simulation, 2001. – 8 p.
7. Мартынов А. К. Экспериментальная аэродинамика, 1950. – 479 с.
8. Beard R. W. Quadrotor Dynamics and Control // Brigham Young University., 2008. – p.
9. Vasiliev O. V. Optimization methods // Atlanta, USA: Worls Federation Publishers Company, Inc., 1996. – 276 p.
10. Bryson A.E., Ho Yu-Chi. Applied Optimal Control: Optimization, Estimation, and Control. Taylor & Francis Group, 2017. 481 p.
11. Кісельова О.М., Гарт Л.Л. Математичні методи системного аналізу: навч. посіб. – Дніпро: РВВ ДНУ, 2019. – 124 с.
12. Киселева Е.М., Коряшкина Л.С. Методы решения задач оптимального управления: учебн. пособие. – Д.: ДНУ, 1997. – 120 с.
13. Федоренко Р.П. Приближенное решение задач оптимального управления. – М.: Наука, 1978. – 488 с.
14. А.М. Красовський, О.О. Суслова. Спрощена математична модель керованого руху квадрокоптера, 2016. – 5 с.
15. Safaei, A., Mahyuddin, M.N. Modeling and Adaptive Control Design for a Quadrotor // 9th International Conference on Robotic, Vision, Signal Processing

and Power Applications. Lecture Notes in Electrical Engineering, vol 398. Springer, Singapore, 2017. – 443–452 p.

16. R.A. García, F.R. Rubio, M.G. Ortega, Robust PID Control of the Quadrotor Helicopter // IFAC Proceedings Volumes, Volume 45, Issue 3, 2012, – 5 p.
17. Zuo, Zongyu. Trajectory tracking control design with command-filtered compensation for a quadrotor // Iet Control Theory and Applications 4, 2010. – 2343-2355 p.

ДОДАТОК А

Математична модель керованого руху дрона-квадракоптера

1 Математична модель руху БПЛА

1.1 Система координат

Як буде розглянуто пізніше, дрон має шість ступенів свободи і його рух описується відповідно шістьма диференціальними рівняннями другого порядку (рівняння Ейлера). Розв'язок цих рівнянь в загальному випадку дозволив би визначити характер просторового руху дрона в будь-який момент часу. Зрозуміло, що безпосереднє розв'язання цих рівнянь представляє складнощі. Якщо ж за вихідний режим польоту прийняти прямолінійний сталій політ без ковзання і вважати відхилення параметрів руху вихідних значень незначним, то, завдяки симетричній будові дрона-квадрокоптера, систему з шести рівнянь руху можна розділити на дві незалежні системи рівнянь, що описують рух дрона в площині симетрії (поздовжній рух) і в двох інших площинах (бічний рух).

Аби кількісно описати положення та рух дрона у просторі, можна використовувати різні системи координат, такі як інерціальні, земні та рухомі. Вибір певної системи координат залежить від задачі, яку потрібно вирішити.

Нормальна земна система координат (система відліку) – $O_з X_з Y_з Z_з$. Початок $O_з$ цієї системи координат лежить на поверхні землі, а відповідно осі направлені наступним чином:

- $O_з X_з$ та $O_з Y_з$ – лежать в горизонтальній місцевій площині, утворюючи тим самим праву прямокутну декартову систему координат;
- $O_з Z_з$ – спрямована перпендикулярно до площини $O_з X_з Y_з$, в протилежному напрямку сили тяжіння.

Пов'язана з дроном система координат (рухома система координат) $O_b X_b Y_b Z_b$. Ця система координат збігається з осями конструкції дрона, центр O_b лежить в центрі маси дрона, а осі напрямлені наступним чином:

- $O_b X_b$ – лежить вздовж площини симетрії дрону і спрямована від хвостової частини дрона до носової частини дрона;
- $O_b Y_b$ – лежить перпендикулярно осі $O_b X_b$ в площині симетрії дрона;

- O_bZ_b – спрямована перпендикулярно до площини $O_bX_bY_b$, в напрямку дії підйомної сили дрона.

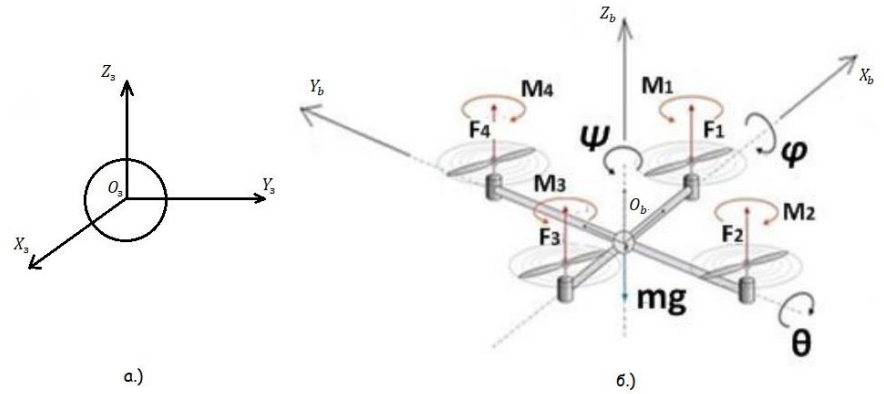


Рисунок 1.1 – Системи координат: а.) система відліку положення дрона, б.) система координат дрона

Аби задати відхилення дрона у просторі в момент часу t $(t_0 \leq t \leq T)$, використовують так звані кути Ейлера, які позначають $\varphi \equiv \varphi(t)$, $\theta \equiv \theta(t)$, $\psi \equiv \psi(t)$. Дамо опис кожного з цих кутів:

- Кут φ (кут крену) – обчислюється як кут повороту площини $O_bY_bZ_b$ відносно нерухомої площини $O_зY_зZ_з$, від $-90°$ проти годинникової стрілки до $90°$ за годинниковою стрілкою.

- Кут θ (кут тангажу) – обчислюється як кут повороту площини $O_bX_bZ_b$ відносно нерухомої площини $O_зX_зZ_з$, від $-90°$ проти годинникової стрілки до $90°$ за годинниковою стрілкою.

- Кут ψ (кут рискання) – обчислюється як кут повороту площини $O_bX_bY_b$ відносно нерухомої площини $O_зX_зY_з$, від $-90°$ проти годинникової стрілки до $90°$ за годинниковою стрілкою.

Поступальний рух дрона як тіла в просторі представлятимемо як рух його центру мас відносно нерухомої системи координат. Тоді положення дрона у просторі в момент часу t $(t_0 \leq t \leq T)$ можна повністю описати за такими параметрами, як широта $\Phi \equiv \Phi(t)$, довгота $L \equiv L(t)$ і висота $H \equiv H(t)$.

Також дрон-квадрокоптер відносно нерухомої системи координат здійснює обертальний рух за віссю обертання, що проходить через центр мас дрона й, взагалі-то кажучи, перпендикулярно до осі симетрії дрона.

Отже задані шість параметрів, що характеризують положення дрона у просторі в кожен момент часу t ($t_0 \leq t \leq T$): кут крену φ, кут тангажу θ, кут рискання ψ, широта Ф, довгота L, висота H. Можемо перейти до аналогічної групи параметрів: кут крену φ, кут тангажу θ, кут рискання ψ, пройдена відстань L, бічне відхилення δ, висота H.

Аби перейти від однієї системи координат до іншої, використаємо обертальні матриці. Процес обертання пояснюється за допомогою моделі Ньютона-Ейлера, яка включає в себе матриці $R_x \equiv R_x(t)$, $R_y \equiv R_y(t)$, $R_z \equiv R_z(t)$, $t_0 \leq t \leq T$, що використовуються для визначення обертання дрону в фіксованій системі координат, що рухається в рухомій системі координат. Детальніше:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_\varphi & S_\varphi \\ 0 & -S_\varphi & C_\varphi \end{bmatrix}; \quad R_y = \begin{bmatrix} C_\theta & 0 & -S_\theta \\ 0 & 1 & 0 \\ S_\theta & 0 & -C_\theta \end{bmatrix}; \quad R_z = \begin{bmatrix} C_\psi & S_\psi & 0 \\ -S_\psi & C_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

де, заради зручності, використані позначення

$$S_\alpha = \sin\alpha, \quad C_\alpha = \cos\,\alpha, \quad T_\alpha = \operatorname{tg}\alpha.$$

Нам знадобиться узагальнена матриця обертання, що визначається за формулами

$$R = R_x * R_y * R_z;$$

$$R = \begin{bmatrix} S_\theta S_\varphi S_\psi + C_\theta S_\psi & C_\varphi S_\psi & C_\theta S_\varphi S_\psi - S_\theta C_\psi \\ S_\theta S_\varphi C_\psi - C_\theta S_\psi & C_\theta C_\varphi & C_\theta S_\varphi C_\psi - S_\theta S_\psi \\ S_\theta C_\varphi & -S_\varphi & C_\theta C_\varphi \end{bmatrix}, \quad t_0 \leq t \leq T.$$

Тоді за допомогою $R\,(R_x, R_y, R_z)$ можливо буде визначити поведінку в дрона в просторі в будь-який момент часу на $[t_0; T]$.

Матрицю перетворення кутових швидкостей дрона з нерухомої системи координат в рухому позначимо як

$$W_\eta = \begin{bmatrix} 1 & S_\varphi T_\theta & C_\varphi T_\theta \\ 0 & C_\varphi & -S_\varphi \\ 0 & \frac{S_\psi}{C_\theta} & \frac{C_\psi}{C_\theta} \end{bmatrix}, \quad t_0 \le t \le T. \tag{1.1}$$

Вочевидь, матриця $W_\eta \equiv W_\eta(t)$ оборотна при $\theta \neq \frac{(2k-1)}{2}\varphi,\ k \in \mathbb{Z}$.

1.2 Кінематичні рівняння

При моделюванні руху дрону були враховані такі припущення:

- тіло дрона є абсолютно жорстким;
- осі рухомої системи координат збігаються з основною віссю моменту інерції, тому можна вважати $[I_{xy}, I_{yz}, I_{zx}] \approx 0$;
- обертання відносно осі $O_b Z_b$ є симетричним, тому $I_x = I_y$;
- гіроскопічні сили ротора є незначними;
- інерція двигуна є малою, отже відставання в двигунах є незначним.

Динамічна система представлена за допомогою чотирьох векторів, відношення між якими описують закони польоту дрона-квадрокоптера:

$$\bar{X} = \begin{bmatrix} \bar{V} \\ \bar{W} \\ \bar{E} \\ \bar{P} \end{bmatrix}, \quad t_0 \le t \le T. \tag{1.2}$$

Тут $\bar{P}$ – вектор положення дрона відносно нерухомої системи координат:

$$\bar{P} = \begin{bmatrix} P_x \\ P_y \\ P_z \end{bmatrix} = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix}, \quad t_0 \le t \le T, \tag{1.3}$$

де індекси x, y, z визначають проєкції вектора положення в рухомій системі координат.

Вектор поступальної швидкості $\bar{V}$ має вигляд

$$\bar{V} = \begin{bmatrix} u(t) \\ v(t) \\ w(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{z}(t) \end{bmatrix}, \quad t_0 \le t \le T, \tag{1.4}$$

де $u \equiv u(t)$, $v \equiv v(t)$, $w \equiv w(t)$ – проєкції поступальної швидкості в рухомій системі координат.

Вектор кутової швидкості дрона $\bar{W}$ в залежності від фіксованої системи координат має вигляд

$$\bar{W} = \begin{bmatrix} p(t) \\ q(t) \\ r(t) \end{bmatrix}, \quad t_0 \leq t \leq T, \tag{1.5}$$

де $p \equiv p(t)$, $q \equiv q(t)$, $r \equiv r(t)$ – проєкції обертальної швидкості дрона в рухомій системі координат. Взагалі-то кажучи, за фізичним змістом, $p \geq 0$, $q \geq 0$, $r \geq 0$ ($t_0 \leq t \leq T$). Наступні обмеження на фізичні величини будуть враховані на етапі формулювання фазових обмежень у математичній моделі.

Вектор орієнтації дрона $\bar{E}$ в залежності від фіксованої системи координат має вигляд

$$\bar{E} = \begin{bmatrix} \varphi(t) \\ \theta(t) \\ \psi(t) \end{bmatrix}, \quad t_0 \leq t \leq T. \tag{1.6}$$

1.3 Динамічні рівняння

Динаміку дрона-квадрокоптера можна описати за допомогою рівнянь Ньютона-Ейлера:

$$\bar{F} = m\frac{d\bar{V}}{dt}, \quad t_0 \leq t \leq T, \tag{1.7}$$

де m (в кг) – загальна маса дрона; $\bar{F} \equiv \bar{F}(t)$ – вектор сумарної сили, прикладеної до дрона.

При переході з нерухомої системи координат в рухому, перепишемо закон у вигляді

$$\bar{F} = m\left(\frac{d_b\bar{V}}{dt} + \bar{W} + \bar{V}\right), \quad t_0 \leq t \leq T, \tag{1.8}$$

де $\frac{d_b\bar{V}}{dt}$ – лінійне прискорення дрона відносно рухомої системи координат.

При відсутності впливу вітру, діючими силами на дрон є сила тяжіння $\bar{G}$ і сили аеродинамічної тяги $\bar{T}$:

$$\bar{F} = \bar{G} + \bar{T}, \quad t_0 \leq t \leq T. \tag{1.9}$$

Оскільки дія сил мається на увазі в нерухомій системі координат, то вираз (1.9) запишемо відносно рухомої системи координат:

$$\bar{F} = \bar{G}_b + \bar{T}_b, \quad t_0 \le t \le T; \tag{1.10}$$

$$\bar{G}_b = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix}, \tag{1.11}$$

де g (в м/с^2) – прискорення вільного падіння.

Зазначимо, що підйомна сила залежить від факторів середовища та форми й конструкції гвинта. Взагалі-то кажучи, підйомна сила гвинта пропорційна квадрату кутової швидкості гвинта і таке відношення можна виразити формулою [2]

$$F_L = k_L * \omega^2, \quad t_0 \le t \le T, \tag{1.12}$$

де $F_L \equiv F_L(t)$ – підйомна сила; $\omega \equiv \omega(t)$ – кутова швидкість; k_L – коефіцієнт, що враховує такі фактори, як профіль гвинта, кут атаки та густину повітря.

Тоді за (1.12) можемо задати силу тяги $\bar{T}_b$:

$$\bar{T}_b = \sum_{i=1}^{4} \tau_i \approx k_L * \sum_{i=1}^{4} \Omega_i^2, \quad t_0 \le t \le T, \tag{1.13}$$

де $\tau_i \equiv \tau_i(t)$ – тяга i-го ротора, $\Omega_i \equiv \Omega_i(t)$ – швидкість обертання i-го ротора.

Відповідно, сили $\bar{G}$ та $\bar{T}$ запишемо в рухомій системі координат за допомогою перетворення за матрицею обертання:

$$\bar{G}_b = mg \begin{bmatrix} S_\theta \\ -S_\varphi C_\theta \\ -C_\varphi C_\theta \end{bmatrix}, \quad t_0 \le t \le T, \tag{1.14}$$

$$\bar{T}_b = \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}, \quad t_0 \le t \le T. \tag{1.15}$$

Підставимо (1.5), (1.6), (1.14), (1.15) в (1.8):

$$\bar{G}_b + \bar{T}_b = m \left(\frac{d_b \bar{V}}{dt} + \bar{W} + \bar{V} \right), \quad t_0 \le t \le T; \tag{1.16}$$

звідки маємо:

$$\frac{d_b\bar{V}}{dt} = \frac{1}{m}\left(mg\begin{bmatrix} S_\theta \\ -S_\varphi C_\theta \\ -C_\varphi C_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}\right) - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times \begin{bmatrix} u \\ v \\ w \end{bmatrix}; \qquad (1.17)$$

і після скорочення:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} rv - qw \\ pw - ru \\ qu - pv \end{bmatrix} + \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + \begin{bmatrix} gS_\theta \\ -gS_\varphi C_\theta \\ -gC_\varphi C_\theta \end{bmatrix}, \quad t_0 \le t \le T. \qquad (1.18)$$

Враховуючи, що зв'язок між лінійними швидкостями в інерційній та зв'язаній системах відліку задається співвідношенням

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = R\bar{V}_b, \quad t_0 \le t \le T; \qquad (1.19)$$

продиференціюємо (1.19), нехтуючи $\dot{R}$, і отримаємо:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = R\overline{\dot{V}_b}, \quad t_0 \le t \le T. \qquad (1.20)$$

Оскільки інерційна система відліку передбачається нерухомою, то можемо підставити (1.18) в (1.19), нехтуючи першим доданком в правій частині (1.18):

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix}\begin{bmatrix} C_\psi S_\theta C_\varphi + S_\psi S_\varphi \\ S_\psi S_\theta C_\varphi + C_\psi S_\varphi \\ C_\varphi C_\theta \end{bmatrix} - g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad t_0 \le t \le T. \qquad (1.21)$$

Доповнимо (1.21) силою аеродинамічного опору за джерелом [11]:

$$F_\alpha = c_d \frac{\rho V^2}{2} S, \quad t_0 \le t \le T, \qquad (1.22)$$

де c_d – коефіцієнт аеродинамічної сили; ρ (в кг/м^3) – щільність повітря; $V \equiv V(t)$ (в м/с) – швидкість набігаючого потоку повітря; S (в м^2) – площа поверхні апарату, на яку діє потік, що набігає.

Для зручності, розпишемо F_α у вигляді $A_k \cdot \dot{k}$, де $A_k = c_d \frac{\rho \dot{k}}{2} S$ – так звані коефіцієнти (в кг/с) проєкції сили аеродинамічного опору:

$$A_x = c_d \frac{\rho \dot{x}}{2} S, \qquad A_y = c_d \frac{\rho \dot{y}}{2} S, \qquad A_z = c_d \frac{\rho \dot{z}}{2} S, \quad t_0 \le t \le T. \qquad (1.23)$$

Тоді (1.21) набуває вигляду

$$\begin{bmatrix}\ddot{x}\\ \ddot{y}\\ \ddot{z}\end{bmatrix} = \frac{1}{m}\begin{bmatrix}0\\0\\T\end{bmatrix}\begin{bmatrix}C_\psi S_\theta C_\varphi + S_\psi S_\varphi\\ S_\psi S_\theta C_\varphi + C_\psi S_\varphi\\ C_\varphi C_\theta\end{bmatrix} - g\begin{bmatrix}0\\0\\1\end{bmatrix} - \frac{1}{m}\begin{bmatrix}A_x & 0 & 0\\ 0 & A_y & 0\\ 0 & 0 & A_z\end{bmatrix}\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{z}\end{bmatrix}. \quad (1.24)$$

Тепер запишемо другий закон Ньютона для обертального руху в нерухомій системі координат:

$$\frac{d\bar{L}}{dt} = \bar{M}, \quad t_0 \le t \le T, \quad (1.25)$$

де $\bar{L} \equiv \bar{L}(t)$ (в кг $\cdot$ м$^2 \cdot$ с$^{-1}$) – вектор кутового моменту; $\bar{M} \equiv \bar{M}(t)$ (в $H \cdot$ м) – вектор моменту сили обертання.

За для переходу до рухомої системи координат, запишемо (1.25) як:

$$\frac{d\bar{L}}{dt} = \frac{d_b\bar{L}}{dt} + \bar{W} \times \bar{L} = \bar{M}, \quad t_0 \le t \le T, \quad (1.26)$$

В інерційній системі $\bar{L} = J\bar{W}$, де J – тензор інерції. Вважатимемо дрон кулею радіуса R_s (в м), масою m_s (в кг), на відстані від центру якого розташовані матеріальні точки масою m_m (в кг), що схематично зображено на рис. 1.2.

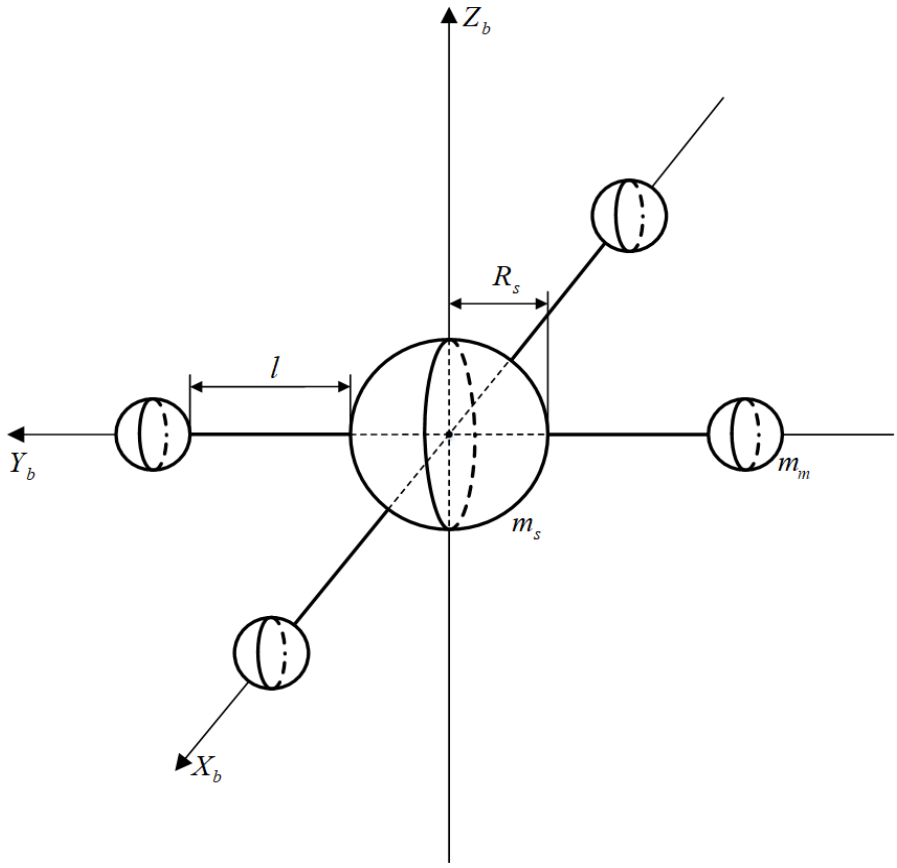


Рисунок 1.2 – Схематичне представлення конструкції дрона

Оскільки вважаємо дрон симетричним тілом, то його тензор інерції матиме вигляд

$$J = \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix}. \tag{1.27}$$

Враховуючи схему за рис. 1.7, можемо задати формули компонентів тензора інерції:

$$J_x = J_y = \frac{2m_s R_s^2}{5} + 2l^2 m_m;$$

$$J_z = \frac{2m_s R_s^2}{5} + 4l^2 m_m.$$

Вектор моменту обертальної сили в рухомій системі координат запишемо наступним чином:

$$\bar{M}_b = \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}, \quad t_0 \le t \le T; \tag{1.28}$$

де $\tau_\varphi \equiv \tau_\varphi(t)$, $\tau_\theta \equiv \tau_\theta(t)$, $\tau_\psi \equiv \tau_\psi(t)$ – моменти обертальної сили крену, тангажу та рискання відповідно.

Враховуючи (1.17) та (1.18), маємо вирази для τ_φ, τ_θ, τ_ψ:

$$\begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk_L(\Omega_4^2 - \Omega_2^2) \\ lk_L(\Omega_3^2 - \Omega_1^2) \\ b(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix}; \tag{1.29}$$

$$\bar{T} = k_L(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \quad t_0 \le t \le T, \tag{1.30}$$

де b – коефіцієнт обертального моменту.

Підставимо (1.27) – (1.29) в (1.26) та отримаємо:

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = J^{-1}\left(\begin{bmatrix} 0 & r & -q \\ -r & 0 & p \\ q & -p & 0 \end{bmatrix} \begin{bmatrix} J_x & 0 & 0 \\ 0 & J_y & 0 \\ 0 & 0 & J_z \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \begin{bmatrix} \tau_\varphi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} \right); \tag{1.31}$$

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{J_y - J_z}{J_x} qr\\ \frac{J_z - J_x}{J_y} pr\\ \frac{J_x - J_y}{J_z} pq\end{bmatrix} + \begin{bmatrix}\frac{1}{J_x}\tau_\varphi\\ \frac{1}{J_y}\tau_\theta\\ \frac{1}{J_z}\tau_\psi\end{bmatrix}, \quad t_0 \le t \le T. \tag{1.32}$$

Проведемо спрощення отриманих рівнянь, припустивши за джерелом [12], що кути φ і θ доволі малі. Тоді:

$$\begin{bmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix} = W_\eta \begin{bmatrix}p\\ q\\ r\end{bmatrix} \approx \begin{bmatrix}p\\ q\\ r\end{bmatrix}, \quad t_0 \le t \le T. \tag{1.33}$$

Аналогічно до (1.21), вважатимемо компоненти qr, pr, pq малими:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}\frac{1}{J_x}\tau_\varphi\\ \frac{1}{J_y}\tau_\theta\\ \frac{1}{J_z}\tau_\psi\end{bmatrix}, \quad t_0 \le t \le T. \tag{1.34}$$

Враховуючи (1.34), похідна від (1.33) матиме вигляд

$$\begin{bmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{bmatrix} = \begin{bmatrix}\frac{1}{J_x}\tau_\varphi\\ \frac{1}{J_y}\tau_\theta\\ \frac{1}{J_z}\tau_\psi\end{bmatrix}, \quad t_0 \le t \le T. \tag{1.35}$$

Враховуючи (1.24) та (1.35), математична модель, що описує рух дрона-квадрокоптера в просторі, набуває наступного вигляду:

$$\begin{cases} \ddot{x} = \left(C_\psi S_\theta C_\varphi + S_\psi S_\varphi\right)\dfrac{T}{m} - \dfrac{A_x}{m}\dot{x}\,, \\ \ddot{y} = \left(S_\psi S_\theta C_\varphi + C_\psi S_\varphi\right)\dfrac{T}{m} - \dfrac{A_y}{m}\dot{y}\,, \\ \ddot{z} = C_\varphi C_\theta \dfrac{T}{m} - g - \dfrac{A_z}{m}\dot{z}\,, \\ \ddot{\varphi} = \dfrac{1}{J_x}\tau_\varphi\,, \\ \ddot{\theta} = \dfrac{1}{J_y}\tau_\theta\,, \\ \ddot{\psi} = \dfrac{1}{J_z}\tau_\psi\,. \end{cases} \tag{1.36}$$

1.4 Лінеаризація системи

Відповідно до поставленої задачі синтезу керування, нас цікавить лінійне наближення системи (1.36). Оскільки напрям дії гвинтів дрона вже враховано в рівняннях, візьмемо за керування квадрати кутових швидкостей роторів дрона:

$$u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix}, \quad u_i(t) = \Omega_i^2(t), \quad t_0 \le t \le T. \tag{1.37}$$

Отже матимемо таку математичну модель, що описує рух дрона-квадрокоптера у кожен момент часу t ($t_0 \le t \le T$):

$$\begin{cases} \ddot{x} = \left(C_\psi S_\theta C_\varphi + S_\psi S_\varphi\right)\dfrac{k_L}{m}(u_1 + u_2 + u_3 + u_4) - \dfrac{A_x}{m}\dot{x}, \\ \ddot{y} = \left(S_\psi S_\theta C_\varphi + C_\psi S_\varphi\right)\dfrac{k_L}{m}(u_1 + u_2 + u_3 + u_4) - \dfrac{A_y}{m}\dot{y}, \\ \ddot{z} = C_\varphi C_\theta \dfrac{k_L}{m}(u_1 + u_2 + u_3 + u_4) - g - \dfrac{A_z}{m}\dot{z}, \\ \ddot{\varphi} = \dfrac{k_L}{J_x} l(u_4 - u_2), \\ \ddot{\theta} = \dfrac{k_L}{J_y} l(u_3 - u_1), \\ \ddot{\psi} = \dfrac{b}{J_z}(-u_1 + u_2 - u_3 + u_4). \end{cases} \tag{1.38}$$

Вдавшись до формальних роздумів, видно, що в системі (1.38) рівняння з $\ddot{x}$, $\ddot{y}$, $\ddot{z}$ містять доданки, що за фізичним змістом вимірюються в $\frac{\text{м}}{\text{с}^2}$, тобто

прискорення. Доданки $\left(C_\psi S_\theta C_\varphi + S_\psi S_\varphi\right)\frac{k_L}{m}(u_1+u_2+u_3+u_4)$ та $\left(S_\psi S_\theta C_\varphi + C_\psi S_\varphi\right)\frac{k_L}{m}(u_1+u_2+u_3+u_4)$ представляють собою прискорення, спричинене тягою гвинтів дрона. Вочевидь, рух дрона в площині $O_3X_3Y_3$ відбувається за рівняннями з $\ddot{x}$, $\ddot{y}$, й для руху в площині $O_3X_3Y_3$, взагалі-то, має виконуватись рівність прискорення вільного падіння та прискорення, спричиненого тягою гвинтів дрона, тобто $\frac{T}{m} \approx g$. Тоді маємо спрощену модель на $[t_0; T]$:

$$\begin{cases} \ddot{x} = gS_\theta - \dfrac{A_x}{m}\dot{x}, \\ \ddot{y} = -gS_\varphi - \dfrac{A_y}{m}\dot{y}, \\ \ddot{z} = C_\varphi C_\theta \dfrac{k_L}{m}(u_1+u_2+u_3+u_4) - g - \dfrac{A_z}{m}\dot{z}, \\ \ddot{\varphi} = \dfrac{k_L}{J_x}(u_4-u_2), \\ \ddot{\theta} = \dfrac{k_L}{J_y}(u_3-u_1), \\ \ddot{\psi} = \dfrac{b}{J_z}(-u_1+u_2-u_3+u_4). \end{cases} \tag{1.39}$$

Вдавшись до лінеаризації, отримаємо систему

$$\begin{cases} \ddot{x} = g\theta - \dfrac{A_x}{m}\dot{x}, \\ \ddot{y} = -g\varphi - \dfrac{A_y}{m}\dot{y}, \\ \ddot{z} = \dfrac{k_L}{m}(u_1+u_2+u_3+u_4) - g - \dfrac{A_z}{m}\dot{z}, \\ \ddot{\varphi} = \dfrac{k_L}{J_x}(u_4-u_2), \\ \ddot{\theta} = \dfrac{k_L}{J_y}(u_3-u_1), \\ \ddot{\psi} = \dfrac{b}{J_z}(-u_1+u_2-u_3+u_4). \end{cases} \tag{1.40}$$

ДОДАТОК Б

Дані та результати до проведених експериментів

Таблиця 1 – Параметри математичної моделі

Параметр	Величина	Розмірність
g	9,81	$\frac{м}{с^2}$
m	1,44	кг
I_x	0,0151	кг * $м^2$
I_y	0,0151	кг * $м^2$
I_z	0,0253	кг * $м^2$
A_x	0,25	$\frac{кг}{с}$
A_y	0,25	$\frac{кг}{с}$
A_z	0,25	$\frac{кг}{с}$
l	0,225	м

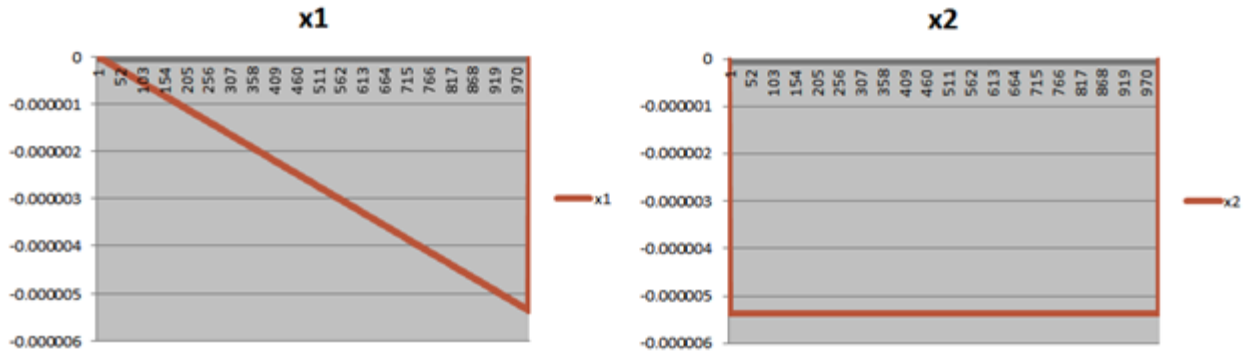


Рисунок 1.1 – Результати другого експерименту для x_1, x_2

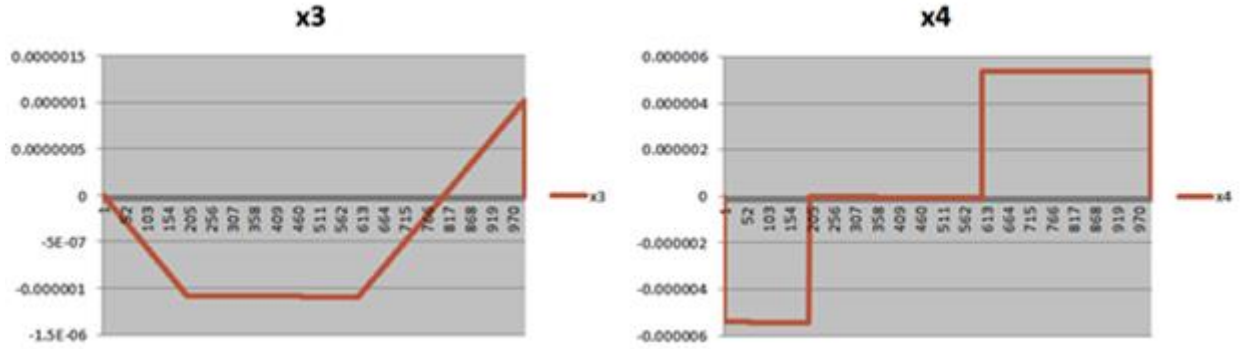


Рисунок 1.2 – Результати другого експерименту для x_3, x_4

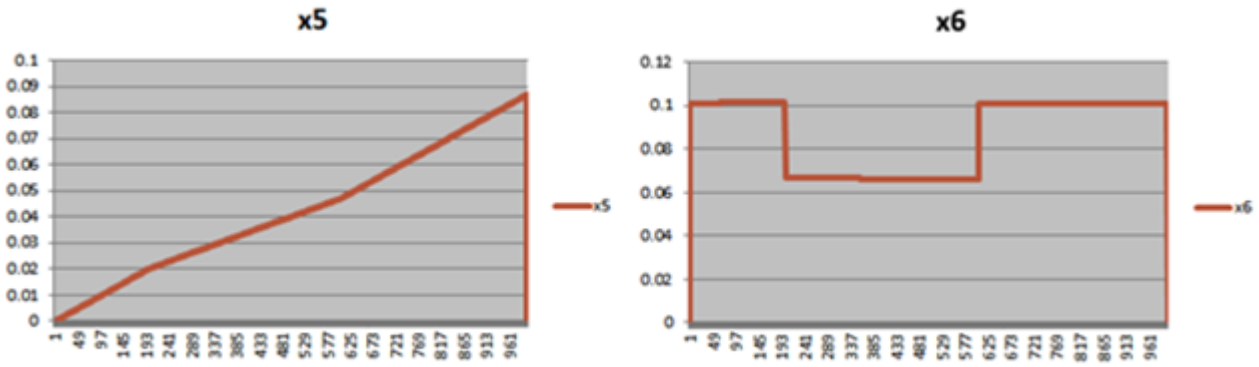


Рисунок 1.3 – Результати другого експерименту для x_5, x_6

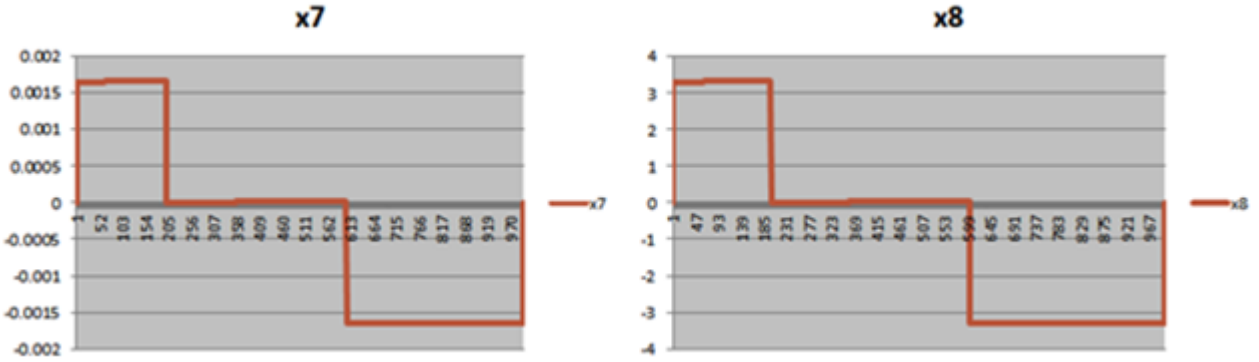


Рисунок 1.4 – Результати другого експерименту для x_7, x_8

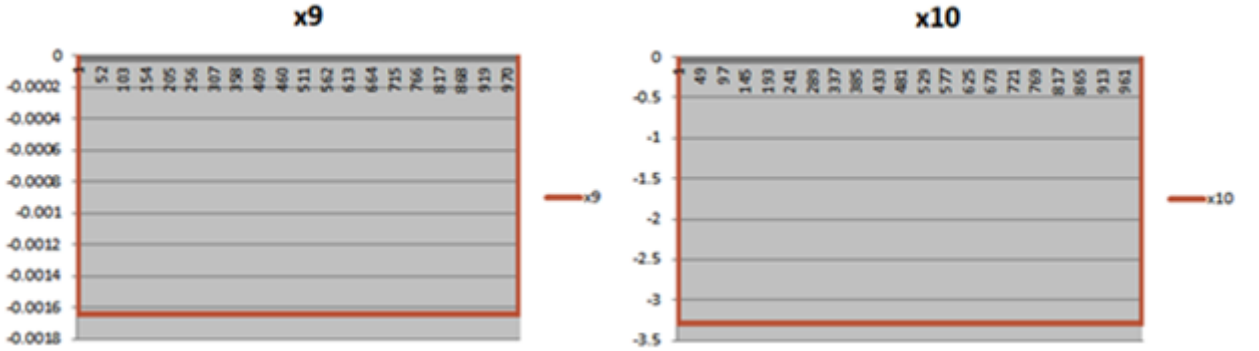


Рисунок 1.5 – Результати другого експерименту для x_9, x_{10}

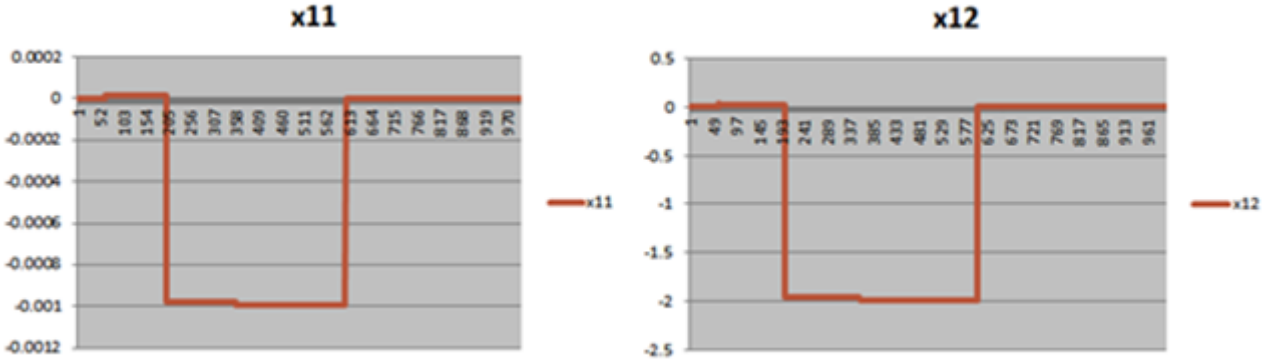


Рисунок 1.6 – Результати другого експерименту для x_{11}, x_{12}

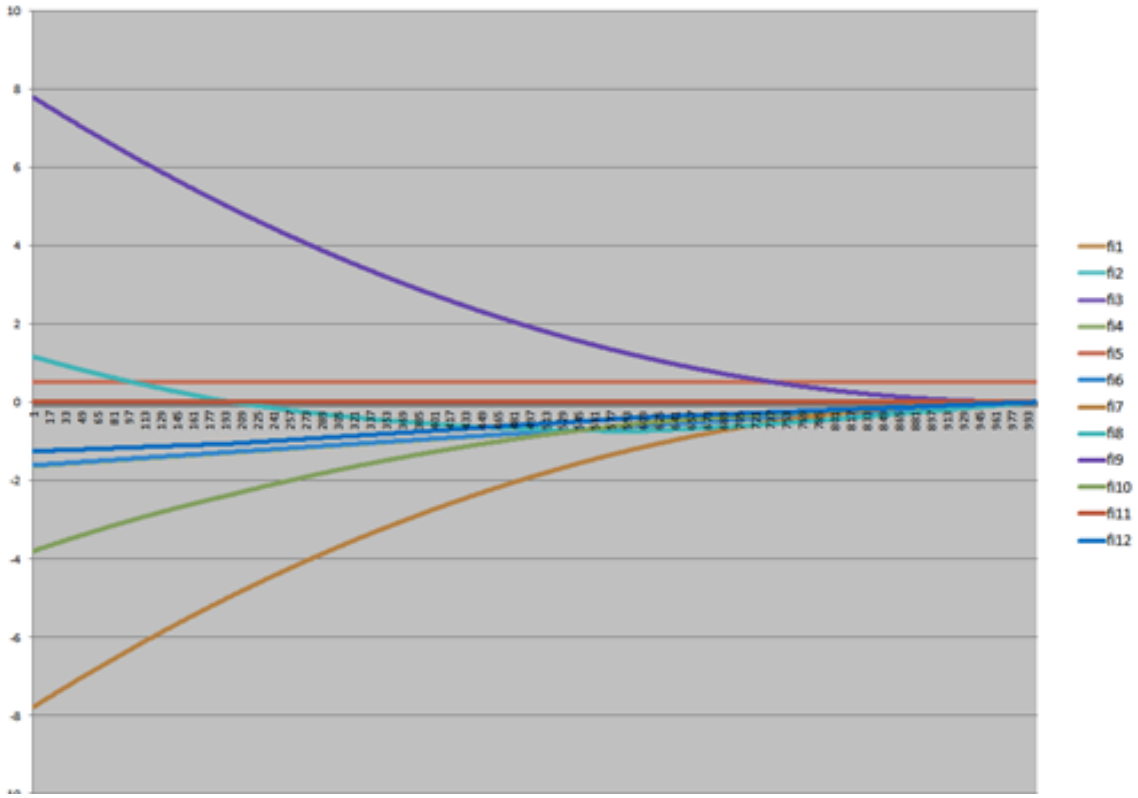


Рисунок 1.7 – Результати другого експерименту для $\psi_1, \dots, \psi_{12}$

ДОДАТОК В

Структура та код програми

1.1 Структура програми

- Class Constants - в класі визначимо такі значення відомих додатних сталих t_0, T, g, c, K, $\bar{A}_x$, $\bar{A}_y$, $\bar{A}_z$, $\bar{J}_x$, $\bar{J}_y$, $\bar{J}_z$ та інших; граничні умови на кінцях $[t_0, T]$ у вигляді векторів $X_{to} = \{x0, 0, y0, 0, z0, 0, \dots, 0\}$, $X_T = \{x1, 0, y1, 0, z1, 0, \dots, 0\}$ константи що відповідають фазовим обмеженням $fi\_min = -\pi / 2$, $fi\_max = \pi / 2$, $teta\_min = -\pi / 2$, $teta\_max = \pi / 2$, $psi\_min = -\pi / 2$, $psi\_max = \pi / 2$, обмеження на керування $U\_min = 0.0$, $U\_max = 1.0$, значення точності обчислень $eps_{shtr} = 0.01, eps_{um_{grad}} = 0.01$, $epsAlpha = 0.75$, та інші необхідні константи для роботи програми, як кількість розбиттів часового проміжку $[t_0, T]$, $N = 1000$, або ж максимальна кількість циклів обчислень для ітераційних процесів $end_M = 100$, початкові значення до ітераційних коефіцієнтів метода штрафних функціоналів $M = 0.2$, метода умовного градієнта $a\_beg = 0.1$;

- Class J – клас що містить методи для обчислення значення цільового функціоналу із врахуванням штрафних функціоналів за допомогою формули правих прямокутників;

- Class X – клас, що містить функції правих частин рівнянь системи (2.2);

- Class Fi – клас, що містить функції правих частин рівнянь системи (2.21) та (2.22);

- Class Gradient_J – клас, що містить функції правих частин рівнянь (2.23);

- Class RungeKutta_forFi – клас, що реалізує розв’язок методу Рунге-Кутта четвертого порядку точності для задачі Коші, що представлена як (2.21) та (3.22);

- Class RungeKutta_forX – клас, що реалізує розв’язок методу Рунге-Кутта четвертого порядку точності для задачі Коші, що представлена як (2.2) та (2.6);

- Class Jk – клас, що реалізовую метод обчислення скалярного добутку за формулою (1.69a).

- Class Norm– клас, що реалізовую метод обчислення норми за формулою (1.69b).

Модуль Constants:

```
package com.company;
public class Constants {
    //Стеля циклів обчислень
    public static final int end_M = 100;
    public static final int end_MSHtraf = 15;
    //Кількість розбиттів прміжку [t0; T]
    public static final int N = 1000;
    public static final double t0 = 0.0;//s
    public static final double T = 1.0;//s
    public static final double g = 9.81;//m/s^2
    public static final double m = 1.44;//kg
    public static final double b = 1.0;//???
    public static final double Jx = 0.0151;//kg/s
    public static final double Jy = 0.0151;//kg/s
    public static final double Jz = 0.0253;//kg/s
    public static final double Ax = 0.25;//kg/s
    public static final double Ay = 0.25;//kg/s
    public static final double Az = 0.25;//kg/s
    public static final double l = 0.225;//kg/s
    public static final double u_min = 5;
    public static final double u_max = 105;
    public static final double k_l = 1.0;
    public static final double K = k_l * (u_max - u_min) / m;
    public static final double _Ax = Ax / m;
    public static final double _Ay = Ay / m;
    public static final double _Az = Az / m;
    public static final double _Jx = k_l * (u_max - u_min) / Jx;
    public static final double _Jy = k_l * (u_max - u_min) / Jy;
    public static final double _Jz = b * (u_max - u_min) / Jz;
    public static final double c = (4 * u_min) / (u_max - u_min);
    public static final double U_min = 0.0;
    public static final double U_max = 1.0;
    public static final double x0 = 0.0;
    public static final double y0 = 0.0;
    public static final double z0 = 0.0;
    public static final double x1 = 1.0;
    public static final double y1 = 1.0;
    public static final double z1 = 1.0;
    public static final double fi_min = - Math.PI / 2;
    public static final double fi_max = Math.PI / 2;
    public static final double teta_min = - Math.PI / 2;
    public static final double teta_max = Math.PI / 2;
    public static final double psi_min = - Math.PI / 2;
    public static final double psi_max = Math.PI / 2;
    public static final double angle_min = - Math.PI / 2;
    public static final double angle_max = Math.PI / 2;
    public static final double[] X_t0 = {x0, 0, y0, 0, z0, 0, 0, 0, 0, 0, 0, 0};
    public static final double[] X_T = {x1, 0, y1, 0, z1, 0, 0, 0, 0, 0, 0, 0};
    public static final double eps_shtr = 0.01;
    public static final double eps_um_grad = 0.01;
```

```
    public static final double a_beg = 0.1;
    public static final double epsAlpha = 0.75;
    public static final double M = 0.0001;
}
```

Модуль Main:

```
package com.company;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
public class Main {
    public static void main(String[] args) {
        int k_g;
        int k_sh = 0;
        double[] t = new double[Constants.N + 1];
        t[0] = Constants.t0;
        double h = (Constants.T - Constants.t0) / Constants.N;
        for (int i = 1; i <= Constants.N; i++) {
            t[i] = t[i - 1] + h;
        }
        double a_k = Constants.a_beg;
        double[] uk1 = new double[Constants.N + 1];
        double[] _uk1 = new double[Constants.N + 1];
        double[] uk1_new = new double[Constants.N + 1];
        double[] uk2 = new double[Constants.N + 1];
        double[] _uk2 = new double[Constants.N + 1];
        double[] uk2_new = new double[Constants.N + 1];
        double[] uk3 = new double[Constants.N + 1];
        double[] _uk3 = new double[Constants.N + 1];
        double[] uk3_new = new double[Constants.N + 1];
        double[] uk4 = new double[Constants.N + 1];
        double[] _uk4 = new double[Constants.N + 1];
        double[] uk4_new = new double[Constants.N + 1];
        double[] x1 = new double[Constants.N + 1];
        double[] x1_new = new double[Constants.N + 1];
        double[] x2 = new double[Constants.N + 1];
        double[] x2_new = new double[Constants.N + 1];
        double[] x3 = new double[Constants.N + 1];
        double[] x3_new = new double[Constants.N + 1];
        double[] x4 = new double[Constants.N + 1];
        double[] x4_new = new double[Constants.N + 1];
        double[] x5 = new double[Constants.N + 1];
        double[] x5_new = new double[Constants.N + 1];
        double[] x6 = new double[Constants.N + 1];
        double[] x6_new = new double[Constants.N + 1];
        double[] x7 = new double[Constants.N + 1];
        double[] x7_new = new double[Constants.N + 1];
        double[] x8 = new double[Constants.N + 1];
        double[] x8_new = new double[Constants.N + 1];
        double[] x9 = new double[Constants.N + 1];
```

```
double[] x9_new = new double[Constants.N + 1];
double[] x10 = new double[Constants.N + 1];
double[] x10_new = new double[Constants.N + 1];
double[] x11 = new double[Constants.N + 1];
double[] x11_new = new double[Constants.N + 1];
double[] x12 = new double[Constants.N + 1];
double[] x12_new = new double[Constants.N + 1];
double[] fi1 = new double[Constants.N + 1];
double[] fi2 = new double[Constants.N + 1];
double[] fi3 = new double[Constants.N + 1];
double[] fi4 = new double[Constants.N + 1];
double[] fi5 = new double[Constants.N + 1];
double[] fi6 = new double[Constants.N + 1];
double[] fi7 = new double[Constants.N + 1];
double[] fi8 = new double[Constants.N + 1];
double[] fi9 = new double[Constants.N + 1];
double[] fi10 = new double[Constants.N + 1];
double[] fi11 = new double[Constants.N + 1];
double[] fi12 = new double[Constants.N + 1];
double u_min = Constants.U_min;
double u_max = Constants.U_max;
double u_mid = 0.2;
for (int i = 0; i <= Constants.N; i++) {
    uk1[i] = u_mid;
    uk2[i] = u_mid;
    uk3[i] = u_mid;
    uk4[i] = u_mid;
}
double M = Constants.M;
RungeKutta_forX RKX;
RungeKutta_forX RKX_new;
RungeKutta_forFi RKFI;
J j_sh;
J j_sh_new;
J jj;
J jj_new;
Gradient_J Gj;
double min_JJJ = 200;
boolean a;
boolean bT1;
boolean bT2;
boolean bT3;
//Метод штрафів
do {
    k_g = 0;
    RKX = new RungeKutta_forX(M, t, uk1, uk2, uk3, uk4);
    //Записали X
    System.arraycopy(RKX.x1, 0, x1, 0, Constants.N + 1);
    System.arraycopy(RKX.x2, 0, x2, 0, Constants.N + 1);
    System.arraycopy(RKX.x3, 0, x3, 0, Constants.N + 1);
    System.arraycopy(RKX.x4, 0, x4, 0, Constants.N + 1);
    System.arraycopy(RKX.x5, 0, x5, 0, Constants.N + 1);
```

```
System.arraycopy(RKX.x6, 0, x6, 0, Constants.N + 1);
System.arraycopy(RKX.x7, 0, x7, 0, Constants.N + 1);
System.arraycopy(RKX.x8, 0, x8, 0, Constants.N + 1);
System.arraycopy(RKX.x9, 0, x9, 0, Constants.N + 1);
System.arraycopy(RKX.x10, 0, x10, 0, Constants.N + 1);
System.arraycopy(RKX.x11, 0, x11, 0, Constants.N + 1);
System.arraycopy(RKX.x12, 0, x12, 0, Constants.N + 1);
j_sh = new J(M, t, uk1, uk2, uk3, uk4, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12);
//Метод умовного градієнта
do {
    jj = new J(M, t, uk1, uk2, uk3, uk4, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12);
    //Нашли fi
    RKFI = new RungeKutta_forFi(M, t, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12);
    System.arraycopy(RKFI.fi1, 0, fi1, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi2, 0, fi2, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi3, 0, fi3, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi4, 0, fi4, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi5, 0, fi5, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi6, 0, fi6, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi7, 0, fi7, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi8, 0, fi8, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi9, 0, fi9, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi10, 0, fi10, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi11, 0, fi11, 0, Constants.N + 1);
    System.arraycopy(RKFI.fi12, 0, fi12, 0, Constants.N + 1);
    //Порахували градієнт
    Gj = new Gradient_J();
    double[] grad1 = Gj.grad_1(uk1, fi6, fi10, fi12);
    double[] grad2 = Gj.grad_2(uk2, fi6, fi8, fi12);
    double[] grad3 = Gj.grad_3(uk3, fi6, fi10, fi12);
    double[] grad4 = Gj.grad_4(uk4, fi6, fi8, fi12);
    for (int i = 0; i <= Constants.N; i++) {
        if (grad1[i] >= 0)
            _uk1[i] = u_min;
        else
            _uk1[i] = u_max;
        if (grad2[i] >= 0)
            _uk2[i] = u_min;
        else
            _uk2[i] = u_max;
        if (grad3[i] >= 0)
            _uk3[i] = u_min;
        else
            _uk3[i] = u_max;
        if (grad4[i] >= 0)
            _uk4[i] = u_min;
        else
            _uk4[i] = u_max;
    }
    //a_k = getA(t, uk1,uk2,uk3,uk4,_uk1, _uk2, _uk3, _uk4);
    for (int i = 0; i <= Constants.N; i++) {
        double temp1 = uk1[i] + a_k * (_uk1[i] - uk1[i]);
```

```
            if (temp1 < u_min)
                uk1_new[i] = u_min;
            else uk1_new[i] = Math.min(temp1, u_max);
            double temp2 = uk2[i] + a_k * (_uk2[i] - uk2[i]);
            if (temp2 < u_min)
                uk2_new[i] = u_min;
            else uk2_new[i] = Math.min(temp2, u_max);
            double temp3 = uk3[i] + a_k * (_uk3[i] - uk3[i]);
            if (temp3 < u_min)
                uk3_new[i] = u_min;
            else uk3_new[i] = Math.min(temp3, u_max);
            double temp4 = uk4[i] + a_k * (_uk4[i] - uk4[i]);
            if (temp4 < u_min)
                uk4_new[i] = u_min;
            else uk4_new[i] = Math.min(temp4, u_max);
        }
        RKX_new = new RungeKutta_forX(M, t, uk1_new, uk2_new, uk3_new, uk4_new);
        System.arraycopy(RKX_new.x1, 0, x1_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x2, 0, x2_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x3, 0, x3_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x4, 0, x4_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x5, 0, x5_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x6, 0, x6_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x7, 0, x7_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x8, 0, x8_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x9, 0, x9_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x10, 0, x10_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x11, 0, x11_new, 0, Constants.N + 1);
        System.arraycopy(RKX_new.x12, 0, x12_new, 0, Constants.N + 1);
        jj_new = new J(M, t, uk1_new, uk2_new, uk3_new, uk4_new, x1_new, x2_new, x3_new,
x4_new, x5_new, x6_new, x7_new, x8_new, x9_new, x10_new, x11_new, x12_new);
        Jk jk = new Jk(grad1, grad2, grad3, grad4, uk1, uk2, uk3, uk4, _uk1, _uk2, _uk3, _uk4, t);
        //Бінарний пошук
        a_k = findOptimalAlpha(jj.getResult(), jj_new.getResult(), jk.getResult());
        System.arraycopy(uk1_new, 0, uk1, 0, Constants.N + 1);
        System.arraycopy(uk2_new, 0, uk2, 0, Constants.N + 1);
        System.arraycopy(uk3_new, 0, uk3, 0, Constants.N + 1);
        System.arraycopy(uk4_new, 0, uk4, 0, Constants.N + 1);
        System.arraycopy(x1_new, 0, x1, 0, Constants.N + 1);
        System.arraycopy(x2_new, 0, x2, 0, Constants.N + 1);
        System.arraycopy(x3_new, 0, x3, 0, Constants.N + 1);
        System.arraycopy(x4_new, 0, x4, 0, Constants.N + 1);
        System.arraycopy(x5_new, 0, x5, 0, Constants.N + 1);
        System.arraycopy(x6_new, 0, x6, 0, Constants.N + 1);
        System.arraycopy(x7_new, 0, x7, 0, Constants.N + 1);
        System.arraycopy(x8_new, 0, x8, 0, Constants.N + 1);
        System.arraycopy(x9_new, 0, x9, 0, Constants.N + 1);
        System.arraycopy(x10_new, 0, x10, 0, Constants.N + 1);
        System.arraycopy(x11_new, 0, x11, 0, Constants.N + 1);
        System.arraycopy(x12_new, 0, x12, 0, Constants.N + 1);
        k_g++;
        a = Math.abs(jj.getResult() - jj_new.getResult()) >= Constants.eps_um_grad;
```

```
            if (k_g > Constants.end_M)
                break;
        } while (a);
        j_sh_new = jj_new;
        Gradient_J gj = new Gradient_J();
        double[] g1 = gj.grad_1(uk1_new, fi6, fi10, fi12);
        double[] g2 = gj.grad_2(uk2_new, fi6, fi8, fi12);
        double[] g3 = gj.grad_3(uk3_new, fi6, fi10, fi12);
        double[] g4 = gj.grad_4(uk4_new, fi6, fi8, fi12);
        Norm norm = new Norm();
        double norma = norm.norma(g1, g2, g3, g4, h);
        System.out.println(k_sh + ".m = " + M + ", k_g = " + k_g  +", J = " + j_sh.getResult() + "
Norma_Grad = " + norma);
System.out.println("_____________________________________________________________
____________________");
        k_sh++;
        if (min_JJJ > j_sh.getResult()) {
            min_JJJ = j_sh.getResult();
        }
        bT1 = Math.abs(x1_new[Constants.N] - Constants.X_T[0]) >= Constants.eps_shtr;
        bT2 = Math.abs(x3_new[Constants.N] - Constants.X_T[2]) >= Constants.eps_shtr;
        bT3 = Math.abs(x5_new[Constants.N] - Constants.X_T[4]) >= Constants.eps_shtr;
        if (k_sh > Constants.end_MSHtraf)
            break;
        break;
    } while (bT1 && bT2 && bT3);
    {
        try {
            //x
            BufferedWriter writer_x1 = new BufferedWriter(new FileWriter("x1.txt"));
            BufferedWriter writer_x2 = new BufferedWriter(new FileWriter("x2.txt"));
            BufferedWriter writer_x3 = new BufferedWriter(new FileWriter("x3.txt"));
            BufferedWriter writer_x4 = new BufferedWriter(new FileWriter("x4.txt"));
            BufferedWriter writer_x5 = new BufferedWriter(new FileWriter("x5.txt"));
            BufferedWriter writer_x6 = new BufferedWriter(new FileWriter("x6.txt"));
            BufferedWriter writer_x7 = new BufferedWriter(new FileWriter("x7.txt"));
            BufferedWriter writer_x8 = new BufferedWriter(new FileWriter("x8.txt"));
            BufferedWriter writer_x9 = new BufferedWriter(new FileWriter("x9.txt"));
            BufferedWriter writer_x10 = new BufferedWriter(new FileWriter("x10.txt"));
            BufferedWriter writer_x11 = new BufferedWriter(new FileWriter("x11.txt"));
            BufferedWriter writer_x12 = new BufferedWriter(new FileWriter("x12.txt"));
            //u
            BufferedWriter writer_u1 = new BufferedWriter(new FileWriter("u1.txt"));
            BufferedWriter writer_u2 = new BufferedWriter(new FileWriter("u2.txt"));
            BufferedWriter writer_u3 = new BufferedWriter(new FileWriter("u3.txt"));
            BufferedWriter writer_u4 = new BufferedWriter(new FileWriter("u4.txt"));
            //fi
            BufferedWriter writer_fi1 = new BufferedWriter(new FileWriter("fi1.txt"));
            BufferedWriter writer_fi2 = new BufferedWriter(new FileWriter("fi2.txt"));
            BufferedWriter writer_fi3 = new BufferedWriter(new FileWriter("fi3.txt"));
            BufferedWriter writer_fi4 = new BufferedWriter(new FileWriter("fi4.txt"));
            BufferedWriter writer_fi5 = new BufferedWriter(new FileWriter("fi5.txt"));
```

```
BufferedWriter writer_fi6 = new BufferedWriter(new FileWriter("fi6.txt"));
BufferedWriter writer_fi7 = new BufferedWriter(new FileWriter("fi7.txt"));
BufferedWriter writer_fi8 = new BufferedWriter(new FileWriter("fi8.txt"));
BufferedWriter writer_fi9 = new BufferedWriter(new FileWriter("fi9.txt"));
BufferedWriter writer_fi10 = new BufferedWriter(new FileWriter("fi10.txt"));
BufferedWriter writer_fi11 = new BufferedWriter(new FileWriter("fi11.txt"));
BufferedWriter writer_fi12 = new BufferedWriter(new FileWriter("fi12.txt"));
for (int i = 0; i <= Constants.N; i++) {
    String str_x1 = x1_new[i] + "\n";
    String str_x2 = x2_new[i] + "\n";
    String str_x3 = x3_new[i] + "\n";
    String str_x4 = x4_new[i] + "\n";
    String str_x5 = x5_new[i] + "\n";
    String str_x6 = x6_new[i] + "\n";
    String str_x7 = x7_new[i] + "\n";
    String str_x8 = x8_new[i] + "\n";
    String str_x9 = x9_new[i] + "\n";
    String str_x10 = x10_new[i] + "\n";
    String str_x11 = x11_new[i] + "\n";
    String str_x12 = x12_new[i] + "\n";
    writer_x1.write(str_x1);
    writer_x2.write(str_x2);
    writer_x3.write(str_x3);
    writer_x4.write(str_x4);
    writer_x5.write(str_x5);
    writer_x6.write(str_x6);
    writer_x7.write(str_x7);
    writer_x8.write(str_x8);
    writer_x9.write(str_x9);
    writer_x10.write(str_x10);
    writer_x11.write(str_x11);
    writer_x12.write(str_x12);
    //fi
    String str_fi1 = fi1[i] + "\n";
    String str_fi2 = fi2[i] + "\n";
    String str_fi3 = fi3[i] + "\n";
    String str_fi4 = fi4[i] + "\n";
    String str_fi5 = fi5[i] + "\n";
    String str_fi6 = fi6[i] + "\n";
    String str_fi7 = fi7[i] + "\n";
    String str_fi8 = fi8[i] + "\n";
    String str_fi9 = fi9[i] + "\n";
    String str_fi10 = fi10[i] + "\n";
    String str_fi11 = fi11[i] + "\n";
    String str_fi12 = fi12[i] + "\n";
    writer_fi1.write(str_fi1);
    writer_fi2.write(str_fi2);
    writer_fi3.write(str_fi3);
    writer_fi4.write(str_fi4);
    writer_fi5.write(str_fi5);
    writer_fi6.write(str_fi6);
    writer_fi7.write(str_fi7);
```

```
                writer_fi8.write(str_fi8);
                writer_fi9.write(str_fi9);
                writer_fi10.write(str_fi10);
                writer_fi11.write(str_fi11);
                writer_fi12.write(str_fi12);
                //u
                String str_u1 = uk1_new[i] + "\n";
                String str_u2 = uk2_new[i] + "\n";
                String str_u3 = uk3_new[i] + "\n";
                String str_u4 = uk4_new[i] + "\n";
                writer_u1.write(str_u1);
                writer_u2.write(str_u2);
                writer_u3.write(str_u3);
                writer_u4.write(str_u4);
            }
            writer_x1.close();
            writer_x2.close();
            writer_x3.close();
            writer_x4.close();
            writer_x5.close();
            writer_x6.close();
            writer_x7.close();
            writer_x8.close();
            writer_x9.close();
            writer_x10.close();
            writer_x11.close();
            writer_x12.close();
            writer_fi1.close();
            writer_fi2.close();
            writer_fi3.close();
            writer_fi4.close();
            writer_fi5.close();
            writer_fi6.close();
            writer_fi7.close();
            writer_fi8.close();
            writer_fi9.close();
            writer_fi10.close();
            writer_fi11.close();
            writer_fi12.close();
            writer_u1.close();
            writer_u2.close();
            writer_u3.close();
            writer_u4.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
//Перебір значень альфа
public static double getA(double[] t, double[] uk1, double[] uk2, double[] uk3, double[] uk4,
double[] _uk1, double[] _uk2, double[] _uk3, double[] _uk4) {
    double u_min = 0.0;
```

```
        double u_max = 1.0;
        double[] uk1_new = new double[Constants.N + 1];
        double[] uk2_new = new double[Constants.N + 1];
        double[] uk3_new = new double[Constants.N + 1];
        double[] uk4_new = new double[Constants.N + 1];
        double[] res = new double[Constants.N];
        int Y = 0;
        for (double a_k = 0.0; a_k < 1.0; a_k += 0.001) {

            for (int i = 0; i <= Constants.N; i++) {
                double temp1 = uk1[i] + a_k * (_uk1[i] - uk1[i]);
                if (temp1 < u_min)
                    uk1_new[i] = u_min;
                else uk1_new[i] = Math.min(temp1, u_max);
                double temp2 = uk2[i] + a_k * (_uk2[i] - uk2[i]);
                if (temp2 < u_min)
                    uk2_new[i] = u_min;
                else uk2_new[i] = Math.min(temp2, u_max);
                double temp3 = uk3[i] + a_k * (_uk3[i] - uk3[i]);
                if (temp3 < u_min)
                    uk3_new[i] = u_min;
                else uk3_new[i] = Math.min(temp3, u_max);
                double temp4 = uk4[i] + a_k * (_uk4[i] - uk4[i]);
                if (temp4 < u_min)
                    uk4_new[i] = u_min;
                else uk4_new[i] = Math.min(temp4, u_max);
            }
            RungeKutta_forX rkx = new RungeKutta_forX(Constants.M, t, uk1_new, uk2_new,
uk3_new, uk4_new);
            J J1 = new J(Constants.M, t, uk1_new, uk2_new, uk3_new, uk4_new, rkx.x1, rkx.x2,
rkx.x3, rkx.x4, rkx.x5, rkx.x6, rkx.x7, rkx.x8, rkx.x9, rkx.x10, rkx.x11, rkx.x12);
            res[Y] = J1.getResult();
            System.out.println("J(" + Y + ") = " + J1.getResult());
            Y++;
        }
        int ind_min = 0;
        double min = 1000.0;
        for (int i = 0; i < Constants.N; i++) {
            if (res[i] < min) {
                min = res[i];
                ind_min = i;
            }
        }
        System.out.println("a_k = " + ind_min * 0.001 + " J(amin) = " + res[ind_min]);
        return ind_min * 0.001;
    }
    public static double findOptimalAlpha(double J_h_u_k, double J_h_u_k_plus_1, double
J_k_u_bar) {
        double low = 0.0;
        double high = 1.0;
        double alpha = 0.5; // Начальное значение α
        while (high - low > Constants.epsAlpha) {
```

```
            double left = Math.pow(alpha, 2);
            double right = (J_h_u_k - J_h_u_k_plus_1) / (Constants.epsAlpha * Math.abs(J_k_u_bar));
            if (left <= right) {
                high = alpha;
            } else {
                low = alpha;
            }
            alpha = (low + high) / 2.0;
        }
        return alpha;
    }
}
class J {
    private final double[] t;
    private final double[] u1;
    private final double[] u2;
    private final double[] u3;
    private final double[] u4;
    private final double[] x1;
    private final double[] x2;
    private final double[] x3;
    private final double[] x4;
    private final double[] x5;
    private final double[] x6;
    private final double[] x7;
    private final double[] x8;
    private final double[] x9;
    private final double[] x10;
    private final double[] x11;
    private final double[] x12;
    private double result;
    private final double m;
    public J(double m, double[] t, double[] u1, double[] u2, double[] u3, double[] u4, double[] x1,
double[] x2, double[] x3, double[] x4, double[] x5, double[] x6, double[] x7, double[] x8, double[]
x9, double[] x10, double[] x11, double[] x12) {
        this.t = t;
        this.u1 = u1;
        this.u2 = u2;
        this.u3 = u3;
        this.u4 = u4;
        this.x1 = x1;
        this.x2 = x2;
        this.x3 = x3;
        this.x4 = x4;
        this.x5 = x5;
        this.x6 = x6;
        this.x7 = x7;
        this.x8 = x8;
        this.x9 = x9;
        this.x10 = x10;
        this.x11 = x11;
        this.x12 = x12;
```

```
        this.m = m;
        rightRectangles();
    }
    private void rightRectangles() {
        result = 0.0;
        double result_1 = 0.0;
        double result_2 = 0.0;
        double result_3 = 0.0;
        double f0;
        double f0_P1;
        double f0_P2;
        double f0_P3;
        for (int i = 1; i < Constants.N; i++) {
            f0 = function_f0(u1[i], u2[i], u3[i], u4[i]);
            result_1 += f0 * (t[i] - t[i - 1]);
        }
        for (int i = 1; i < Constants.N; i++) {
            f0_P2 = function_f0_P2(m, x2[i], x4[i], x6[i], x8[i], x10[i], x12[i]);
            result_2 += f0_P2 * (t[i] - t[i - 1]);
        }
        //System.out.println("f0_P2 = " + result_2);
        for (int i = 1; i < Constants.N; i++) {
            f0_P3 = function_f0_P3(m, x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i],
x11[i], x12[i]);
            result_3 += f0_P3 * (t[i] - t[i - 1]);
        }
        f0_P1 = function_f0_P1(m, x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12);
        result = f0_P1 + result_1 + result_2 + result_3;
    }
    private double function_f0(double u1, double u2, double u3, double u4) {
        return Math.pow(u1, 2) + Math.pow(u2, 2) + Math.pow(u3, 2) + Math.pow(u4, 2);
    }
    private double function_f0_P1(double m, double[] x1, double[] x2, double[] x3, double[] x4,
double[] x5, double[] x6, double[] x7, double[] x8, double[] x9, double[] x10, double[] x11,
double[] x12) {
        double result = 0.0;
        result += Math.pow(x1[Constants.N] - Constants.X_T[0], 2);
        result += Math.pow(x2[Constants.N] - Constants.X_T[1], 2);
        result += Math.pow(x3[Constants.N] - Constants.X_T[2], 2);
        result += Math.pow(x4[Constants.N] - Constants.X_T[3], 2);
        result += Math.pow(x5[Constants.N] - Constants.X_T[4], 2);
        result += Math.pow(x6[Constants.N] - Constants.X_T[5], 2);
        result += Math.pow(x7[Constants.N] - Constants.X_T[6], 2);
        result += Math.pow(x8[Constants.N] - Constants.X_T[7], 2);
        result += Math.pow(x9[Constants.N] - Constants.X_T[8], 2);
        result += Math.pow(x10[Constants.N] - Constants.X_T[9], 2);
        result += Math.pow(x11[Constants.N] - Constants.X_T[10], 2);
        result += Math.pow(x12[Constants.N] - Constants.X_T[11], 2);
        return m * result;
    }
    private double function_f0_P2(double m, double x2, double x4, double x6, double x8, double
x10, double x12) {
```

```
        double result = 0.0;
        result += Math.pow(Math.E, -m * x2);
        result += Math.pow(Math.E, -m * x4);
        result += Math.pow(Math.E, -m * x6);
        result += Math.pow(Math.E, -m * x8);
        result += Math.pow(Math.E, -m * x10);
        result += Math.pow(Math.E, -m * x12);
        return result / m;
    }
    private double function_f0_P3(double m, double x1, double x2, double x3, double x4, double x5,
double x6, double x7, double x8, double x9, double x10, double x11, double x12) {
        double result = 0.0;
        result += Math.pow(Math.E, m * (x1 + Constants.angle_min) * (x1 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x2 + Constants.angle_min) * (x2 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x3 + Constants.angle_min) * (x3 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x4 + Constants.angle_min) * (x4 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x5 + Constants.angle_min) * (x5 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x6 + Constants.angle_min) * (x6 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x7 + Constants.angle_min) * (x7 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x8 + Constants.angle_min) * (x8 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x9 + Constants.angle_min) * (x9 + Constants.angle_max));
        result += Math.pow(Math.E, m * (x10 + Constants.angle_min) * (x10 +
Constants.angle_max));
        result += Math.pow(Math.E, m * (x11 + Constants.angle_min) * (x11 +
Constants.angle_max));
        result += Math.pow(Math.E, m * (x12 + Constants.angle_min) * (x12 +
Constants.angle_max));
        return result / m;
    }
    public double getResult() {
        return result;
    }
}
class X {
    public X() {
    }
    public double f_x1(double x2) {
        return x2;
    }
    public double f_x2(double x9, double x2) {
        return Constants.g * x9 - Constants._Ax * x2;
    }
    public double f_x3(double x4) {
        return x4;
    }
    public double f_x4(double x7, double x4) {
        return -Constants.g * x7 - Constants._Ay * x4;
    }
    public double f_x5(double x6) {
        return x6;
    }
    public double f_x6(double u1, double u2, double u3, double u4, double x6) {
```

```
        return Constants.K * ((u1 + u2 + u3 + u4) + Constants.c) - Constants.g - Constants._Az * x6;
    }
    public double f_x7(double x8) {
        return x8;
    }
    public double f_x8(double u4, double u2) {
        return Constants._Jx * (u4 - u2);
    }
    public double f_x9(double x10) {
        return x10;
    }
    public double f_x10(double u3, double u1) {
        return Constants._Jy * (u3 - u1);
    }
    public double f_x11(double x12) {
        return x12;
    }
    public double f_x12(double u1, double u2, double u3, double u4) {
        return Constants._Jz * (-u1 + u2 - u3 + u4);
    }
}
class Fi {
    public Fi() {
    }
    public double f_fi_1() {
        return 0.0;
    }
    public double f_fi_2(double m, double x2, double fi1, double fi2) {
        return -Math.pow(Math.E, -m * x2) - fi1 + Constants._Ax * fi2;
    }
    public double f_fi_3() {
        return 0.0;
    }
    public double f_fi_4(double m, double x4, double fi3, double fi4) {
        return -Math.pow(Math.E, -m * x4) - fi3 + Constants._Ay * fi4;
    }
    public double f_fi_5() {
        return 0.0;
    }
    public double f_fi_6(double m, double x6, double fi5, double fi6) {
        return -Math.pow(Math.E, -m * x6) - fi5 + Constants._Az * fi6;
    }
    public double f_fi_7(double m, double x7, double fi4) {
        return 2 * x7 * Math.pow(Math.E, m * (x7 + Constants.angle_min) * (x7 +
Constants.angle_max)) + Constants.g * fi4;
    }
    public double f_fi_8(double m, double x8, double fi7) {
        return -Math.pow(Math.E, -m * x8) - fi7;
    }
    public double f_fi_9(double m, double x9, double fi2) {
        return 2 * x9 * Math.pow(Math.E, m * (x9 + Constants.angle_min) * (x9 +
Constants.angle_max)) - Constants.g * fi2;
```

```
    }
    public double f_fi_10(double m, double x10, double fi9) {
        return -Math.pow(Math.E, -m * x10) - fi9;
    }
    public double f_fi_11(double m, double x11) {
        return 2 * x11 * Math.pow(Math.E, m * (x11 + Constants.angle_min) * (x11 +
Constants.angle_max));
    }
    public double f_fi_12(double m, double x12, double fi11) {
        return -Math.pow(Math.E, -m * x12) - fi11;
    }
    public double f_fi_T_1(double m, double x1_T) {
        return -2.0 * m * (x1_T - Constants.X_T[0]);
    }
    public double f_fi_T_2(double m, double x2_T) {
        return -2.0 * m * (x2_T - Constants.X_T[1]);
    }
    public double f_fi_T_3(double m, double x3_T) {
        return -2.0 * m * (x3_T - Constants.X_T[2]);
    }
    public double f_fi_T_4(double m, double x4_T) {
        return -2.0 * m * (x4_T - Constants.X_T[3]);
    }
    public double f_fi_T_5(double m, double x5_T) {
        return -2.0 * m * (x5_T - Constants.X_T[4]);
    }
    public double f_fi_T_6(double m, double x6_T) {
        return -2.0 * m * (x6_T - Constants.X_T[5]);
    }
    public double f_fi_T_7(double m, double x7_T) {
        return -2.0 * m * (x7_T - Constants.X_T[6]);
    }
    public double f_fi_T_8(double m, double x8_T) {
        return -2.0 * m * (x8_T - Constants.X_T[7]);
    }
    public double f_fi_T_9(double m, double x9_T) {
        return -2.0 * m * (x9_T - Constants.X_T[8]);
    }
    public double f_fi_T_10(double m, double x10_T) {
        return -2.0 * m * (x10_T - Constants.X_T[9]);
    }
    public double f_fi_T_11(double m, double x11_T) {
        return -2.0 * m * (x11_T - Constants.X_T[10]);
    }
    public double f_fi_T_12(double m, double x12_T) {
        return -2.0 * m * (x12_T - Constants.X_T[11]);
    }
}
class Gradient_J {
    public Gradient_J() {
    }
    public double[] grad_1(double[] u1, double[] fi6, double[] fi10, double[] fi12) {
```

```
        double[] result = new double[Constants.N + 1];
        for (int i = 0; i <= Constants.N; i++) {
            result[i] = 2 * u1[i] - Constants.K * fi6[i] + Constants._Jy * fi10[i] + Constants._Jz * fi12[i];
        }
        return result;
    }
    public double[] grad_2(double[] u2, double[] fi6, double[] fi8, double[] fi12) {
        double[] result = new double[Constants.N + 1];
        for (int i = 0; i <= Constants.N; i++) {
            result[i] = 2 * u2[i] - Constants.K * fi6[i] + Constants._Jx * fi8[i] - Constants._Jz * fi12[i];
        }
        return result;
    }
    public double[] grad_3(double[] u3, double[] fi6, double[] fi10, double[] fi12) {
        double[] result = new double[Constants.N + 1];
        for (int i = 0; i <= Constants.N; i++) {
            result[i] = 2 * u3[i] - Constants.K * fi6[i] - Constants._Jy * fi10[i] + Constants._Jz * fi12[i];
        }
        return result;
    }
    public double[] grad_4(double[] u4, double[] fi6, double[] fi8, double[] fi12) {
        double[] result = new double[Constants.N + 1];
        for (int i = 0; i <= Constants.N; i++) {
            result[i] = 2 * u4[i] - Constants.K * fi6[i] - Constants._Jx * fi8[i] - Constants._Jz * fi12[i];
        }
        return result;
    }
}
class RungeKutta_forFi {
    private final int N = Constants.N;
    private final double m;
    public final double[] t;
    public final double[] fi1;
    public final double[] fi2;
    public final double[] fi3;
    public final double[] fi4;
    public final double[] fi5;
    public final double[] fi6;
    public final double[] fi7;
    public final double[] fi8;
    public final double[] fi9;
    public final double[] fi10;
    public final double[] fi11;
    public final double[] fi12;
    public final double[] x1;
    public final double[] x2;
    public final double[] x3;
    public final double[] x4;
    public final double[] x5;
    public final double[] x6;
    public final double[] x7;
    public final double[] x8;
```

```
    public final double[] x9;
    public final double[] x10;
    public final double[] x11;
    public final double[] x12;
    public RungeKutta_forFi(double m, double[] t, double[] x1, double[] x2, double[] x3, double[]
x4, double[] x5, double[] x6, double[] x7, double[] x8, double[] x9, double[] x10, double[] x11,
double[] x12) {
        this.m = m;
        fi1 = new double[N + 1];
        fi2 = new double[N + 1];
        fi3 = new double[N + 1];
        fi4 = new double[N + 1];
        fi5 = new double[N + 1];
        fi6 = new double[N + 1];
        fi7 = new double[N + 1];
        fi8 = new double[N + 1];
        fi9 = new double[N + 1];
        fi10 = new double[N + 1];
        fi11 = new double[N + 1];
        fi12 = new double[N + 1];
        this.t = t;
        this.x1 = x1;
        this.x2 = x2;
        this.x3 = x3;
        this.x4 = x4;
        this.x5 = x5;
        this.x6 = x6;
        this.x7 = x7;
        this.x8 = x8;
        this.x9 = x9;
        this.x10 = x10;
        this.x11 = x11;
        this.x12 = x12;
        calculate();
    }
    private void calculate() {
        Fi FI = new Fi();
        fi1[N] = FI.f_fi_T_1(m, x1[N]);
        fi2[N] = FI.f_fi_T_2(m, x2[N]);
        fi3[N] = FI.f_fi_T_3(m, x3[N]);
        fi4[N] = FI.f_fi_T_4(m, x4[N]);
        fi5[N] = FI.f_fi_T_5(m, x5[N]);
        fi6[N] = FI.f_fi_T_6(m, x6[N]);
        fi7[N] = FI.f_fi_T_7(m, x7[N]);
        fi8[N] = FI.f_fi_T_8(m, x8[N]);
        fi9[N] = FI.f_fi_T_9(m, x9[N]);
        fi10[N] = FI.f_fi_T_10(m, x10[N]);
        fi11[N] = FI.f_fi_T_11(m, x11[N]);
        fi12[N] = FI.f_fi_T_12(m, x12[N]);
        for (int i = N; i > 0; i--) {
            fi1[i - 1] = getRK("fi1", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
```

```
            fi2[i - 1] = getRK("fi2", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi3[i - 1] = getRK("fi3", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi4[i - 1] = getRK("fi4", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi5[i - 1] = getRK("fi5", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi6[i - 1] = getRK("fi6", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi7[i - 1] = getRK("fi7", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi8[i - 1] = getRK("fi8", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi9[i - 1] = getRK("fi9", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi10[i - 1] = getRK("fi10", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi11[i - 1] = getRK("fi11", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            fi12[i - 1] = getRK("fi12", t[i], fi1[i], fi2[i], fi3[i], fi4[i], fi5[i], fi6[i], fi7[i], fi8[i], fi9[i],
fi10[i], fi11[i], fi12[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
        }
    }
    public double getRK(String str, double t, double fi1, double fi2, double fi3, double fi4, double
fi5, double fi6, double fi7, double fi8, double fi9, double fi10, double fi11, double fi12, double x1,
double x2, double x3, double x4, double x5, double x6, double x7, double x8, double x9, double
x10, double x11, double x12) {
        Fi FI = new Fi();
        double h = (Constants.T - Constants.t0) / Constants.N;
        double k_fi1_1 = h * FI.f_fi_1();
        double k_fi2_1 = h * FI.f_fi_2(m, x2, fi1, fi2);
        double k_fi3_1 = h * FI.f_fi_3();
        double k_fi4_1 = h * FI.f_fi_4(m, x4, fi3, fi4);
        double k_fi5_1 = h * FI.f_fi_5();
        double k_fi6_1 = h * FI.f_fi_6(m, x6, fi5, fi6);
        double k_fi7_1 = h * FI.f_fi_7(m, x7, fi4);
        double k_fi8_1 = h * FI.f_fi_8(m, x8, fi7);
        double k_fi9_1 = h * FI.f_fi_9(m, x9, fi2);
        double k_fi10_1 = h * FI.f_fi_10(m, x12, fi9);
        double k_fi11_1 = h * FI.f_fi_11(m, x11);
        double k_fi12_1 = h * FI.f_fi_12(m, x12, fi11);
        //
        double k_fi1_2 = h * FI.f_fi_1();
        double k_fi2_2 = h * FI.f_fi_2(m, x2, fi1 - k_fi1_1 / 2, fi2 - k_fi2_1 / 2);
        double k_fi3_2 = h * FI.f_fi_3();
        double k_fi4_2 = h * FI.f_fi_4(m, x4, fi3 - k_fi3_1 / 2, fi4 - k_fi4_1 / 2);
        double k_fi5_2 = h * FI.f_fi_5();
        double k_fi6_2 = h * FI.f_fi_6(m, x6, fi5 - k_fi5_1 / 2, fi6 - k_fi6_1 / 2);
        double k_fi7_2 = h * FI.f_fi_7(m, x7, fi4 - k_fi4_1 / 2);
        double k_fi8_2 = h * FI.f_fi_8(m, x8, fi7 - k_fi7_1 / 2);
        double k_fi9_2 = h * FI.f_fi_9(m, x9, fi2 - k_fi2_1 / 2);
```

```
double k_fi10_2 = h * FI.f_fi_10(m, x12, fi9 - k_fi9_1 / 2);
double k_fi11_2 = h * FI.f_fi_11(m, x11);
double k_fi12_2 = h * FI.f_fi_12(m, x12, fi11 - k_fi11_1 / 2);
//
double k_fi1_3 = h * FI.f_fi_1();
double k_fi2_3 = h * FI.f_fi_2(m, x2, fi1 - k_fi1_2 / 2, fi2 - k_fi2_2 / 2);
double k_fi3_3 = h * FI.f_fi_3();
double k_fi4_3 = h * FI.f_fi_4(m, x4, fi3 - k_fi3_2 / 2, fi4 - k_fi4_2 / 2);
double k_fi5_3 = h * FI.f_fi_5();
double k_fi6_3 = h * FI.f_fi_6(m, x6, fi5 - k_fi5_2 / 2, fi6 - k_fi6_2 / 2);
double k_fi7_3 = h * FI.f_fi_7(m, x7, fi4 - k_fi4_2 / 2);
double k_fi8_3 = h * FI.f_fi_8(m, x8, fi7 - k_fi7_2 / 2);
double k_fi9_3 = h * FI.f_fi_9(m, x9, fi2 - k_fi2_2 / 2);
double k_fi10_3 = h * FI.f_fi_10(m, x12, fi9 - k_fi9_2 / 2);
double k_fi11_3 = h * FI.f_fi_11(m, x11);
double k_fi12_3 = h * FI.f_fi_12(m, x12, fi11 - k_fi11_2 / 2);
//
double k_fi1_4 = h * FI.f_fi_1();
double k_fi2_4 = h * FI.f_fi_2(m, x2, fi1 - k_fi1_3, fi2 - k_fi2_3);
double k_fi3_4 = h * FI.f_fi_3();
double k_fi4_4 = h * FI.f_fi_4(m, x4, fi3 - k_fi3_3, fi4 - k_fi4_3);
double k_fi5_4 = h * FI.f_fi_5();
double k_fi6_4 = h * FI.f_fi_6(m, x6, fi5 - k_fi5_3, fi6 - k_fi6_3);
double k_fi7_4 = h * FI.f_fi_7(m, x7, fi4 - k_fi4_3);
double k_fi8_4 = h * FI.f_fi_8(m, x8, fi7 - k_fi7_3);
double k_fi9_4 = h * FI.f_fi_9(m, x9, fi2 - k_fi2_3);
double k_fi10_4 = h * FI.f_fi_10(m, x12, fi9 - k_fi9_3);
double k_fi11_4 = h * FI.f_fi_11(m, x11);
double k_fi12_4 = h * FI.f_fi_12(m, x12, fi11 - k_fi11_3);

double answer = 0.0;
if (str.equals("fi1")) {
   answer = fi1 + (k_fi1_1 + 2 * k_fi1_2 + 2 * k_fi1_3 + k_fi1_4) / 6;
} else if (str.equals("fi2")) {
   answer = fi2 + (k_fi2_1 + 2 * k_fi2_2 + 2 * k_fi2_3 + k_fi2_4) / 6;
} else if (str.equals("fi3")) {
   answer = fi3 + (k_fi3_1 + 2 * k_fi3_2 + 2 * k_fi3_3 + k_fi3_4) / 6;
} else if (str.equals("fi4")) {
   answer = fi4 + (k_fi4_1 + 2 * k_fi4_2 + 2 * k_fi4_3 + k_fi4_4) / 6;
} else if (str.equals("fi5")) {
   answer = fi5 + (k_fi5_1 + 2 * k_fi5_2 + 2 * k_fi5_3 + k_fi5_4) / 6;
} else if (str.equals("fi6")) {
   answer = fi6 + (k_fi6_1 + 2 * k_fi6_2 + 2 * k_fi6_3 + k_fi6_4) / 6;
} else if (str.equals("fi7")) {
   answer = fi7 + (k_fi7_1 + 2 * k_fi7_2 + 2 * k_fi7_3 + k_fi7_4) / 6;
} else if (str.equals("fi8")) {
   answer = fi8 + (k_fi8_1 + 2 * k_fi8_2 + 2 * k_fi8_3 + k_fi8_4) / 6;
} else if (str.equals("fi9")) {
   answer = fi9 + (k_fi9_1 + 2 * k_fi9_2 + 2 * k_fi9_3 + k_fi9_4) / 6;
} else if (str.equals("fi10")) {
   answer = fi10 + (k_fi10_1 + 2 * k_fi10_2 + 2 * k_fi10_3 + k_fi10_4) / 6;
} else if (str.equals("fi11")) {
```

```
            answer = fi11 + (k_fi11_1 + 2 * k_fi11_2 + 2 * k_fi11_3 + k_fi11_4) / 6;
        } else if (str.equals("fi12")) {
            answer = fi12 + (k_fi12_1 + 2 * k_fi12_2 + 2 * k_fi12_3 + k_fi12_4) / 6;
        }
        return answer;
    }
}
class RungeKutta_forX {
    private final int N = Constants.N;
    private final double m;
    public final double[] t;
    public final double[] u1;
    public final double[] u2;
    public final double[] u3;
    public final double[] u4;
    public final double[] x1 = new double[N + 1];
    public final double[] x2 = new double[N + 1];
    public final double[] x3 = new double[N + 1];
    public final double[] x4 = new double[N + 1];
    public final double[] x5 = new double[N + 1];
    public final double[] x6 = new double[N + 1];
    public final double[] x7 = new double[N + 1];
    public final double[] x8 = new double[N + 1];
    public final double[] x9 = new double[N + 1];
    public final double[] x10 = new double[N + 1];
    public final double[] x11 = new double[N + 1];
    public final double[] x12 = new double[N + 1];

    public RungeKutta_forX(double m, double[] t, double[] u1, double[] u2, double[] u3, double[]
u4) {
        this.m = m;
        this.t = t;
        this.u1 = u1;
        this.u2 = u2;
        this.u3 = u3;
        this.u4 = u4;
        calculate();
    }
    private void calculate() {
        x1[0] = Constants.X_t0[0];
        x2[0] = Constants.X_t0[1];
        x3[0] = Constants.X_t0[2];
        x4[0] = Constants.X_t0[3];
        x5[0] = Constants.X_t0[4];
        x6[0] = Constants.X_t0[5];
        x7[0] = Constants.X_t0[6];
        x8[0] = Constants.X_t0[7];
        x9[0] = Constants.X_t0[8];
        x10[0] = Constants.X_t0[9];
        x11[0] = Constants.X_t0[10];
        x12[0] = Constants.X_t0[11];
        for (int i = 1; i < N; i++) {
```

```
            x1[i] = getRK("x1", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x2[i] = getRK("x2", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x3[i] = getRK("x3", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x4[i] = getRK("x4", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x5[i] = getRK("x5", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x6[i] = getRK("x6", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x7[i] = getRK("x7", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x8[i] = getRK("x8", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x9[i] = getRK("x9", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i], x7[i],
x8[i], x9[i], x10[i], x11[i], x12[i]);
            x10[i] = getRK("x10", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i],
x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            x11[i] = getRK("x11", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i],
x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
            x12[i] = getRK("x12", t[i], u1[i], u2[i], u3[i], u4[i], x1[i], x2[i], x3[i], x4[i], x5[i], x6[i],
x7[i], x8[i], x9[i], x10[i], x11[i], x12[i]);
        }
    }
    public double getRK(String str, double t, double u1, double u2, double u3, double u4, double x1,
double x2, double x3, double x4, double x5, double x6, double x7, double x8, double x9, double
x10, double x11, double x12) {
        X x = new X();
        double h = (Constants.T - Constants.t0) / Constants.N;
        double k_x1_1 = h * x.f_x1(x2);
        double k_x2_1 = h * x.f_x2(x9, x2);
        double k_x3_1 = h * x.f_x3(x4);
        double k_x4_1 = h * x.f_x4(x7, x4);
        double k_x5_1 = h * x.f_x5(x6);
        double k_x6_1 = h * x.f_x6(u1, u2, u3, u4, x6);
        double k_x7_1 = h * x.f_x7(x8);
        double k_x8_1 = h * x.f_x8(u4, u2);
        double k_x9_1 = h * x.f_x9(x10);
        double k_x10_1 = h * x.f_x10(u3, u1);
        double k_x11_1 = h * x.f_x11(x12);
        double k_x12_1 = h * x.f_x12(u1, u2, u3, u4);
        //
        double k_x1_2 = h * x.f_x1(x2 + k_x1_1 / 2);
        double k_x2_2 = h * x.f_x2(x9 + k_x9_1 / 2, x2 + k_x2_1 / 2);
        double k_x3_2 = h * x.f_x3(x4 + k_x4_1 / 2);
        double k_x4_2 = h * x.f_x4(x7 + k_x7_1 / 2, x4 + k_x4_1 / 2);
        double k_x5_2 = h * x.f_x5(x6 + k_x6_1 / 2);
        double k_x6_2 = h * x.f_x6(u1, u2, u3, u4, x6 + k_x6_1 / 2);
        double k_x7_2 = h * x.f_x7(x8 + k_x8_1 / 2);
        double k_x8_2 = h * x.f_x8(u4, u2);
```

```
double k_x9_2 = h * x.f_x9(x10 + k_x10_1 / 2);
double k_x10_2 = h * x.f_x10(u3, u1);
double k_x11_2 = h * x.f_x11(x12 + k_x12_1 / 2);
double k_x12_2 = h * x.f_x12(u1, u2, u3, u4);
//
double k_x1_3 = h * x.f_x1(x2 + k_x1_2 / 2);
double k_x2_3 = h * x.f_x2(x9 + k_x9_2 / 2, x2 + k_x2_2 / 2);
double k_x3_3 = h * x.f_x3(x4 + k_x4_2 / 2);
double k_x4_3 = h * x.f_x4(x7 + k_x7_2 / 2, x4 + k_x4_2 / 2);
double k_x5_3 = h * x.f_x5(x6 + k_x6_2 / 2);
double k_x6_3 = h * x.f_x6(u1, u2, u3, u4, x6 + k_x6_2 / 2);
double k_x7_3 = h * x.f_x7(x8 + k_x8_2 / 2);
double k_x8_3 = h * x.f_x8(u4, u2);
double k_x9_3 = h * x.f_x9(x10 + k_x10_2 / 2);
double k_x10_3 = h * x.f_x10(u3, u1);
double k_x11_3 = h * x.f_x11(x12 + k_x12_2 / 2);
double k_x12_3 = h * x.f_x12(u1, u2, u3, u4);
//
double k_x1_4 = h * x.f_x1(x2 + k_x1_3);
double k_x2_4 = h * x.f_x2(x9 + k_x9_3, x2 + k_x2_3);
double k_x3_4 = h * x.f_x3(x4 + k_x4_3);
double k_x4_4 = h * x.f_x4(x7 + k_x7_3, x4 + k_x4_3);
double k_x5_4 = h * x.f_x5(x6 + k_x6_3);
double k_x6_4 = h * x.f_x6(u1, u2, u3, u4, x6 + k_x6_3);
double k_x7_4 = h * x.f_x7(x8 + k_x8_3);
double k_x8_4 = h * x.f_x8(u4, u2);
double k_x9_4 = h * x.f_x9(x10 + k_x10_3);
double k_x10_4 = h * x.f_x10(u3, u1);
double k_x11_4 = h * x.f_x11(x12 + k_x12_3);
double k_x12_4 = h * x.f_x12(u1, u2, u3, u4);
double answer = 0.0;
if (str.equals("x1")) {
   answer = x1 + (k_x1_1 + 2 * k_x1_2 + 2 * k_x1_3 + k_x1_4) / 6;
} else if (str.equals("x2")) {
   answer = x2 + (k_x2_1 + 2 * k_x2_2 + 2 * k_x2_3 + k_x2_4) / 6;
} else if (str.equals("x3")) {
   answer = x3 + (k_x3_1 + 2 * k_x3_2 + 2 * k_x3_3 + k_x3_4) / 6;
} else if (str.equals("x4")) {
   answer = x4 + (k_x4_1 + 2 * k_x4_2 + 2 * k_x4_3 + k_x4_4) / 6;
} else if (str.equals("x5")) {
   answer = x5 + (k_x5_1 + 2 * k_x5_2 + 2 * k_x5_3 + k_x5_4) / 6;
} else if (str.equals("x6")) {
   answer = x6 + (k_x6_1 + 2 * k_x6_2 + 2 * k_x6_3 + k_x6_4) / 6;
} else if (str.equals("x7")) {
   answer = x7 + (k_x7_1 + 2 * k_x7_2 + 2 * k_x7_3 + k_x7_4) / 6;
} else if (str.equals("x8")) {
   answer = x8 + (k_x8_1 + 2 * k_x8_2 + 2 * k_x8_3 + k_x8_4) / 6;
} else if (str.equals("x9")) {
   answer = x9 + (k_x9_1 + 2 * k_x9_2 + 2 * k_x9_3 + k_x9_4) / 6;
} else if (str.equals("x10")) {
   answer = x10 + (k_x10_1 + 2 * k_x10_2 + 2 * k_x10_3 + k_x10_4) / 6;
} else if (str.equals("x11")) {
```

```
            answer = x11 + (k_x11_1 + 2 * k_x11_2 + 2 * k_x11_3 + k_x11_4) / 6;
        } else if (str.equals("x12")) {
            answer = x12 + (k_x12_1 + 2 * k_x12_2 + 2 * k_x12_3 + k_x12_4) / 6;
        }
        return answer;
    }
}
class Jk {
    private final double[] gradient1;
    private final double[] gradient2;
    private final double[] gradient3;
    private final double[] gradient4;
    private final double[] uk1;
    private final double[] uk2;
    private final double[] uk3;
    private final double[] uk4;
    private final double[] _u1;
    private final double[] _u2;
    private final double[] _u3;
    private final double[] _u4;
    private final double[] t;
    private double result;

    public Jk(double[] gradient1, double[] gradient2, double[] gradient3, double[] gradient4, double[]
uk1, double[] uk2, double[] uk3, double[] uk4, double[] _u1, double[] _u2, double[] _u3, double[]
_u4, double[] t) {
        this.gradient1 = gradient1;
        this.gradient2 = gradient2;
        this.gradient3 = gradient3;
        this.gradient4 = gradient4;
        this.uk1 = uk1;
        this.uk2 = uk2;
        this.uk3 = uk3;
        this.uk4 = uk4;
        this._u1 = _u1;
        this._u2 = _u2;
        this._u3 = _u3;
        this._u4 = _u4;
        this.t = t;
        rightRectangles();
    }
    public double getResult() {
        return result;
    }
    private void rightRectangles() {
        double[] temp = new double[4];
        for (int i = 1; i < Constants.N; i++) {
            temp[0] = gradient1[i - 1] * (_u1[i - 1] - uk1[i - 1]);
            temp[1] = gradient2[i - 1] * (_u2[i - 1] - uk2[i - 1]);
            temp[2] = gradient3[i - 1] * (_u3[i - 1] - uk3[i - 1]);
            temp[3] = gradient4[i - 1] * (_u4[i - 1] - uk4[i - 1]);
            result += temp[0] * (t[i] - t[i - 1]);
```

```
            result += temp[1] * (t[i] - t[i - 1]);
            result += temp[2] * (t[i] - t[i - 1]);
            result += temp[3] * (t[i] - t[i - 1]);
        }
    }
}
class Norm {
    public Norm() {
    }
    public double norma(double[] grad1, double[] grad2, double[] grad3, double[] grad4, double h) {
        double sum = 0.0;
        for (int i = 1; i < Constants.N; i++) {
            sum += Math.pow(grad1[i], 2) * h;
            sum += Math.pow(grad2[i], 2) * h;
            sum += Math.pow(grad3[i], 2) * h;
            sum += Math.pow(grad4[i], 2) * h;
        }
        return Math.sqrt(sum);
    }
}
```