FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Transport břemene pomocí bezpilotního letounu

Lukáš Kozel

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA
KYBERNETIKY

Bakalářská práce

Transport břemene pomocí bezpilotního letounu

Lukáš Kozel

Vedoucí práce
Ing. Zdeněk Bouček

PLZEŇ 2024

Citace v seznamu literatury:
KOZEL, Lukáš. *Transport břemene pomocí bezpilotního letounu*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra kybernetiky. Vedoucí práce Ing. Zdeněk Bouček.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš KOZEL**
Osobní číslo:	**A21B0384P**
Studijní program:	**B0714A150005 Kybernetika a řídicí technika**
Specializace:	**Automatické řízení a robotika**
Téma práce:	**Transport břemene pomocí bezpilotního letounu**
Zadávající katedra:	**Katedra kybernetiky**

Zásady pro vypracování

1. Seznámení se s chováním soustavy kvadrotorová helikoptéra-břemeno.
2. Vytvoření modelu chování soustavy.
3. Návrh řídicích algoritmů pro transport a stabilizaci nákladu.
4. Otestování algoritmů v simulačním prostředí.

Rozsah bakalářské práce: **30-40 stránek A4**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

- Kotaru, P., Wu, G., & Sreenath, K. (2017). Dynamics and control of a quadrotor with a payload suspended through an elastic cable. 2017 American Control Conference (ACC), 3906–3913. https://doi.org/10.23919/ACC.2017.7963553
- Saunders, J., Saeedi, S. & Li, W. (2023) Autonomous aerial robotics for package delivery: a technical review. Journal of Field Robotics, 1–47. https://doi.org/10.1002/rob.22231
- Villa, D. K. D., Brandão, A. S., & Sarcinelli-Filho, M. (2020). A Survey on Load Transportation Using Multirotor UAVs. Journal of Intelligent and Robotic Systems: Theory and Applications, 98(2), 267–296. https://doi.org/10.1007/s10846-019-01088-w

Vedoucí bakalářské práce: **Ing. Zdeněk Bouček**
Výzkumný program 1

Datum zadání bakalářské práce: **17. října 2023**
Termín odevzdání bakalářské práce: **20. května 2024**

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

V Plzni dne 17. října 2023

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 20. května 2024

. . . . . .
Lukáš Kozel

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Tato bakalářská práce se soustředí na vývoj a implementaci algoritmu pro řízení dronu s nákladem zavěšeným volně pod dronem s cílem efektivně tlumit oscilace nákladu během letu. Nejdříve je odvozen matematický model řízeného systému, který je klíčový pro následný návrh LQ (lineárně kvadratického) regulátoru. Následuje návrh a implementace řídícího algoritmu, který je zásadní pro aktivní tlumení oscilací nákladu. Algoritmus je poté nasazen na simulační model. Algoritmus je testován v prostředí PX4 Autopilot v software in the loop, což umožňuje důkladné ověření funkčnosti před potenciálním nasazením na reálný systém. Dále byl pro propojení řídícího systému s virtuálním soupravou dron s nákladem využit ROS2, což zajišťuje efektivní komunikaci. Závěrečná část práce se věnuje validaci implementovaného algoritmu prostřednictvím testovacích scénářů s cílem ověřit jeho schopnost efektivně řídit dron a aktivně tlumit oscilace nákladu za různých podmínek.

Abstract

This bachelor's thesis focuses on the development and implementation of a control algorithm for a drone with suspended load, aimed at effectively reducing cargo oscillations during flight. First of all, a mathematical model of the controlled system is derived, which is crucial for the development of the LQ (linear quadratic) regulator. This is followed by the development and implementation of the control algorithm. It focuses on the active damping of cargo oscillations. The control algorithm is then deployed on a simulation model and tested in the PX4 Autopilot environment using software in the loop. This approach enables verification of functionality for detailed testing before potential deployment on a real-world system. ROS2 is used to connect the control system with the virtual assembly of drone and load, which leads to ensuring efficient communication. The final part of this thesis focuses on validating the implementation of the control algorithm. It is done by testing different scenarios to verify its ability to efficiently control the drone and actively damp cargo oscillations under specific conditions.

Klíčová slova

bezpilotní letoun • dron • transport nákladu • LQR • ROS2 • PX4 • Gazebo

Poděkování

Tímto bych chtěl poděkovat především mému vedoucímu bakalářské práce, panu Ing. Zdeňku Boučkovi, za možnost zabývat se tímto tématem a za jeho vstřícný přístup, trpělivost a poskytnutí cenných studijních materiálů, které byly klíčové pro mé pochopení problematiky. Velká vděčnost patří také mým rodičům, kteří mě podporovali během celého mého studia.

Lukáš Kozel,
autor
(květen 2024)

Obsah

Úvod 1

V dnešní době se bezpilotní letouny (UAV – Unmanned Aerial Vehicles), známé také jako drony, stávají stále více populárními v různých oblastech. Jejich aplikace sahají od monitorování prostředí, přes inspekce infrastruktury, až po zábavu. Tato práce se zaměřuje na konkrétní aplikaci dronů v oblasti transportu nákladu, což je oblast s rostoucím potenciálem a zájmem nejen ve výzkumu, ale i v průmyslovém využití. Jedním z příkladů je využití dronů pro doručování zásilek, což bylo podrobně prozkoumáno v článku [SSL24].

Dron je sám o sobě charakterizován jako podaktuovaný systém. To, že je systém podaktuovaný znamená, že systém obsahuje více stupňů volnosti, než je počet dostupných aktuátorů. U nejčastěji používané konstrukce dronu, tedy kvadrokoptéry, se jedná o šest stupňů volnosti, ale pouze o čtyři aktuátory (rotory).

Problematika samotného transportu nákladu se zabývá především způsoby upevnění nákladu k dronům, které mají zásadní vliv na manévrovatelnost a dynamiku systému. Mezi nejpoužívanější přístupy ovšem patří připevnění nákladu přímo do konstrukce dronu či zavěšení pod dron na volném závěsu. Každý z těchto přístupů je vhodnější pro různé specifické aplikace a nese s sebou odlišné výzvy. Oba tyto přístupy jsou diskutovány v článku [VBS20].

Co se týče případu, kdy je náklad zavěšený na volném závěsu pod dronem, tak autor především zdůrazňuje, že tento přístup velmi významně mění dynamiku systému, a to právě přidáním pasivních stupňů volnosti. Řídící algoritmy musí tedy nejen dokázat stabilizovat samotný dron, ale také počítat s oscilacemi nákladu, které je nutné správně navrženým řídícím algoritmem kompenzovat.

Na druhou stranu, pokud by náklad byl upevněn přímo v konstrukci dronu, tak dojde ke zvýšení setrvačných momentů, což negativně ovlivňuje schopnost dronu reagovat na řídící signály s požadovanou rychlostí a přesností, zejména v kontextu rychlých manévrů. Tento přístup zjednodušuje návrh řídícího algoritmu. Ovšem, aby byl dron schopen dosáhnout jisté úrovně manévrovatelnosti, je potřeba generovat více energie, což může být mnohdy velmi omezující. Omezení to může být zejména v případě, kdy je dron vybaven bateriemi s omezenou kapacitou, což významně snižuje maximální dobu provozu.

Tato práce se tedy zabývá návrhem řídícího systému pro soustavu dronu s volně zavěšeným nákladem na závěsu. V kapitole 2 je detailně prozkoumána problematika sestavení samotného matematického modelu soustavy, na které je poté postaveno matematické odvození modelu. Pro získaný matematický model je v kapitole 3 navržen regulátor, jenž je před dalším použitím důkladně validován. Následuje kapitola 4, která je věnována detailnímu představení použitých technologií v rámci implementace řídícího algoritmu a simulace. Je zde také detailně rozebrán simulační model a vytvořené uživatelské rozhraní. Samotný řídící algoritmus a jeho implementace je detailně popsán v kapitole 5. V závěru této práce, tedy v kapitole 6, je výsledný řídící algoritmus testován na experimentálních scénářích za účelem vyhodnocení dosažené kvality řízení.

Veškeré zdrojové kódy jsou k dispozici na odkazu (https://github.com/Lukas-Kozel/px4_drone_control_bachelors_thesis)

Návrh modelu dronu s nákladem — 2

Tato kapitola se zabývá problematikou odvození matematického modelu dronu se zavěšeným nákladem a jeho následným matematickým popisem. Nejprve je detailně popsána problematika při sestavování zjednodušeného matematického modelu, jenž je zaměřena především na modelování samotného závěsu. Následuje analýza omezení, se kterými se model potýká, což je klíčové pro pochopení jeho aplikačních limit. Na základě této analýzy je sestaven nelineární matematický model pomocí Lagrangeovy metody. Dále, pro účely návrhu regulátoru, je tento model linearizován v předem zvoleném pracovním bodě. Vzhledem k tomu, že reálné měření systémových veličin pomocí senzorů probíhá v diskrétním čase, tak je také provedena diskretizace modelu.

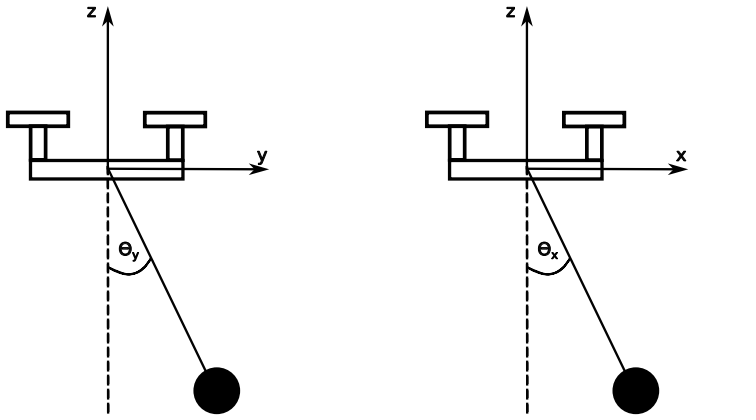
2.1 Problematika sestavení modelu

Modelovaný reálný systém se skládá z dronu a nákladu zavěšeného na závěsu pod dronem. V literatuře jsou uvedeny dva hlavní přístupy k modelování samotného závěsu [Est+24]. První přístup považuje závěs za nehmotný a pevný, přičemž náklad je modelován jako hmotný bod. Ačkoliv, na první pohled se tento přístup zdá až příliš zjednodušený, je schopen dostatečně zachytit dynamiku zavěšeného nákladu, proto byl vybrán pro tuto práci. Druhý přístup modeluje závěs jako sérii konečného počtu spojených hmotných tyčí, což poskytuje detailnější popis dynamiky systému, ale zároveň značně zvyšuje složitost modelu.

Pro návrh řízení je nutné znát co nejpřesněji matematický model soustavy. Výzva spočívá v odvození popisu takového modelu systému, který je komplikovaný především kvůli jeho nelineární povaze. Při sestavení matematického modelu, aby byl schopen co nejlépe napodobovat reálný systém, je vycházeno z několika omezeních a předpokladů [Est+24]. Patří sem předpoklad, že dron je modelován jako symetrické tuhé těleso. Závěs mezi nákladem a dronem je reprezentován nehmotnou tyčí, která je připojena do těžiště těla dronu a náklad je hmotný bod pevně spojený se závěsem. Za předpokladu, že samotný dron je symetrický v jeho osách,

tak lze komplexní prostorový model zjednodušit na použití identického planárního modelu dronu s nákladem pro osy x a y nezávisle, za zachováním dynamiky celého systému. Tento přístup je znázorněn na obrázku 2.1. Toto zjednodušení by ovšem bylo možné i pro případ, že by samotný dron symetrický nebyl, jelikož v uvažovaném modelu jsou zohledněny pouze hmotnosti a délka závěsu.

Tento přístup k modelování má samozřejmě své limitace. Jedna z nich je zmíněna v práci [KFJ17], kde je uvažován stejný princip zavěšení nákladu, jako je použit v této práci. Autor tvrdí, že jako závěs lze uvažovat tuhou nehmotnou tyč pouze v případě, že nedochází k agresivním manévrům a závěs zůstane napnutý. Tatáž informace je zmíněna v [Est+24], kde autor říká, že použití tuhé nehmotné tyče reprezentuje většinu dynamiky volného závěsu dostatečně podrobně. Problém nastává při agresivních manévrech, kdy se projevuje nelinearita volného závěsu.



(a) Model řízeného systému pro osu y

(b) Model řízeného systému pro osu x

Obrázek 2.1: Modely řízeného systému

2.2 Matematické odvození modelu

Uvažovaný planární model, pro který bude odvozen matematický model, je ilustrován na obr. 2.1. Pro odvození matematického modelu je využito Lagrangeovy metody [Sch16]. Tato metoda byla zvolena především kvůli jednoduché aplikaci na tento popisovaný systém. Alternativní přístupy, jako je aplikace Newton-Eulerových metod pro stanovení dynamického rovnovážného stavu všech těles v systému, by představovaly významnou komplexitu při aplikaci na tento specifický systém.

Princip Lagrangeovy metody bude demonstrován na příkladu v rámci odvození modelu. Použití této metody je založeno na znalosti potenciální a kinetické energie celé soustavy. Jelikož je uvažován pouze planární model, tak bude uvažován pohyb

pouze ve směru jedné osy, konkrétně osy x. Pro odvození energie soustavy lze úlohu rozdělit na samostatný popis dronu a samostatný popis nákladu, a poté je sloučit.

2.2.1 Energie dronu

Za předpokladu, že dron se pohybuje pouze v ose x a potenciální energie bude vyjádřena vůči počátku souřadného systému, jak bylo ilustrováno na obrázku 2.1, pak je potenciální energie nulová. Tedy platí, že

$$V_{dron} = 0. \tag{2.1}$$

Kinetická energie dronu je tedy dána jako

$$T_{dron} = \frac{1}{2} \cdot M \cdot \dot{x}(t)^2, \tag{2.2}$$

kde M je hmotnost samotného dronu a $\dot{x}(t)$ je rychlost dronu v ose x.

2.2.2 Energie nákladu

Náklad v tomto zjednodušeném modelu vykazuje stejné chování jako kyvadlo. Potenciální energie vztažená k počátku souřadného systému je tedy dána jako

$$V_{\text{náklad}} = -m \cdot g \cdot l \cdot cos(\theta(t)), \tag{2.3}$$

kde m je hmotnost nákladu, g je tíhové zrychlení, l je délka závěsu a $\theta(t)$ je výchylka nákladu z rovnovážné polohy.

Kinetická energie nákladu je ovšem aktivně ovlivňována pohybem dronu. Je tedy nutné počítat i se samotnou rychlostí dronu a jejím vlivem na kinetickou energii nákladu. Kinetickou energii lze tedy zapsat vztahem

$$T_{\text{náklad}} = \frac{1}{2} \cdot m \cdot (\dot{v}_x^2(t) + \dot{v}_z^2(t)), \tag{2.4}$$

kde $\dot{v}_x$ je rychlost nákladu ve směru osy x a $\dot{v}_z(t)$ je rychlost nákladu ve směru osy z. Tyto jednotlivé složky jsou dány jako

$$\dot{v}_x(t) = -l \cdot cos(\theta(t)) \cdot \dot{\theta}(t) + \dot{x}(t), \tag{2.5}$$

$$\dot{v}_z(t) = l \cdot sin(\theta(t)) \cdot \dot{\theta}(t). \tag{2.6}$$

2.2.3 Stanovení pohybových rovnic

Pro stanovení pohybových rovnic pomocí Lagrangeovy metody je třeba určit celkovou potenciální energii soustavy $V_{soustavy}$ a celkovou kinetickou energii soustavy $T_{soustavy}$. Ty jsou dány jako

$$V_{soustavy} = V_{dron} + V_{\text{náklad}} = -m \cdot g \cdot l \cdot cos(\theta(t)), \tag{2.7}$$

$$T_{soustavy} = T_{dron} + T_{\text{náklad}} = \frac{1}{2} \cdot M \cdot \dot{x}(t)^2 + \frac{1}{2} \cdot m \cdot (\dot{x}(t) - \dot{\theta}(t) \cdot l \cdot cos(\theta(t)))^2 + \\ + \dot{\theta}(t)^2 \cdot l^2 \cdot sin(\theta(t))^2. \tag{2.8}$$

Lagrangeova funkce L je definována jako rozdíl kinetické a potenciální energie soustavy, tedy

$$L = T_{soustavy} - V_{soustavy} = \frac{1}{2} \cdot M \cdot \dot{x}(t)^2 + \frac{1}{2} \cdot m \cdot (\dot{x}(t) - \dot{\theta}(t) \cdot l \cdot \cos(\theta(t)))^2 + \\ + \dot{\theta}(t)^2 \cdot l^2 \cdot \sin(\theta(t))^2 + m \cdot g \cdot l \cdot \cos(\theta(t)). \tag{2.9}$$

Výsledné pohybové rovnice jsou získány pomocí Lagrangeovy rovnice, jejíž předpis je

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\boldsymbol{q}}}\right) - \frac{\partial L}{\partial \boldsymbol{q}} = Q, \tag{2.10}$$

kde $\boldsymbol{q}$ jsou zobecněné souřadnice, nebo-li stupně volnosti. Planární model dronu s nákladem má ze své podstaty právě dva stupně volnosti, které lze zapsat maticově jako

$$\boldsymbol{q} = \begin{bmatrix} x & \theta \end{bmatrix}^T, \tag{2.11}$$

kde x je poloha dronu a θ je úhel vychýlení nákladu od osy z, jak již bylo znázorněno na obrázku 2.1.

Jelikož má systém dva stupně volnosti, tak musí být popsán právě dvěma pohybovými rovnicemi. Ty lze získat dosazením Lagrangeovy funkce (2.9) a zobecněných souřadnic do Lagrangeovy rovnice (2.10). Po dosazení a zderivování mají pohybové rovnice tvar

$$l \cdot m \cdot sin(\theta(t)) \cdot \dot{\theta}(t)^2 + \ddot{x}(t) \cdot (M + m) - \ddot{\theta}(t) \cdot l \cdot m \cdot cos(\theta(t)) = Q_1, \tag{2.12}$$

$$l \cdot m \cdot (\ddot{\theta}(t) \cdot l - \ddot{x}(t) \cdot cos(\theta(t)) + g \cdot sin(\theta(t))) = Q_2. \tag{2.13}$$

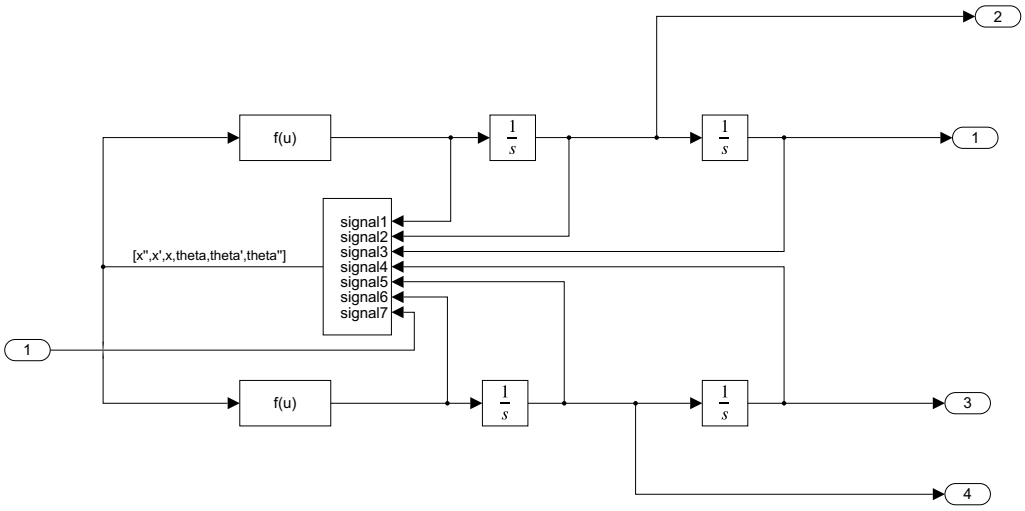
Platí, že Q_1 je síla působící na dron, jež lze interpretovat jako průmět tahu dronu do osy x. Tato veličina bude uvažována jako vstup do systému.

Pro Q_2 platí, že se jedná o sílu, která působí na náklad. Ta bude ovšem zanedbána. Pohybové rovnice lze upravit vyjádřením nejvyšších řádů derivací, což bude později užitečné při linearizaci. Výsledný tvar pohybových rovnic je

$$\ddot{x}(t) = \frac{-l \cdot m \cdot \sin(\theta(t)) \cdot \dot{\theta}(t)^2 + Q_1 - g \cdot m \cdot \cos(\theta(t)) \cdot \sin(\theta(t))}{-m \cdot \cos(\theta(t))^2 + M + m}, \tag{2.14}$$

$$\ddot{\theta}(t) = \frac{-l \cdot m \cdot \cos(\theta(t)) \cdot \sin(\theta(t)) \cdot \dot{\theta}(t)^2 + Q_1 \cdot \cos(\theta(t))}{l \cdot (-m \cdot \cos(\theta(t))^2 + M + m)} + \\ + \frac{-g \cdot m \cdot \sin(\theta(t)) - M \cdot g \cdot \sin(\theta(t))}{l \cdot (-m \cdot \cos(\theta(t))^2 + M + m)}. \tag{2.15}$$

V Simulinku byl vytvořen model tohoto nelineárního systému na základě rovnic (2.14) a (2.15). Schéma je ilustrováno na obrázku 2.2.



Obrázek 2.2: Schéma nelineárního modelu systému v Simulinku

2.3 Linearizace

Z pohybových rovnic (2.12) a (2.13) je zřejmé, že popisovaný systém je silně nelineární. Pro návrh řízení to znamená, že je systém potřeba linearizovat. Linearizace je aproximací nelineárního systému v blízkosti daného pracovního bodu. Tento bod se nazývá rovnovážný bod, pokud platí, že jsou derivace rovny nule a řízení je konstantní či nulové. Pro linearizaci je nutné si zvolit stavové proměnné a sestavit soustavu rovnic. Stavy a vstup do systému byly zvoleny jako

$$\boldsymbol{x} = \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \theta(t) \\ \dot{\theta}(t) \end{bmatrix}, u(t) = Q_1. \tag{2.16}$$

Derivací jednotlivých stavů a dosazením rovnic (2.14) a (2.15) lze získat soustavu nelineárních rovnic popisující daný systém pomocí stavových proměnných. Počet těchto rovnic je dán počtem stavů. Tyto rovnice jsou popsány funkcemi, jenž popisují derivaci příslušného stavu. Jsou dány předpisem $f_i(\boldsymbol{x}(t), u(t))$, kde $i \in \{1, 2, 3, 4\}$.

Konkrétní tvary těchto funkcí jsou

$$\dot{x}_1(t) = x_2(t) = f_1(\boldsymbol{x}(t), u(t)), \tag{2.17}$$

$$\dot{x}_2(t) = \frac{-1 \cdot m \cdot \sin(x_3(t)) \cdot x_4^2(t) + u(t)}{-m \cdot \cos(x_3(t))^2 + M + m} + \frac{-m \cdot g \cdot \cos(x_3(t)) \cdot \sin(x_3(t))}{-m \cdot \cos(x_3(t))^2 + M + m} = f_2(\boldsymbol{x}(t), u(t)), \tag{2.18}$$

$$\dot{x}_3(t) = x_4(t) = f_3(\boldsymbol{x}(t), u(t)), \tag{2.19}$$

$$\dot{x}_4(t) = \frac{-l \cdot m \cdot \cos(x_3(t)) \cdot \sin(x_3(t)) \cdot x_4^2(t) + u(t) \cdot \cos(x_3(t))}{l \cdot (-m \cdot \cos(x_3(t))^2 + M + m)} + \frac{-g \cdot m \cdot \sin(x_3(t)) - M \cdot g \cdot \sin(x_3(t))}{l \cdot (-m \cdot \cos(x_3(t))^2 + M + m)} = f_4(\boldsymbol{x}(t), u(t)). \tag{2.20}$$

Linearizace dynamického systému umožňuje transformaci nelineárního modelu na jeho lineární aproximaci v blízkosti vybraného pracovního bodu $\boldsymbol{x_r}$. Tento přístup zahrnuje přechod od původního stavového vektoru $\boldsymbol{x}$, který zahrnuje všechny stavové proměnné, k nově definovanému vektoru odchylek $\Delta\boldsymbol{x}$. Vektor $\Delta\boldsymbol{x}$ zde reprezentuje rozdíly mezi aktuálním stavem systému a jeho rovnovážným stavem. Analogicky, pro vstupy systému se transformuje u na Δu, což jsou odchylky od rovnovážného stavu vstupu.
Pro pracovní bod, okolo kterého bude provedena linearizace platí, že

$$\boldsymbol{x_r} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T, \tag{2.21}$$

$$u_r = 0. \tag{2.22}$$

Dosazením do obecného předpisu pro matice stavového popisu pro linearizovaný systém lze získat kompletní stavový popis linearizovaného systému. Samotné matice $\boldsymbol{A}$ a $\boldsymbol{B}$ ve stavové rovnici jsou dány jako

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} & \frac{\partial f_1}{\partial x_4} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} & \frac{\partial f_2}{\partial x_4} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} & \frac{\partial f_3}{\partial x_4} \\ \frac{\partial f_4}{\partial x_1} & \frac{\partial f_4}{\partial x_2} & \frac{\partial f_4}{\partial x_3} & \frac{\partial f_4}{\partial x_4} \end{bmatrix}\Bigg|_{\boldsymbol{x}=\boldsymbol{x_r}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{-g \cdot m}{M} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{M \cdot g + g \cdot m}{M \cdot l} & 0 \end{bmatrix}, \tag{2.23}$$

$$\boldsymbol{B} = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \\ \frac{\partial f_3}{\partial u} \\ \frac{\partial f_4}{\partial u} \end{bmatrix}\Bigg|_{u=u_r} = \begin{bmatrix} 0 & \frac{1}{M} & 0 & \frac{1}{M \cdot l} \end{bmatrix}^T. \tag{2.24}$$

Výstup systému je uvažován jako každý ze stavů. Matice $\boldsymbol{C}$ a $\boldsymbol{D}$ ve výstupní rovnici mají tedy tvar

$$\boldsymbol{C} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.25}$$

$$\boldsymbol{D} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix}^T. \tag{2.26}$$

S využitím matic $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, a $\boldsymbol{D}$, které byly odvozeny během procesu linearizace, lze efektivně popsat, jaký vliv mají malé odchylky $\boldsymbol{\Delta x}$ a změny v řízení Δu na dynamiku systému. Tento vliv je formulován skrze stavové a výstupní rovnice, které modelují chování systému v okolí pracovního bodu:

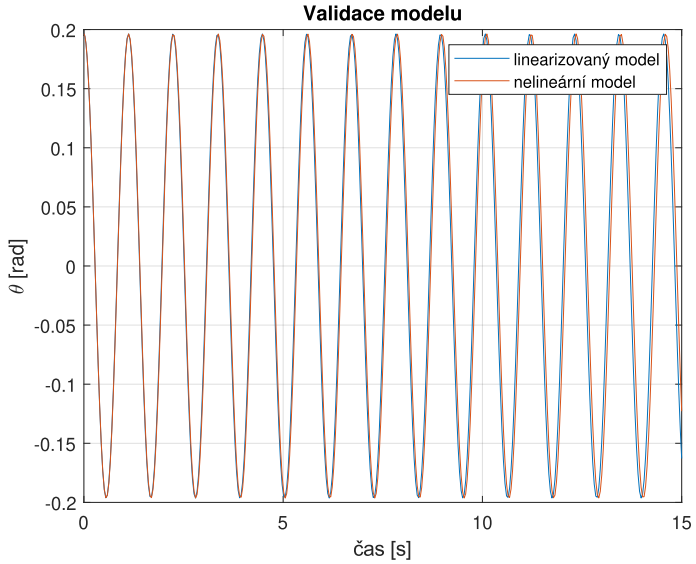
$$\begin{aligned} \Delta\dot{\mathbf{x}}(\mathrm{t}) &= \mathbf{A}\Delta\mathbf{x}(\mathrm{t}) + \mathbf{B}\Delta u(\mathrm{t}), \\ \Delta\mathbf{y}(\mathrm{t}) &= \mathbf{C}\Delta\mathbf{x}(\mathrm{t}) + \mathbf{D}\Delta u(\mathrm{t}). \end{aligned} \tag{2.27}$$

Dosazením lze získat úplnou stavovou reprezentaci linearizovaného systému, vyjádřenou stavovou a výstupní rovnicí. Uvažované parametry soustavy jsou gravitační zrychlení $g = 9{,}81\ \mathrm{m \cdot s^{-2}}$, délka závěsu $l = 0{,}35$ m, hmotnost nákladu $m = 0{,}25$ kg a hmotnost dronu $M = 2$ kg. V této práci jsou zmíněné parametry považovány za konstantní a zůstanou nezměněné pro všechny následující analýzy a výpočty. Stavová a výstupní rovnice mají tedy tvar

$$\begin{aligned} \Delta\dot{\mathbf{x}}(\mathrm{t}) &= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -1{,}23 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -31{,}53 & 0 \end{bmatrix} \Delta\mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ 0{,}5 \\ 0 \\ 1{,}43 \end{bmatrix} \Delta u(\mathrm{t}), \\ \Delta\mathbf{y}(\mathrm{t}) &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \Delta\mathbf{x}(\mathrm{t}) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta u(\mathrm{t}). \end{aligned} \tag{2.28}$$

2.4 Validace modelu

Validaci linearizovaného modelu, tedy systému popsaného stavovým popisem (2.28), lze provést tak, že dojde ke spuštění systému z nenulových počátečních podmínek a graficky lze porovnat chování obou modelů. Počáteční podmínky jsou zvoleny jako nulové, kromě počáteční výchylky nákladu $\theta(0) = \frac{\pi}{16}$. Tato hodnota je zvolena z toho důvodu, že z definice linearizovaného systému platí, že čím blíže se stavy systému přibližují pracovnímu bodu, tím přesnější linearizace je. Z obrázku 2.3 je vidět, že dynamika linearizovaného modelu a nelineárního modelu je velmi podobná a liší se pouze ve frekvenci. Tato odchylka je dána právě již zmíněnou vlastností linearizace.



Obrázek 2.3: Porovnání linearizovaného a nelineárního modelu

2.5 Diskretizace systému

Jelikož při nasazení regulátoru či přímo řídícího algoritmu jsou všechna data a měření diskrétní, tak i regulátor musí být navrhován na diskrétní systém. Diskretizace je realizována pomocí metody založené na tvarovači nultého řádu s periodou vzorkování $T_s = 0,033\ s$ a tedy frekvencí 30 Hz. Volba této periody je dána frekvencí, s jakou přicházejí data na simulačním modelu, což je detailněji prostudováno v sekci 5.1. Tato metoda funguje tak, že se za spojitý model systému zapojí tvarovač, který přidržuje konstantní hodnotu vstupu mezi okamžiky vzorkování.

Pro stavový popis systému to znamená, že je potřeba jej správně přepočítat na diskrétní ekvivalent. Výpočet proběhl tedy právě pomocí metody založené na tvarovači nultého řádu. Diskrétní stavový popis má obecný tvar

$$\begin{aligned} \boldsymbol{\Delta x}(k+1) &= \mathbf{A_D}\boldsymbol{\Delta x}(k) + \mathbf{B_D}\Delta u(k), \\ \boldsymbol{\Delta y}(k) &= \mathbf{C_D}\boldsymbol{\Delta x}(k) + \mathbf{D_D}\Delta u(k). \end{aligned} \tag{2.29}$$

Po vyčíslení lze diskrétní stavový popis soustavy popsat jako

$$\boldsymbol{\Delta x}(k+1) = \begin{bmatrix} 1 & 0{,}033 & 0{,}0007 & 0 \\ 0 & 1 & -0{,}04 & 0{,}00007 \\ 0 & 0 & 0{,}98 & 0{,}033 \\ 0 & 0 & -1{,}035 & 0{,}99 \end{bmatrix} \boldsymbol{\Delta x}(k) + \begin{bmatrix} 0{,}0003 \\ 0{,}02 \\ 0{,}0008 \\ 0{,}047 \end{bmatrix} \Delta u(k),$$

$$\boldsymbol{\Delta y}(k) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{\Delta x}(k) + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Delta u(k). \tag{2.30}$$

Návrh řízení soustavy 3

Tato kapitola se zabývá návrhem regulátoru pro linearizovaný model, jenž byl odvozen v předchozí kapitole. Cílem regulace je co nejrychleji snížit vychýlení nákladu a úspěšně tlumit veškeré kmity. Důraz je kladen především na přesnost a rychlost regulace. Pro tyto účely byl zvolen LQ regulátor, jelikož umožňuje velmi komplexní řízení jednotlivých stavů současně [GM19]. Také je velmi výhodný díky znalosti modelu, protože se jedná o regulátor, jehož návrh je prováděn přímo na základě modelu. Po návrhu bude regulátor validován na nelineárním modelu.

3.1 LQ regulátor

Jedná se o zpětnovazební lineární stavový regulátor, jehož řízení je realizováno prostřednictvím záporné zpětné vazby a matice zesílení $\boldsymbol{K}$. Řízení má tvar

$$\Delta u(k) = -\boldsymbol{K}\Delta\boldsymbol{x}(k), \tag{3.1}$$

kde $\Delta u(k)$ je vstup do systému, $\boldsymbol{K}$ je matice zesílení a $\Delta\boldsymbol{x}(k)$ je stavový vektor linearizovaného systému. U běžného lineárního zpětnovazebního stavového regulátoru lze matici $\boldsymbol{K}$ určit například přiřazením pólů či jinými technikami.

Výpočet matice zpětnovazebního zesílení u LQ regulátoru je založený na nalezení optimálního řízení, které minimalizuje kvadratickou ztrátovou funkci, jejíž tvar pro diskrétní variantu a konkrétní uvažovaný linearizovaný systém je

$$J = \sum_{k=0}^{\infty} \left(\Delta\mathbf{x}(k)^T \mathbf{Q} \Delta\mathbf{x}(k) + \Delta u(k)^T \mathbf{R} \Delta u(k) \right), \tag{3.2}$$

kde matice $\boldsymbol{Q}$ a $\boldsymbol{R}$ jsou návrhové parametry, které obecně reprezentují kompromis mezi kvalitou řízení a cenou či spotřebou. Obě tyto matice jsou pozitivně definitní. Praktické je volit matici $\boldsymbol{Q}$ jako diagonální a to z toho důvodu, že lze interpretovat hodnoty na diagonále jako váhy, které přímo ovlivňují náležitý stav.

Pro nalezení matice $\boldsymbol{K}$ platí, že

$$\Delta u(k) = -\boldsymbol{K}\Delta\boldsymbol{x}(k) = -\boldsymbol{R}^{-1}\boldsymbol{B_D}^T\boldsymbol{P}, \tag{3.3}$$

kde matice $\boldsymbol{P}$ je řešení algebraické Riccatiovy rovnice. Ta má tvar

$$\boldsymbol{A_D}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A_D} - \boldsymbol{P}\boldsymbol{B_D}\boldsymbol{R}^{-1}\boldsymbol{B_D}^T\boldsymbol{P} + \boldsymbol{Q} = 0. \tag{3.4}$$

Aby řešení mohlo existovat, tak musí platit, že dvojice $(\boldsymbol{A_D}, \boldsymbol{B_D})$ je řiditelná a dvojice $(\boldsymbol{Q}, \boldsymbol{A_D})$ je pozorovatelná.

Nesporná výhoda použití právě tohoto typu regulátoru je zaručená robustnost ve stabilitě, jelikož pro libovolnou volbu váhových matic $\boldsymbol{Q}$ a $\boldsymbol{R}$ Nyquistova křivka nevstupuje nikdy do jednotkové kružnice kolem kritického bodu $[-1, j0]$.

3.2 Postup návrhu

Před návrhem konkrétního zpětnovazebního lineárního stavového regulátoru je nejprve nutné určit, zda systém splňuje nutné podmínky pro nalezení konkrétního řízení ze sekce 3.1.

3.2.1 Řiditelnost

Aby bylo možné navrhnout zpětnovazební lineární stavový regulátor, musí být dvojice $(\boldsymbol{A_D}, \boldsymbol{B_D})$ řiditelná. Lineární dynamický systém je řiditelný právě tehdy, když je hodnost matice řiditelnosti rovna dimenzi vektoru stavu. Jelikož uvažovaný systém má právě 4 stavy, je matice řiditelnosti dána jako

$$\boldsymbol{Q}_{\check{R}} = \left[\boldsymbol{B}_D, \boldsymbol{A}_D\boldsymbol{B}_D, \boldsymbol{A}_D^2\boldsymbol{B}_D, \boldsymbol{A}_D^3\boldsymbol{B_D}\right]. \tag{3.5}$$

Dosazením vznikne matice řiditelnosti, která má tvar

$$\boldsymbol{Q}_{\check{R}} = \begin{bmatrix} 0{,}0003 & 0{,}0008 & 0{,}0014 & 0{,}0019 \\ 0{,}0165 & 0{,}0164 & 0{,}0163 & 0{,}0161 \\ 0{,}0008 & 0{,}0023 & 0{,}0037 & 0{,}0051 \\ 0{,}0469 & 0{,}0453 & 0{,}0421 & 0{,}0375 \end{bmatrix}. \tag{3.6}$$

Hodnost matice řiditelnosti je

$$rank(\boldsymbol{Q}_{\check{R}}) = 4. \tag{3.7}$$

Matice řiditelnosti má plnou hodnost a systém je tedy řiditelný, což znamená, že lze pokračovat v návrhu zpětnovazebního LQ regulátoru.

3.2.2 Váhové matice

Po splnění všech podmínek na existenci řešení zbývá vhodně zvolit váhové matice $\boldsymbol{Q}$ a $\boldsymbol{R}$. Tento proces vyžaduje iterativní úpravu těchto matic, dokud nedochází k požadovanému chování zapojeného regulátoru do systému.

Matice $\boldsymbol{Q}$ je tvořena tak, že na diagonále obsahuje váhy jednotlivých stavů. Vzhledem k tomu, že primární cíl je co nejrychlejší odregulování kmitů nákladu, tak bude mít tento stav nejvyšší váhu. Poloha dronu je na druhou stranu veličina, u které je vhodné zvolit nižší váhu, aby primárně došlo ke tlumení kmitů, a následně ke sledování trajektorie. Zvolená matice $\boldsymbol{Q}$ má tvar

$$\boldsymbol{Q} = \begin{bmatrix} 0{,}06 & 0 & 0 & 0 \\ 0 & 0{,}25 & 0 & 0 \\ 0 & 0 & 25{,}94 & 0 \\ 0 & 0 & 0 & 0{,}44 \end{bmatrix}. \tag{3.8}$$

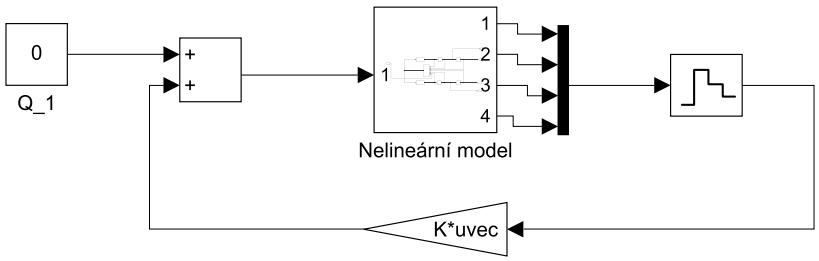
Matice $\boldsymbol{R}$ je váhová matice vstupů do systému. Systém obsahuje pouze jeden vstup, a tedy matice $\boldsymbol{R}$ přechází na skalár. Zvolená hodnota je

$$R = 2{,}3. \tag{3.9}$$

Po stanovení váhových matic lze vyřešit rovnice (3.3) a (3.4). Řešením lze získat matici $\boldsymbol{K}$, která má tvar

$$\boldsymbol{K} = \begin{bmatrix} 0{,}53 & 1{,}98 & 3{,}29 & 2{,}26 \end{bmatrix} \tag{3.10}$$

Nyní lze kolem systému uzavřít zápornou zpětnou vazbu a do ní připojit odvozený LQ regulátor. Schéma zapojení je znázorněno na obrázku 3.1. Jelikož je regulátor odvozován pro diskrétní systém, ale nelineární model je spojitý, je tedy nutné za něj připojit tvarovač nultého řádu, a tím převést spojitý výstup ze systému na diskrétní.



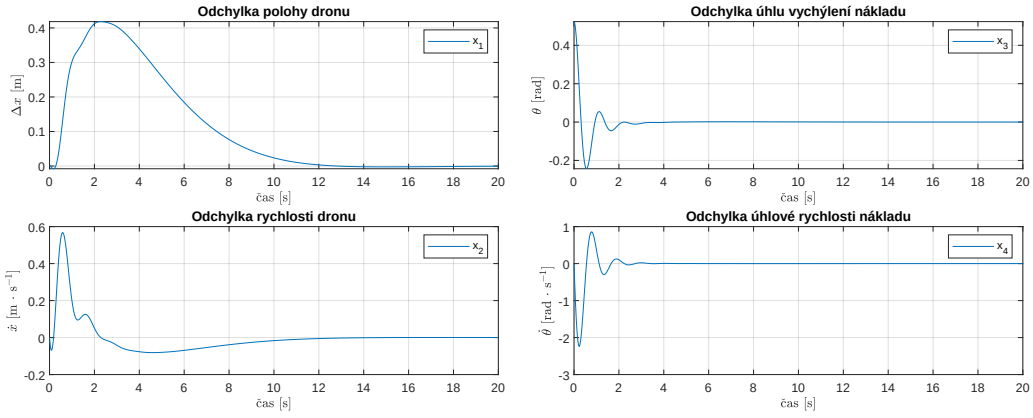
Obrázek 3.1: Schéma zapojení regulátoru v Simulinku

3.3 Validace řízení

V této části jsou diskutovány dosažené výsledky a kvalita dosažené regulace. Pro účely zvalidování navrženého řízení byl na systém kladen důraz na odregulování všech stavů do uvažovaného pracovního bodu. Regulátor je tedy nasazován přímo na nelineární systém, jak je znázorněno na obrázku 3.1. Nastavené počáteční podmínky jsou všechny nulové, kromě stavu x_3, tedy výchylky nákladu, která je nastavená na hodnotu $\frac{\pi}{6}$.

3.3.1 Řízení nelineárního systému

Získané grafy průběhů jednotlivých stavů lze vidět na obrázku 3.2, kde je znázorněn výstup z experimentu, kdy byl regulátor nasazen na nelineární systém bez uvažování poruch či šumů měření.



(a) Vývoj stavů x_1 a x_2

(b) Vývoj stavů x_3 a x_4

Obrázek 3.2: Vývoj stavů při řízení nelineárního systému

Vývoj odchylky polohy dronu velmi dobře reflektuje vliv volby nastavení nízké váhy na tento stav v matici $\boldsymbol{Q}$ v rovnici (3.8) a tedy nízké zesílení na tomto stavu v matici $\boldsymbol{K}$. Je vidět, že nastavením nízké váhy v poměru ku ostatním stavům, dochází k velké toleranci, co se odchylky od požadované hodnoty týče. Také doba regulace je znatelně delší v porovnání s ostatními průběhy stavů. Totéž platí pro graf odchylky rychlosti dronu, jejíž váha není tak nízká, ale je přímo ovlivňována polohou dronu.

Ačkoliv bylo počáteční vychýlení zvoleno jako relativně velké oproti pracovnímu bodu, tak dochází k velmi rychlé regulaci na požadovanou hodnotu. To je dáno především volbou velkého zesílení na tento stav v poměru ku ostatním v matici $\boldsymbol{K}$.

Tento LQ regulátor je navržen tak, aby co nejrychleji odreguloval právě vychýlení nákladu, což je splněno. Problém ovšem nastává v případě, že by byl uvažován šum měření. Pak by toto zesílení velmi významně zvyšovalo působení chyby měření a mohlo by dojít k destabilizaci systému. Tato situace bude testována následovně.

3.3.2 Řízení nelineárního systému s aditivním šumem

Nyní je zřejmé, že řízení planárního modelu odpovídá očekávání. Pro testování robustnosti je k jednotlivým stavům přidán šum. Tímto lze otestovat řízení v neideálních, a tedy zašumělých podmínkách, což je žádoucí pro nasazení na skutečném systému.

K výstupu simulovaného nelineárního systému byl přičten šum. Tento šum je generován z normálního rozdělení, jelikož tak lze jednoduše modelovat chybu měření senzorů. Hodnota šumu se liší v závislosti na konkrétním stavu a je tedy reprezentován vektorem. V kontextu tohoto testování je tedy šum $\boldsymbol{v_n}$ charakterizován jeho kovarianční maticí Σ a vektorem středních hodnot μ ve tvaru

$$\boldsymbol{v_n} \sim \mathcal{N}(\mu, \Sigma). \tag{3.11}$$

Pro každý stav systému byla stanovena jeho specifická varianční hodnota, škálovaná vzhledem k charakteristikám odpovídající fyzikální veličiny. Porucha byla nastavena tak, aby simulovala jednoduchou chybu měření. Střední hodnota generovaného normálního rozdělení je tedy vždy uvažována jako nulová, především kvůli tomu, že všechny stavy mají pracovní bod uvažovaný právě v nule. Tedy platí, že

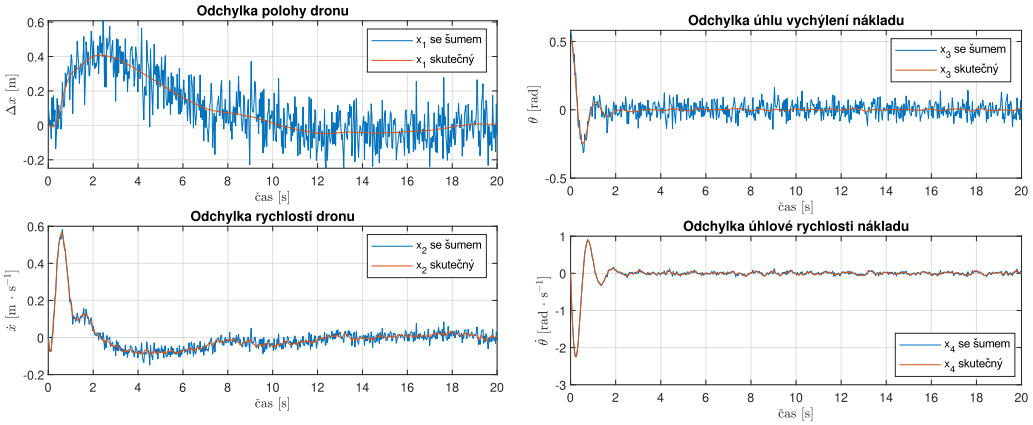
$$\mu = \mathbf{0}, \tag{3.12}$$

$$\Sigma = \begin{bmatrix} 0{,}1^2 & 0 & 0 & 0 \\ 0 & 0{,}02^2 & 0 & 0 \\ 0 & 0 & 0{,}05^2 & 0 \\ 0 & 0 & 0 & 0{,}02^2 \end{bmatrix}. \tag{3.13}$$

Pak lze simulaci aditivního šumu pro všechny stavy zapsat jako

$$\boldsymbol{x} = \boldsymbol{x}_{\text{skutečný}} + \boldsymbol{v_n}, \tag{3.14}$$

kde vektor $\boldsymbol{v_n}$ je tedy dán jako $\boldsymbol{v_n} \sim \mathcal{N}(\mathbf{0}, \Sigma)$.



(a) Vývoj stavů x_1 a x_2 při aplikaci šumu (b) Vývoj stavů x_3 a x_4 při aplikaci šumu

Obrázek 3.3: Vývoj stavů při řízení nelineárního systému s aditivním šumem

Z tohoto experimentu je zřejmé, že především pro odchylku úhlu vychýlení nákladu a odchylky polohy dronu je vliv šumu velmi významný, jak je znázorněno na obrázku 3.3. Pro vychýlení nákladu přidaný šum způsobí to, že se tato regulační odchylka nikdy neustálí na nule, ale bude kmitat kolem této hodnoty, jak lze vidět na obrázku 3.3b. Ačkoliv se zdá, že toto není očekávané chování a regulátor neplní svou funkci správně, tak opak je pravdou. Regulátor je schopen velmi rychle odregulovat počáteční vychýlení nákladu a držet tuto regulační odchylku na minimálních oscilacích, které lze považovat za zanedbatelné.

Regulační odchylka polohy dronu, jenž je znázorněna na obrázku 3.3a, vykazuje velmi podobné chování s tím rozdílem, že doba regulace je znatelně delší. To je způsobeno již zmíněným nastavením nízkého zesílení v matici $\boldsymbol{K}$ pro tento stav.

Co se týče odchylky rychlosti dronu a odchylky úhlové rychlosti nákladu, tak jejich vývoj vykazuje velmi podobné a očekávané chování. Tyto stavy ovlivňují rychlost a přesnost regulace pro stavy x_1 a x_3.

Závěrem lze tedy prohlásit, že navržené řízení pro planární model je robustní a odolné vůči chybám, což je klíčová vlastnost pro nasazení na reálný či simulační model.

Simulační prostředí 4

V této kapitole jsou detailně popsány technologie a nástroje využité pro simulaci dronu s nákladem a pro implementaci řídícího algoritmu. Následně je představena architektura systému, která integruje simulační prostředí s komponentami řídícího algoritmu, což je zásadní pro pochopení, jakým způsobem a proč jsou specifické technologie využívány v rámci celkového systému. Detailně je zde popsán simulační model dronu, včetně provedených úprav, integrace senzorů a způsobu jejich simulace. Kapitola dále obsahuje popis uživatelského rozhraní, které bylo navrženo pro intuitivní ovládání dronu a sledování jeho operací během simulace. Toto rozhraní umožňuje snadnější monitorování při testování řídících algoritmů a chování dronu v různých simulačních scénářích.

4.1 Použité technologie

Tato sekce se zabývá především uvedením použitých technologií, vymezením důležitých pojmů s nimi svázaných a jejich integrace v systému. Tato sekce je důležitá především pro pochopení architektury řídícího systému a simulačního prostředí jako jednoho celku.

4.1.1 PX4

PX4-autopilot je pokročilý open-source software pro řízení bezpilotních letounů. Klíčovou součástí je platforma pro automatizaci, navigaci a plánování misí. Součástí systému PX4 [24c] je modul PX4-SITL (Software In The Loop). Ten umožňuje simulaci letu bez nutnosti fyzického dronu v simulačním prostředí, jako Gazebo nebo jMAVSim a testování řídících algoritmů v bezpečném virtuálním prostředí.
PX4 používá pro komunikaci mezi dronem a externími systémy, jako například Gazebo, komunikační protokol MAVLink (Micro Air Vehicle Link). Pro interní komunikaci je používán protokol uORB (uAV Object Request Broker).

Tento systém PX4 byl zvolen především kvůli jeho schopnosti poskytnout detailní informace o dronu a umožnit jeho jednoduché a efektivní řízení. Kromě toho, kompatibilita se simulačním prostředím Gazebo a podpora integrace s různými softwarovými platformami, jako například ROS2, jenž je detailně představen v sekci 4.1.3, představují významné výhody pro realizaci a testování řídících algoritmů.

Vhodným doplňkem k PX4 je aplikace QGroundControl. Jedná se o aplikaci, která umožňuje komplexní plánování misí či jejich monitorování. Poskytuje grafické rozhraní pro konfiguraci PX4-autopilota, přehled o letových datech či možnost přepínání mezi letovými režimy. Tato aplikace byla při vývoji používána především pro monitorování stavu dronu a přepínání mezi letovými režimy. Později byla aplikace nahrazena vlastní implementací aplikace specificky pro potřeby této práce, viz sekce 4.4 na str. 25.

4.1.2 Gazebo simulátor

Řídící algoritmy je vhodné nejprve testovat na simulačním modelu. K tomuto účelu bylo použito simulační prostředí Gazebo [24a]. Jedním z hlavních důvodů použití je právě kompatibilita s PX4.

Simulační prostředí Gazebo poskytuje realistické simulace fyzikálních interakcí mezi objekty, včetně kolizí, a umožňuje sledovat vývoj chování v téměř reálném čase. Má podporu pro široké spektrum senzorů a umožňuje vývoj vlastních pluginů, což je klíčové zejména pro získání veškerých aktuálních dat z modelu, které nejsou dostupné z PX4, jako tomu je v případě získávání polohových dat o nákladu v sekci 5.1.

4.1.3 ROS2 a jeho integrace do systému

ROS2 (Robot Operating System 2) představuje modulární a distribuované prostředí pro vývoj a implementaci softwaru v oblasti robotiky. Toto prostředí se skládá z nástrojů a knihoven, které výrazně usnadňují proces návrhu a realizace algoritmů, a to nejen v oblasti řídících systémů, ale i v mnoha dalších robotických aplikacích. Jedním z klíčových prvků architektury ROS2 je koncept uzlu (node), který reprezentuje základní stavební blok celého systému. Každý uzel je zodpovědný za určitou specifickou funkcionalitu nebo část řídícího algoritmu. Například uzel může být zodpovědný za shromažďování a publikování dat z konkrétního senzoru, nebo může řídit rychlost jednotlivých motorů.

Komunikace mezi uzly v ROS2 je založena na modelu Data Distribution Service (DDS), což je robustní middleware protokol, který umožňuje efektivní a spolehlivý přenos dat mezi uzly. Tento model využívá principy tzv. publisher-subscriber. V tomto kontextu mohou jednotlivé uzly buď publikovat zprávy (topics), nebo je naopak odebírat. Díky tomu je zajištěna vysoká míra modularity a rozložení funkčnosti, což umožňuje efektivní škálování a přizpůsobení systému pro různé aplikace. Komunikační protokoly v rámci DDS mohou využívat jak UDP (User Datagram Protocol) pro rychlý a nenáročný přenos dat, tak i TCP (Transmission Control Protocol) pro spolehlivější, ale potenciálně pomalejší komunikaci, v závislosti na konkrétních požadavcích aplikace. Podrobnější informace o architektuře ROS2 lze nalézt v článku [Mac+22].

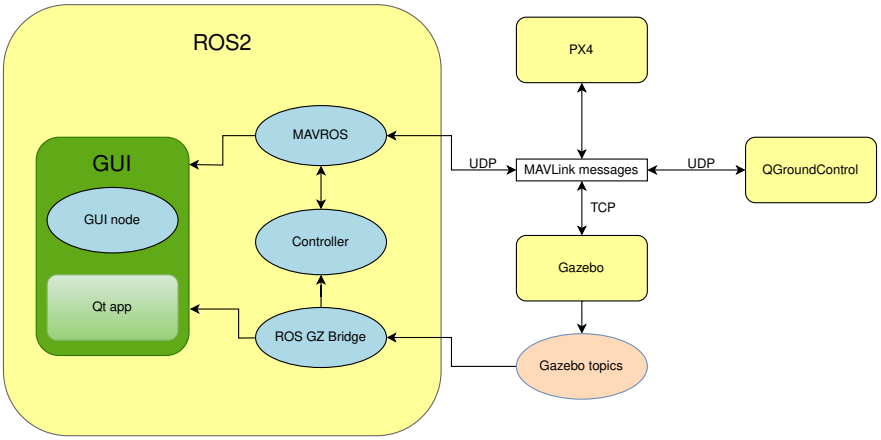
Toto prostředí bylo pro tuto práci zvoleno především kvůli schopnosti strukturovat řídící algoritmus do několika menších částí, které mezi sebou komunikují v reálném čase. Důležitou roli při volbě tohoto prostředí hraje právě kompatibilita s Gazebo simulátorem a s PX4.

Kompatibilita s PX4 ovšem není přímočará. Aby bylo možné zajistit komunikaci mezi PX4 a prostředím ROS2, musí být využito mezičlánku, který se nazývá MAVROS. Jedná se o komunikační uzel v ROS2, který přijímá data v podobě MAVLink zpráv a transformuje je na ROS2 zprávy, které publikuje jako topicy [Erm18].

4.2 Architektura systému

Tato sekce popisuje, jak jsou veškeré použité technologie ze sekce 4.1 používány a především propojeny.

Obrázek 4.1 poskytuje podrobný přehled architektury systému, který ilustruje integraci a komunikaci mezi klíčovými technologickými komponentami. Jednotlivé komponenty jsou vzájemně odděleny tak, aby byly lépe znázorněné vzájemné vazby. Jsou zde také znázorněny protokoly a rozhraní, která zprostředkovávají vzájemnou komunikaci. Detailní implementace uzlů v prostředí ROS2 je diskutována v kapitole 5. Je zde zahrnuta i aplikace, která byla vyvinuta v rámci jednotného přístupu ke sledování letu dronu a ovládání nastavení prostředí.



Obrázek 4.1: Diagram architektury systému

4.3 Simulační model

V této sekci je detailně popsán použitý simulační model dronu a jeho úprava pro účely této práce. Jsou zde také popsány použité senzory, což je klíčové pro pochopení, jakým způsobem jsou data získávána.

4.3.1 Model dronu

Zvolený model dronu, x500, integrovaný v PX4-autopilot, poskytuje ideální model pro testování řídících algoritmů pro bezpilotní letouny. Jedná se o konstrukci se čtyřmi výkonnými motory. Jeho konstrukce je vhodná pro různé vědecké a výzkumné účely, a to především díky adaptabilitě na různé typy užitečného zatížení a kompatibility s různými vědeckými a výzkumnými aplikacemi či zařízeními.

Tento model byl zvolen především díky své jednoduchosti a symetrii v osách x a y, což umožňuje použít stejně navržený LQ regulátor na každou osu zvlášť a není potřeba jej v závislosti na tvaru modelu či počtu motorů upravovat, jak již bylo zmíněno v kapitole 2. Model skutečného dronu x500 je ilustrován na obrázku 4.2.

Obrázek 4.2: Ilustrační obrázek modelu dronu x500 získaný z [24b]

4.3.2 Úprava modelu

Pro účely simulace a řízení chování dronu v reakci na zatížení byl k samotnému modelu x500 přidán náklad. Tento náklad je pro zjednodušení simulován jako kulový objekt, jenž je spojen s dronem prostřednictvím nehmotné tyče, jak již bylo velmi detailně představeno v sekci 2.1. To umožňuje nákladu pohybovat se volně v prostoru a věrně simulovat dynamiku skutečného dronu s nákladem.

Tyč je spojena s konstrukcí dronu pomocí sférického kloubu. Ten byl vybrán kvůli jeho schopnosti umožnit pohyb nákladu do všech směrů a rotaci kolem osy z, což věrně napodobuje reálné podmínky.

Na simulační model není aplikováno žádné přímé tlumení v daném kloubu, jenž spojuje konstrukci dronu se závěsem nákladu, stejně jako tomu bylo u matematického modelu. Tento fakt představuje značnou výzvu pro samotný řídící algoritmus. V klidovém stavu, kdy dron udržuje konkrétní výšku, dochází k netlumenému a nepredikovatelnému chování nákladu.

4.3.3 Senzory

Simulační prostředí Gazebo obsahuje širokou škálu podporovaných senzorů. U většiny senzorů lze simulovat šum či odchylku měření, což umožňuje testovat i situace, kdy jsou data ze senzorů ovlivněna šumem.

Samotný náklad byl osazen IMU (inertial measurement unit) senzorem, neboli inerciální měřící jednotkou. IMU senzor byl použit především z toho důvodu, že v matematickém modelu soustavy je uvažována úhlová rychlost nákladu. Pro samotné řízení je také třeba znát aktuální orientaci natočení nákladu, což lze získat prostřednictvím právě tohoto senzoru.

IMU senzor se skládá z akcelerometru, gyroskopu a magnetometru. Akcelerometr měří lineární zrychlení ve všech třech osách x, y, z. Gyroskop měří úhlovou rychlost také ve všech třech osách. Umožňuje také detekci rotace nebo náklonu. Magnetometr měří zemské magnetické pole země. Spojením dat z těchto tří senzorů prostřednictvím procesu, známého jako senzorická fúze, lze odhadnout kompletní orientaci zařízení v prostoru.

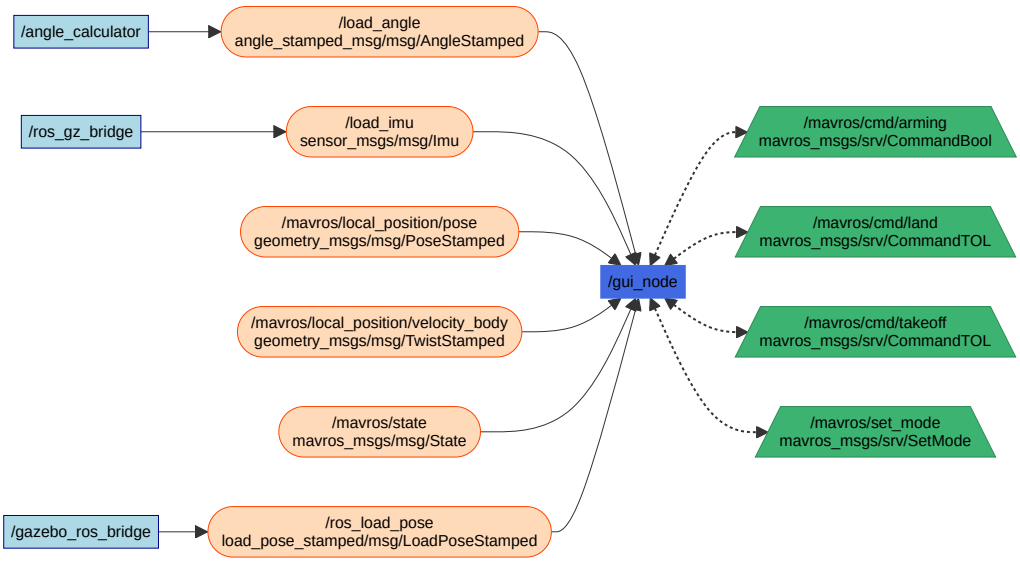
Dron je navíc osazen vlastním IMU senzorem, GPS modulem a také senzorem tlaku vzduchu, který simuluje měření atmosférického tlaku. Tyto senzory nejsou v této práci přímo použity, ale používá je PX4 k odhadu dalších veličin pomocí především Kalmanových filtrů. Tato odhadovaná data jsou poté využívána v řídícím algoritmu.

4.4 Uživatelské rozhraní

Tato sekce se věnuje vývoji uživatelského rozhraní, jehož hlavním úkolem je efektivní vizualizace klíčových stavových veličin systému, monitorování řídících procesů v reálném čase a intuitivní manipulace s řídícími algoritmy a přidruženými softwarovými komponenty. Aplikace byla vytvořena s použitím multiplatformního frameworku Qt, který podporuje programovací jazyk C++. Volba tohoto frameworku byla motivována jeho rozsáhlou dokumentací a schopností efektivní integrace s ROS2 prostředím.

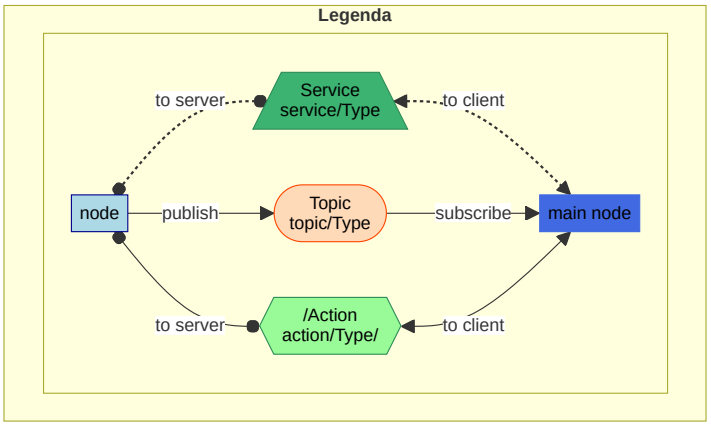
Architektura aplikace byla vytvářena s důrazem na modularitu, kdy data poskytovaná ROS2 topicy jsou zpracovávána v jiných separátních komponentách, než ty, které jsou zodpovědné za jejich vizualizaci. Výsledkem je rozdělení na backendové a frontendové komponenty, založené na zásadách moderních softwarových architektur.

Integrace aplikace do ROS2 prostředí je ilustrována na obrázku 4.3, který prezentuje datové toky mezi ROS2 topicy a samotnou aplikací.



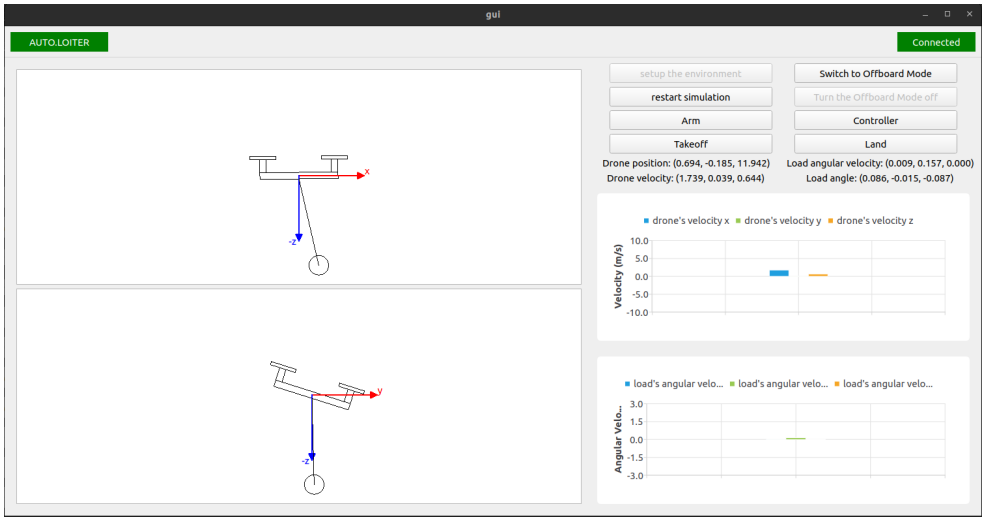
Obrázek 4.3: Diagram zapojení aplikace do ROS2 architektury

Pro všechny použité diagramy je využívána jednotná legenda s vysvětlivkami a lze ji nalézt na obrázku 4.4.



Obrázek 4.4: Vysvětlivky k ROS2 diagramům

Platí, že *main node* je vždy použito v daném kontextu pro označení hlavního uzlu, jenž je diagramem popisován.



Obrázek 4.5: Uživatelské rozhraní

Vizuální uspořádání aplikace je intuitivní, jak ilustruje obrázek 4.5. Aplikace je rozdělena do dvou částí. V levé části je znázorněn pohled na dron ve směru osy x a y. Zde je velmi dobře vidět v jakém stavu je aktuálně náklad a jak se dron naklání, aby náklad stabilizoval. Tato část slouží jako náhrada grafické části, jelikož poskytuje veškeré potřebné vizualizace. Díky tomu by bylo možné algoritmus nasadit na skutečný dron a pomocí této aplikace neztratit pojem o chování dronu i když by nebyl přímo v dohledu.

Pravá část aplikace je určena interaktivním ovládacím prvkům, které umožňují snadno přepínat mezi různými letovými režimy dronu a řídit podpůrné komponenty nezbytné pro simulaci a řízení. Tlačítko *setup the environment* slouží právě ke spuštění všech podpůrných komponent potřebných pro běh simulace. Je totiž nutné vždy spustit PX4 spolu s Gazebo simulátorem, MAVROS uzel a veškeré ROS2 mosty pro dostupnost dat z Gazebo. Tlačítko *restart simulation*, jak již název napovídá, slouží k restartování Gazebo simulace. Ostatní tlačítka využívají toho, že PX4 poskytuje služby, které umožňuje přepínání stavů dronu. Typickým příkladem je tlačíto *Takeoff*, které umožňuje dronu vzlétnout do přednastavené výšky 10 metrů. Poté je zde tlačítko *Land*, které umožňuje dronu přistát na konkrétní pozici, kde se aktuálně nachází. Tlačítka *Switch to Offboard Mode* a *Switch the Offboard Mode off* slouží k přepnutí dronu do režimu, kdy přestává být řízen vnitřními regulátory PX4 a je schopen přijímat pokyny zde konkrétně z ROS2. Aby dron mohl být řízen vyvinutým řídícím algoritmem, tak musí být právě v tomto režimu. Poslední, nejdůležitější, tlačítko je *Controller*. Po stisknutí se objeví dialogové okno, kde je umožněno vybrat trajektorii letu dronu.

Aplikace umožňuje výběr mezi kruhovou trajektorií s definovatelným poloměrem a lineární trajektorií, kde lze specifikovat koncový cílový bod. Po stisknutí libovolného tlačítka je v levém dolním rohu vždy znázorněno, zda se dané akce vykonala úspěšně, či nikoliv.

Pod jednotlivými ovládacími tlačítky se nachází informace o aktuálních sledovaných stavech. Pod nimi se nachází graf vykreslující aktuální rychlost dronu a úhlovou rychlost nákladu ve všech osách. Ačkoliv tyto hodnoty jsou již vypisovány, tak je i přesto vhodné právě tyto vizualizovat pomocí grafu, jelikož to vytváří lepší představu o aktuálním chování dronu.

Aplikace také v horním levém rohu obsahuje indikátor aktuálního režimu, který je nastavený na dronu. Na druhé straně se nachází indikátor připojení k ROS2 prostředí a k Gazebo simulátoru. Tyto indikátory jsou naprosto klíčové a nezbytné pro správný chod aplikace a pro informovanost o aktuálním dění.

Řídící algoritmus 5

Tato kapitola se věnuje komplexnímu rozboru řídícího algoritmu, který aplikuje teoretické koncepty uvedené v předchozích kapitolách 2 a 3. Úvodní část kapitoly podrobně popisuje proces získání stavového vektoru, jehož definice je specifikována v rovnici (2.16). Dále se kapitola zabývá metodikou výpočtu matice zesílení regulátoru $\boldsymbol{K}$ a procesem jejího ladění. Klíčovou součástí kapitoly je pak popis struktury řídícího uzlu, jenž skládá tyto elementární části do jednoho celku, a vývoje algoritmu pro plánování trajektorie. Struktura řídícího algoritmu je demonstrována na přehledném diagramu 5.3.

Implementace řídícího algoritmu byla provedena v rámci prostředí ROS2. Algoritmus je strukturován do několika samostatných uzlů, přičemž každý uzel je zodpovědný za specifickou část řídícího procesu. To umožňuje nejen lepší orientaci v kódu, ale také zvyšuje efektivitu vývoje a umožňuje paralelní zpracování jednotlivých částí algoritmu, což vede k lepší optimalizaci.

Většina těchto uzlů je implementována v programovacím jazyce C++, jelikož je nejvhodnější k použití v časově kritických aplikacích, jako je právě řízení stabilizace nákladu. Uzly, které nejsou časově kritické, jsou implementovány v programovacím jazyce Python, což usnadňuje rychlý vývoj s rozsáhlými dostupnými knihovnami.

5.1 Stavový vektor

Stavový vektor pro řízení dronu v prostoru je uvažován stejný jako v rovnici (2.16). Řídící algoritmus bude tedy uvažovat samostatný stavový vektor pro každou osu zvlášť, jak již bylo uvedeno v kapitole 2 a ilustrováno na obrázku 2.1. Samotný řídící algoritmus je navržen tak, aby řízení obou os probíhalo nezávisle a každá osa byla řízena vlastním LQ regulátorem odvozeným v kapitole 3. Aby to bylo možné, tak je potřeba uvažovat stavy v jednotném souřadném systému. Počátek tohoto souřadného systému je roven počátku souřadnic Gazebo simulace. PX4 spolu s MAVROS uvažují také tento souřadný systém jako počátek. Dalším důvodem zavedení jednotného souřadného systému je umožnění efektivního plánování trajektorie dronu. V této práci bude tento systém označován jako „lokální souřadný systém“.

Je zásadní poznamenat, že tento souřadný systém používá formát ENU (East-North-Up), což znamená, že osa X je orientována směrem k východu (East), osa Y směrem na sever (North) a osa Z směrem vzhůru (Up). Tato orientace os je klíčová pro zachování konzistence a správného přepočtu dat v rámci simulace.

5.1.1 Poloha a rychlost dronu

PX4 ve spojení s MAVROS poskytuje širokou škálu topiců. Jedním z těchto topiců je `/mavros/local_position/pose`. Tento topic poskytuje informace o poloze a orientaci dronu v lokálním souřadném systému. Data jsou získávána PX4 autopilotem přímo ze simulace. Gazebo je interně počítá na základě fyzikálního modelu prostředí a fůze dat ze senzorů. Data jsou poté interně využita v rozšířeném Kalmanově filtru, který dává odhad právě o pozici a orientaci dronu. Publikace dat probíhá s frekvencí 30 Hz. Tato frekvence je dána implmementací v PX4. Pro získání rychlosti platí totéž, jako pro polohu dronu, s jediným rozdílem, že byl použit topic `/mavros/local_position/velocity_local`. Data jsou v lokálním souřadném systému a jsou publikována také s frekvencí 30 Hz.

5.1.2 Úhel vychýlení nákladu

Určení výchylky nákladu vyžaduje specifický přístup, jelikož náklad je k dronu připojen externě a nejsou pro něj dostupné předdefinované zprávy. Výchylkou nákladu je uvažován úhel mezi zápornou částí osy z lokálního souřadného systému a závěsem nákladu, jak již bylo ilustrováno na obrázku 2.1. Tento úhel je počítán za použití trigonometrie, což umožňuje přesné určení orientace nákladu vůči dronu. Výpočet tohoto úhlu umožňuje následně přesně monitorovat a regulovat chování nákladu v jednotlivých osách.

Pro výpočet vychýlení nákladu je nutné znát jeho přesnou polohu. Tato informace je dostupná pouze přímo v Gazebo simulaci. Je tedy nutné vyvinout speciální plugin, který umožňuje extrahovat polohová data o objektu nákladu v simulaci. Tento Gazebo plugin je implementován jako dynamicky linkovaný modul (.so soubor pro systémy založené na linuxu) a následně registrován přímo v konfiguračním sdf souboru Gazebo modelu.

Plugin využívá Gazebo API, které poskytuje funkce a komponenty využitelné právě pro monitorování polohy entity v simulaci. V principu tento plugin funguje tak, že postupně iteruje skrze entitní komponenty modelu a identifikuje požadované objekty, ze kterých jsou extrahována data o poloze a orientaci. Veškerá data jsou následně publikována prostřednictvím Gazebo topicu `/load_pose`. Jelikož je poloha nákladu v simulačním modelu definována relativně vůči tělu dronu, pak i publikovaná data jsou právě relativní vůči tělu dronu.

Gazebo topic ovšem není viditelný pro ROS2 prostředí, takže je potřeba tento topic převést právě na ROS2 topic. K tomu byl vytvořen uzel `gazebo_ros_bridge`, který slouží jako most mezi těmito dvěma systémy. Používá jak Gazebo API, pro přístup k Gazebo topicu, tak i rcl (ROS client library), pro vytvoření ROS2 uzlu a publikování ROS2 topicu.

V principu tento uzel pouze odebírá zprávy z `/load_pose` a převádí je na ROS2 zprávy. Data publikuje jako topic `/ros_load_pose`, který je dále využit pro výpočet vychýlení nákladu.

Pro samotný výpočet úhlu vychýlení nákladu byl implementován uzel s názvem `angle_calculator` v ROS2. Ten přijímá, již zmíněná, data o poloze nákladu a dronu. Dalším důležitým krokem je zajistit, aby všechna data, použita při výpočtu výchylky, byla konzistentně reprezentována ve stejném souřadném systému. Proto je poloha nákladu převedena do lokálního souřadného systému. Tento převod je realizován pomocí rotační matice, která je konstruována z kvaternionu orientace dronu. Kvaternion je obecně reprezentován jako

$$\boldsymbol{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \end{bmatrix}. \tag{5.1}$$

Tento kvaternion představuje orientaci těla dronu vůči lokálnímu souřadnému systému.

Postup je takový, že nejdříve je nutné kvaternion normalizovat, čímž zajistit, že rotace bude validní a rotační matice bude ortogonální. Kvaternion je normalizovaný, pokud platí, že

$$\|\boldsymbol{q}\| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1. \tag{5.2}$$

Pro normalizaci kvaternionu platí vztah

$$\boldsymbol{q}_{\text{normalizovaný}} = \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|}. \tag{5.3}$$

Za předpokladu, že kvaternion je již normalizovaný, tak lze zkonstruovat rotační matici. Ta je dána vztahem

$$\boldsymbol{R} = \begin{bmatrix} 2\left(q_0^2 + q_1^2\right) - 1 & 2\left(q_1 q_2 - q_0 q_3\right) & 2\left(q_1 q_3 + q_0 q_2\right) \\ 2\left(q_1 q_2 + q_0 q_3\right) & 2\left(q_0^2 + q_2^2\right) - 1 & 2\left(q_2 q_3 - q_0 q_1\right) \\ 2\left(q_1 q_3 - q_0 q_2\right) & 2\left(q_2 q_3 + q_0 q_1\right) & 2\left(q_0^2 + q_3^2\right) - 1 \end{bmatrix}. \tag{5.4}$$

Pro rotaci polohy nákladu $\boldsymbol{p}$ do lokálního souřadného systému tedy platí vztah

$$\boldsymbol{p_1} = \boldsymbol{R} \cdot \boldsymbol{p_2}, \tag{5.5}$$

kde $\boldsymbol{p_1}$ je poloha nákladu v lokálním souřadném systému a $\boldsymbol{p_2}$ je poloha nákladu vyjádřená v souřadném systému těla dronu.

Po zarovnání souřadných systémů zbývá pouze využít trigonometrie a určit výchylku nákladu. Vztah pro výpočet výchylky je tedy dán jako

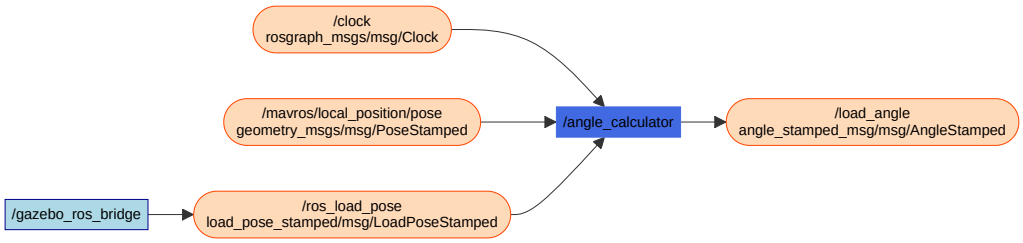
$$\theta = \arctan\left(\frac{x}{z}\right), \tag{5.6}$$

kde θ je průmět úhlu vychýlení nákladu do osy x, x je souřadnice polohy nákladu na ose x v lokálním souřadném systému a je z souřadnice polohy nákladu na ose z v lokálním souřadném systému. Výsledný úhel v každé ose je tedy dán jako

$$\theta_x = \arctan\left(\frac{x}{z}\right), \tag{5.7}$$

$$\theta_y = \arctan\left(\frac{y}{z}\right). \tag{5.8}$$

Tento úhel je následně převeden do ROS2 zprávy *AngleStamped*, vyvinuté specificky pro předávání dat o úhlu, a poté jsou publikována jako topic `/load_angle`.
Na obrázku 5.1 lze vidět začlenění uzlu pro výpočet výchylky do architektury ROS2.



Obrázek 5.1: Diagram integrace uzlu pro výpočet výchylky

5.1.3 Úhlová rychlost nákladu

Pro získání tohoto stavu již není potřeba vyvíjet Gazebo plugin, ale lze použít IMU senzor. Tento senzor je umístěn do středu nákladu. Umožňuje také v Gazebo nastavit topic, na který budou data periodicky publikována. Frekvence dat je nastavena na 30 Hz, což je plně dostačující a odpovídá to frekvencím, na kterých pracuje PX4-autopilot.

Aby byla data dostupná i v prostředí ROS2, je opět nutné použít most mezi ROS2 a Gazebo topicy. K tomuto účelu již nebylo nutné implementovat vlastní uzel, který by fungoval jako most. Místo toho bylo využito již existujícího, který podporuje základní typy zpráv.

Tento senzor publikuje data o úhlové rychlosti, orientaci a lineárním zrychlením. Všechna tato data jsou ve vlastním souřadném systému senzoru a proto je opět nutné je přepočítat do lokálního souřadného systému.

Postup získání matice rotace $\boldsymbol{R}$ bude stejný, jako tomu bylo v rovnici (5.4). Zde dochází pouze ke změně v tom, že uvažovaný kvaternion reprezentuje orientaci IMU senzoru vůči lokálnímu souřadnému systému. Z toho vyplývá, že matice $\boldsymbol{R}$ bude sice počítána stejně jako v rovnici (5.4), ale rotace bude inverzní, tedy matice rotace bude inverzní.
Pro rotaci vektoru úhlové rychlosti nákladu ω_2 do lokálního souřadného systému tedy platí, že

$$\omega_1 = \boldsymbol{R}^{-1} \cdot \omega_2, \tag{5.9}$$

kde ω_1 je vektor úhlové rychlosti nákladu v lokálním souřadném systému a ω_2 je vektor úhlové rychlosti nákladu vyjádřený v souřadném systému samotného IMU senzoru.

5.2 Výpočet matice K

Aby bylo možné odladit parametry LQ regulátoru přímo na řízeném systému za běhu, tak byl vytvořen ROS2 uzel `dlq_k_matrix_calculator`. Tento uzel je jako jediný napsán v Pythonu, jelikož poskytuje knihovnu *control*, která obsahuje optimalizované funkce pro výpočet zpětnovazebního zesílení matice $\boldsymbol{K}$. Tento uzel tedy obsahuje diskretizované matice stavového popisu systému $\boldsymbol{A_D}$, $\boldsymbol{B_D}$ popisující matematický model linearizovaný v rovnovážném stavu z rovnice (2.21). Výsledné parametry regulátoru jsou závislé na volbě návrhových parametrů, tedy na volbě váhové matice $\boldsymbol{Q}$ (3.8) a skaláru R (3.9), které je možné v tomto uzlu upravovat.

Po výpočtu je výsledná matice převedena do požadovaného formátu ROS2 zprávy, který je zvláště vhodný pro manipulaci s vektorovými a maticovými daty v distribuovaném systému. Publikace výsledné matice na topic `/dlq_k_matrix` probíhá s frekvencí 2 Hz, což odráží skutečnost, že úpravy parametrů regulátoru jsou prováděny primárně v rámci ladící fáze a nevyžadují vysokou frekvenci během stabilního provozu systému.

5.3 Řídící uzel

Tento uzel, `lqr_controller`, byl vyvinut jako jádro řídícího systému. Primárním cílem je synchronizovat data z různých zdrojů, vytvořit a zpracovávat stavové vektory a aplikovat zpětnovazební zesílení prostřednictvím matice $\boldsymbol{K}$. Zajišťuje také, aby všechna data použita při výpočtu byla vyjádřena ve správném požadovaném souřadném systému.

Stavové vektory pro osy x a y jsou sestaveny a použity pro výpočet akčních zásahů, které jsou oddělené pro každou osu zvlášť. Uvažované stavové vektory jsou

detailně popsány v sekci 5.1. Akční zásahy jsou vyjádřeny jako

$$u_x = -\boldsymbol{K} \cdot \boldsymbol{x}, \tag{5.10}$$

$$u_y = -\boldsymbol{K} \cdot \boldsymbol{y}, \tag{5.11}$$

kde $\boldsymbol{x}$ je stavový vektor modelu ve směru osy x a $\boldsymbol{y}$ je stavový vektor modelu ve směru osy y.

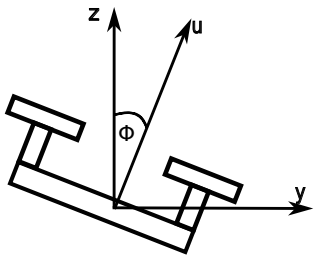
V návaznosti na matematickém modelu soustavy je zřejmé, že tento akční zásah reprezentuje průmět výslednice tahu dronu a gravitační síly do dané osy. Tyto akční zásahy jsou počítány v lokálním souřadném systému. PX4-autopilot poskytuje již předpřipravené topicy pro vlastní řízení dronu. Umožňuje dron řídit pomocí rychlosti, polohy nebo náklonu. Použitý topic pro řízení náklonu je `/mavros/setpoint_raw/attitude`. Ten vyžaduje kromě náklonu také tah, který je reprezentován hodnotou v intervalu <0,1>. Náklon je vyžadován ve tvaru kvaternionu, který musí být v souřadném systému těla dronu a proto je nutné akční zásahy u_x a u_y přepočítat právě do tohoto souřadného systému.

Pro přepočet akčních zásahů do správného souřadného systému je využito faktu, že topic `mavros/local_position/pose` poskytuje i informaci o orientaci dronu vůči lokálnímu souřadnému systému ve formě kvaternionu. Tento kvaternion byl použit pro výpočet matice rotace stejně jako tomu bylo v rovnici (5.4). Zde ovšem nastává situace, kdy dochází k transformaci z lokálního souřadného systému do souřadného systému těla dronu. Z toho vyplývá, že je potřeba opět uvažovat inverzní transformaci a tedy vypočítanou matici rotace $\boldsymbol{R}$ transponovat či určit její inverzi. Pro správný přepočet akčních zásahů do souřadného systému těla dronu lze využít přepočet

$$\boldsymbol{u_1} = \boldsymbol{R}^{-1} \cdot \boldsymbol{u_2}, \tag{5.12}$$

kde $\boldsymbol{u_1}$ jsou akční zásahy v souřadném systému těla dronu a $\boldsymbol{u_2}$ jsou akční zásahy v lokálním souřadném systému.

Tyto akční zásahy je nutné převést z fyzikální veličiny tah na náklon dronu. Pro odvození převodového vztahu lze vycházet z planárního modelu samotného dronu.



Obrázek 5.2: Planární model samotného dronu

Výchozí vztah lze tedy odvodit z obrázku 5.2. Pro zjednodušení bude odvození vyjádřeno pouze pro osu y. Pro osu x je odvození identické.

Výchozí vztah má tedy tvar

$$\ddot{y} \cdot m = -u_y \cdot \sin(\phi), \tag{5.13}$$

kde levá část rovnice reprezentuje tah působící ve směru osy y a pravá část představuje složku vektoru $\boldsymbol{u}$ působící na celkový tah ve směru osy y. Znaménko na pravé straně rovnice je dáno tím, jaký směr rotace je uvažován jako kladný.

Zde lze použít další zjednodušení. Jelikož je pro návrh řízení uvažován linearizovaný model, který očekává náklon dronu minimální, tak lze tento vztah upravit na jednodušší a to za předpokladu, že jsou splněny určité podmínky pro rovnovážný stav, které jsou definované jako

$$\phi_0 = 0, \tag{5.14}$$

$$u_y = m \cdot g. \tag{5.15}$$

Poté lze rovnici (5.13) zjednodušit na

$$\ddot{y} = -g \cdot \phi. \tag{5.16}$$

Vyjádřením lze získat vztah pro rotaci o úhel ϕ kolem osy x a rotaci o úhel θ kolem osy y. Tyto vztahy mají tvar

$$\phi = -\frac{\ddot{y}}{g}, \tag{5.17}$$

$$\theta = -\frac{\ddot{x}}{g}. \tag{5.18}$$

Zároveň také platí, že

$$\ddot{y} = \frac{u_y}{m}, \tag{5.19}$$

$$\ddot{x} = \frac{u_x}{m}, \tag{5.20}$$

kde u_y je akční zásah ve směru osy y, u_x je akční zásah ve směru osy x a m je hmotnost celého systému.

Výpočet pro úhel θ v rovnici (5.18) je sice korektní, ale je potřeba jej upravit tak, aby směr rotace odpovídal formátu ENU a byl interpretován správně. Proto tedy výpočet θ je v řídícím algoritmu upraven korekcí znaménka, tedy na tvar

$$\theta = \frac{\ddot{x}}{g}. \tag{5.21}$$

Tento výpočet určuje požadovaný náklon v souřadném systému těla dronu.

Aby nedocházelo k destabilizaci dronu při nadměrném akčním zásahu, tak je na náklon aplikována saturace. Ta je nastavena tak, aby náklon byl vždy v intervalu $\langle -15°, 15° \rangle$. Tato hodnota byla nastavena na základě pozorování chování systému a testů.

Následně je nutné transformovat vypočítané úhly do kvaternionu, což je realizováno pomocí knihovny *tf2*. Ta poskytuje efektivní algoritmy pro převod mezi Eulerovými úhly a kvaterniony. Výsledný kvaternion je následně integrován do zprávy odesílané na řídící topic.

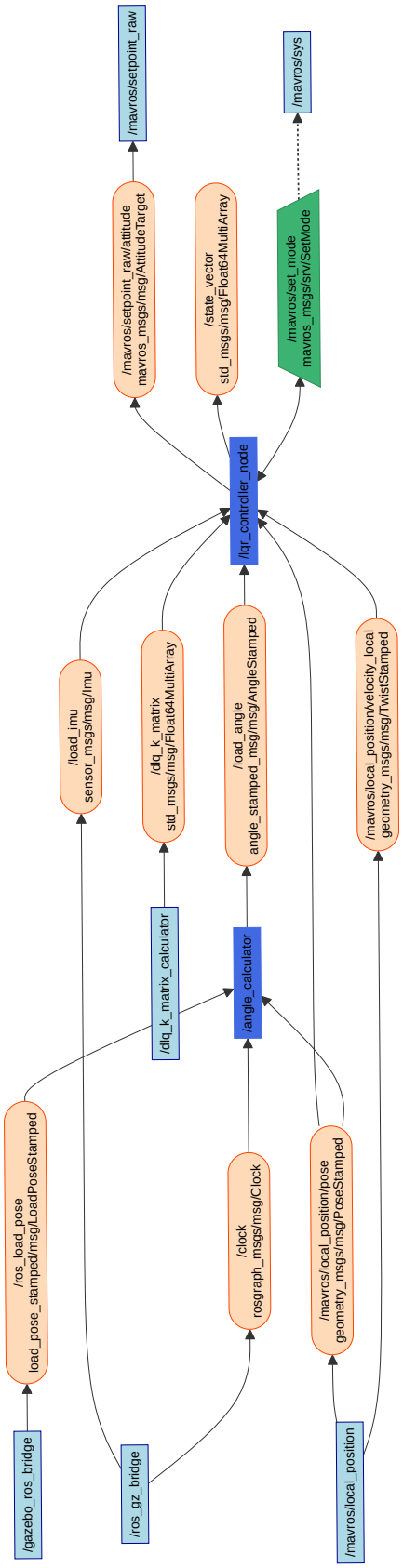
Jak již bylo zmíněno, tak řídící topic požaduje také tah dronu, který působí pouze v ose z, a to v intervalu <0,1>. Aby dron držel během letu stejnou výšku, tak je potřeba k tomu implementovat regulátor.

K tomuto účelu byla vytvořena třída *PIDController*. Ta reprezentuje objekt PID regulátoru se všemi potřebnými nastaveními parametrů. Regulovanou veličinou je zde tedy z souřadnice polohy dronu. Třída obsahuje funkci, která na základě aktuální regulační odchylky a nastavené saturace je schopna určit aktuální akční zásah.

V rámci řídícího uzlu `lqr_controller`, je inicializován objekt PID regulátoru. Parametry jsou optimalizovány pro udržení specifikované výšky dronu na hodnotě přibližně 10 metrů. Parametrizace regulátoru byla stanovena na základě experimentálních testů, kde výsledkem je $K_P = 1{,}2$, $K_I = 0{,}1$, $K_D = 0{,}45$. Výstupní hodnota PID regulátoru je poté přikládána jako součást zprávy `AttitudeTarget`, která se publikuje společně s kvaternionem orientace na řídící topic `/mavros/setpoint_raw/attitude`.

Řídící smyčka je implementována s frekvencí 30 Hz, jelikož vstupní data o dronu jsou poskytována právě s frekvencí 30 Hz.

Na obrázku 5.3 lze vidět výslednou integraci řídícího algoritmu do ROS2 architektury. Je zde znázorněna veškerá komunikace mezi uzly a zároveň i typy předávaných zpráv. Legendu k tomuto diagramu lze nalézt na obrázku 4.4. Tento diagram je klíčový pro pochopení zavedené architektury řídícího algoritmu.



Obrázek 5.3: Diagram architektury řídícího algoritmu

5.4 Algoritmus pro plánování trajektorie

Řídící algoritmus podporuje konfiguraci chování dronu pomocí parametrů zadaných v rámci *launch* souboru. Tyto parametry řídí, zda se dronu bude pohybovat lineárně k určenému cílovému bodu, tedy po přímce, nebo zda bude sledovat kruhovou trajektorii s definovaným poloměrem. Parametry jsou aplikovány při spuštění řídicího uzlu.

5.4.1 Lineární trajektorie

Pro lineární pohyb jsou řídícímu uzlu předány parametry, které definují cílovou pozici trajektorie. Implementace této trajektorie je založena na neustálém předkládání nových setpointů na základě vektorového rozdílu mezi aktuální a požadovanou pozicí $\Delta\boldsymbol{x}$, což lze vyjádřit jako

$$\Delta\boldsymbol{x} = \boldsymbol{x}_{\text{požadované}} - \boldsymbol{x}_{\text{aktuální}}, \tag{5.22}$$

kde $\boldsymbol{x}_{\text{požadované}}$ je vektor obsahující souřadnice následujícího setpointu a $\boldsymbol{x}_{\text{aktuální}}$ je vektor aktuálních souřadnic dronu.

Následuje normalizace vektorového rozdílu $\Delta\boldsymbol{x}$. Pokud je norma tohoto rozdílového vektoru větší než předem stanovená hodnota kroku k, tak dojde k jejímu přeškálování. Toto přeškálování je nastaveno tak, aby dron postupoval směrem k cílové pozici v konzistentních krocích. To zajišťuje postupnou konvergenci k cílové pozici. Normalizace má tvar

$$\Delta\boldsymbol{x} = k \cdot \frac{\Delta\boldsymbol{x}}{||\Delta\boldsymbol{x}||}. \tag{5.23}$$

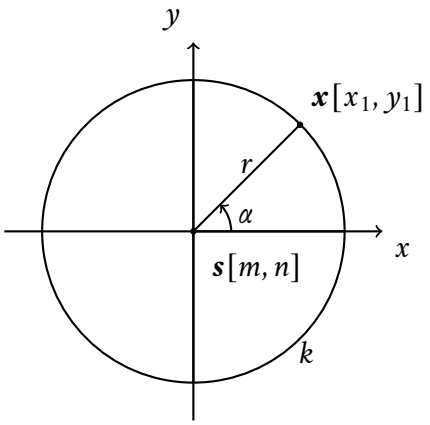
Výsledná změna pozice se v každém cyklu přičte k aktuální pozici, což lze vyjádřit vztahem

$$\boldsymbol{x}_{\text{aktuální}} = \boldsymbol{x}_{\text{aktuální}} + \Delta\boldsymbol{x}. \tag{5.24}$$

Tímto je zajištěno, že dron se plynule přesouvá k cílové pozici s maximálně definovanou povolenou rychlostí pohybu.

5.4.2 Kruhová trajektorie

Podobně jako tomu bylo v případě lineární trajektorie, tak i kruhová trajektorie je konfigurována pomocí parametrů, které jsou předány řídícímu uzlu v rámci *launch* souboru. Trajektorie po kružnici je odvozena z parametrického vyjádření kružnice, jak je ilustrováno na obrázku 5.4.



Obrázek 5.4: Jednotková kružnice

Libovolný bod $\boldsymbol{x}$ na kružnici k lze parametrizovat jako

$$\begin{aligned} x_1 &= m + r \cdot \cos\alpha, \\ y_1 &= n + r \cdot \sin\alpha, \\ \alpha &\in \langle 0; 2\pi). \end{aligned} \tag{5.25}$$

Je zřejmé, že pro určení specifického bodu na kružnici je potřeba znát úhel α a poloměr r. Generování po sobě jdoucích setpointů tedy probíhá tak, že úhel α je v každém řídícím cyklu inkrementován o pevný krok. Nastavením velikosti tohoto kroku lze regulovat rychlost pohybu dronu po kružnici.

Aby dron vždy na kružnici začínal, tak je nutné přepočítat střed kružnice a na základě polohy dronu určit počáteční bod na kružnici.

Pro následující výpočet se vychází z předpokladu, že střed opisované kružnice je dán jako

$$\boldsymbol{s} = \begin{bmatrix} m \\ n \end{bmatrix} = \begin{bmatrix} r \\ 0 \end{bmatrix}, \tag{5.26}$$

kde r je nastavený poloměr kružnice.
Aktuální polohu na kružnici lze odvodit z následujících rovnic.
Aktuální poloha dronu je dána jako

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.27}$$

Výpočet počáteční polohy na kružnici lze určit s využitím trigonometrie a to jako

$$\alpha_0 = atan2(y - m, x - n) = atan2(y - r, x - 0). \tag{5.28}$$

Aby bylo zajištěno, že dron bude od startu následovat správnou trajektorii, tak první setpoint je dán vztahem

$$\boldsymbol{x}_{\text{požadované}} = \boldsymbol{s} + \begin{bmatrix} r \cdot cos(\alpha_0) \\ r \cdot sin(\alpha_0) \end{bmatrix}. \tag{5.29}$$

5.4.3 Úprava orientace dronu

Jelikož řídící uzel počítá se setpointy, které jsou od sebe vzdáleny vždy o pevný krok, tak je nutné řídit i samotnou orientaci dronu tak, aby dron směřoval vždy směrem k dalšímu setpointu. Úhel rotace kolem osy z lze spočítat jako úhel mezi osou x a aktuálním setpointem.
Za předpokladu, že platí

$$\boldsymbol{x}_{\text{aktuální}} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}, \tag{5.30}$$

$$\boldsymbol{x}_{\text{cílové}} = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}. \tag{5.31}$$

Lze úhel požadovaného natočení dronu určit z trojúhelníku jako

$$\Delta x = x_2 - x_1, \tag{5.32}$$

$$\Delta y = y_2 - y_1, \tag{5.33}$$

$$\alpha = atan2(\Delta y, \Delta x). \tag{5.34}$$

Výsledný úhel je vždy normalizován předtím, než je použit v řídícím algoritmu.

Vyhodnocení řídícího algoritmu — 6

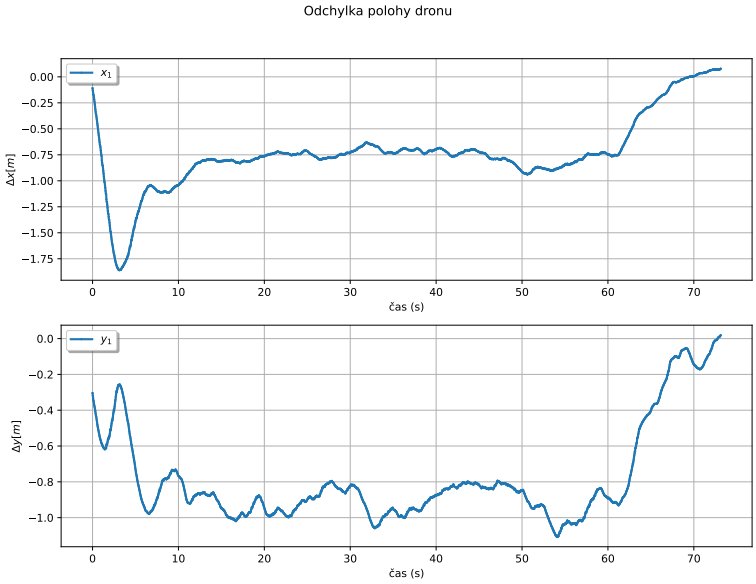
V této kapitole je detailně prostudována kvalita navrženého regulátoru a samotného řídícího systému. Efektivita a robustnost navrženého řídícího algoritmu je vyhodnocována na základě několika scénářů, které jsou navrženy tak, aby umožnily důkladné otestování.

Nejdříve je testována schopnost dronu sledovat lineární trajektorii se současným požadavkem na stabilizaci nákladu. Následně je testována schopnost sledovat složitější trajektorii ve tvaru kružnice. Tato trajektorie je mnohem komplexnější a je schopna otestovat i korektnost všech uvažovaných přepočtů mezi souřadnými systémy a obecně komplexnost navrženého algoritmu řízení. V průběhu této fáze testování je klíčová stabilizace nákladu při variabilitě trajektorie. Finální scénář je opět sledování kruhové trajektorie, ale je zde přidán šum měření. Tímto se otestuje robustnost navrženého řízení a jeho odolnost vůči chybám.

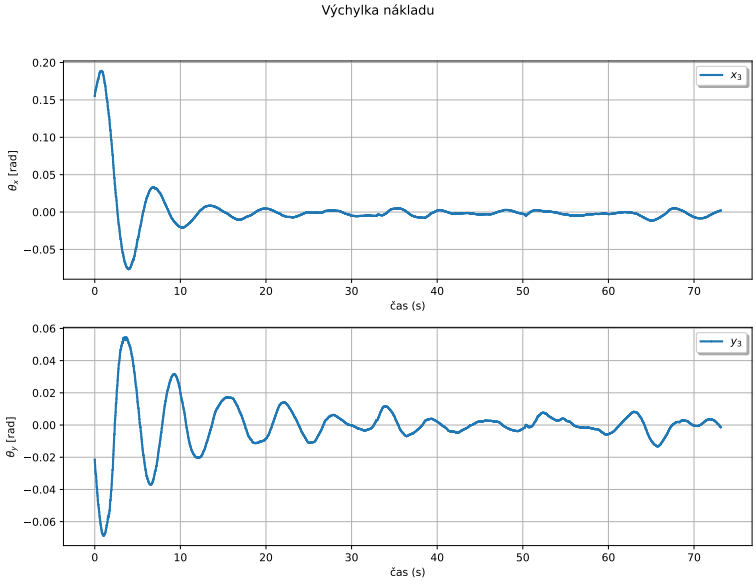
Výstupy každého z testů budou reprezentovány souborem grafů, které ilustrují časový vývoj regulační odchylky dronu, úhlové výchylky nákladu a dráhy letu. Tato vizualizace poskytne přehledné a komplexní informace pro hodnocení efektivity a adaptability řídícího algoritmu. Výsledky těchto testů budou klíčové pro demonstraci spolehlivosti a operativní robustnosti systému.

6.1 Lineární trajektorie

Tato část se zabývá experimentem, kde jsou ověřovány fundamentální schopnosti algoritmu řízení v režimu sledování lineární trajektorie se současnou stabilizací nákladu. V rámci experimentu je dron vyslán do předem definované lokace s cílem zachovat přesnost trajektorie ve shodě se zadáním setpointů, zatímco je prováděna stabilizace oscilací závěsu nákladu.

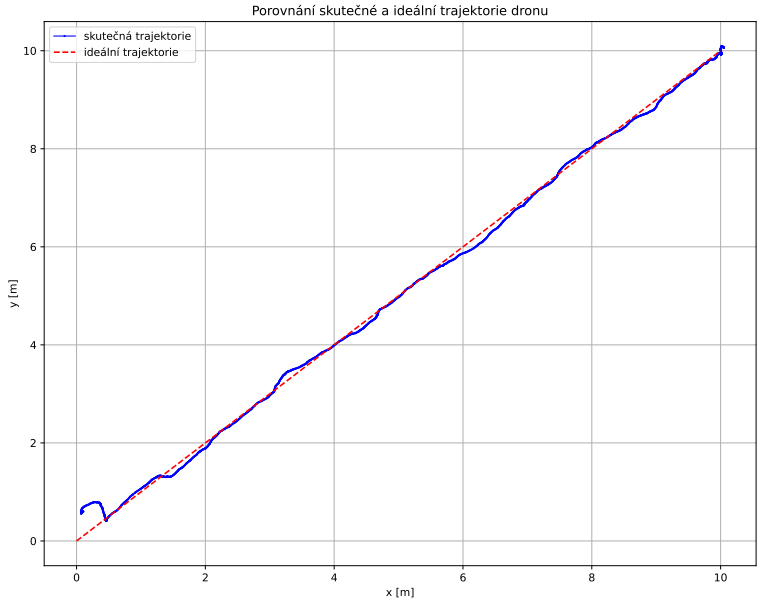


(a) Vývoj regulační odchylky polohy dronu v čase



(b) Vývoj výchylky nákladu v čase

Obrázek 6.1: Vývoj stavů systému v čase při sledování lineární trajektorie



Obrázek 6.2: Porovnání skutečné a ideální lineární trajektorie

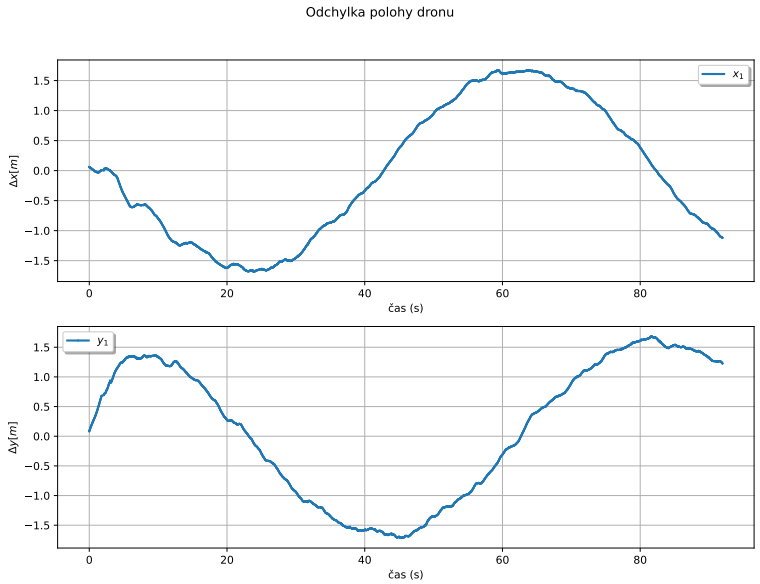
Tento experiment zkoumal především regulační odchylku polohy dronu od nastavené trajektorie a dynamiku vychýlení nákladu. Přestože počáteční poloha dronu nebyla na očekávané hodnotě $[0, 0]$ kvůli iniciální odchylce při vzletu, řídící algoritmus se rychle adaptoval. Výsledky ukazují, že regulační odchylka není konstantně nulová, což odráží zpoždění dronu ve sledování trajektorie, způsobené prioritizací stabilizace nákladu.

Na obrázku 6.1a je znázorněn časový průběh regulační odchylky, který ilustruje kompromis mezi sledováním trajektorie a stabilizací nákladu. Analýza vychýlení nákladu, jež je znázorněna na obrázku 6.1b, zobrazuje, že řídící algoritmus efektivně potlačuje amplitudu oscilací během prvních deseti sekund, což potvrzuje jeho schopnost rychle stabilizovat náklad.

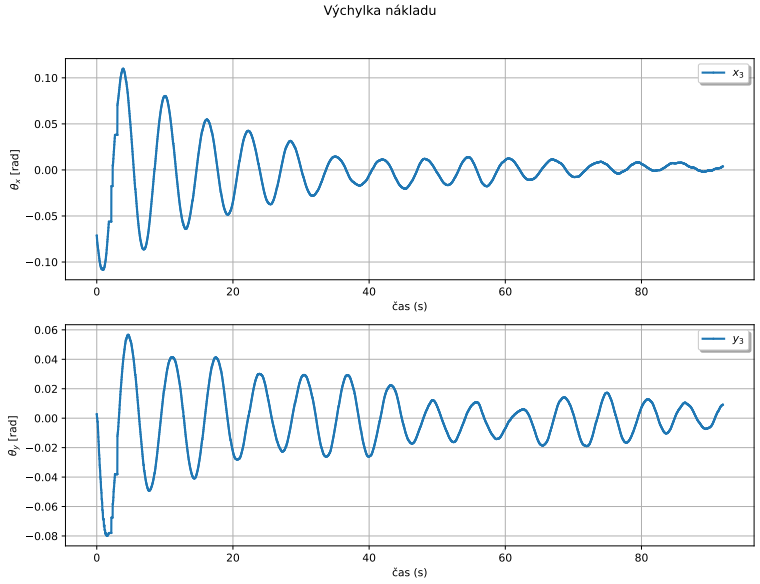
Schopnost dronu sledovat trajektorii s minimální odchylkou je zobrazena na obrázku 6.2. Počáteční větší odchylka, která postupně klesá, je důsledkem počátečních snah řídícího algoritmu o stabilizaci nákladu.

6.2 Kruhová trajektorie

Tato sekce se zabývá testováním schopnosti řídícího algoritmu sledovat složitější typ trajektorie, tedy kružnici, a schopnost aktivně stabilizovat náklad. V tomto experimentu byl sledován opět vývoj regulační odchylky polohy dronu a zároveň schopnost dronu stabilizovat náklad.

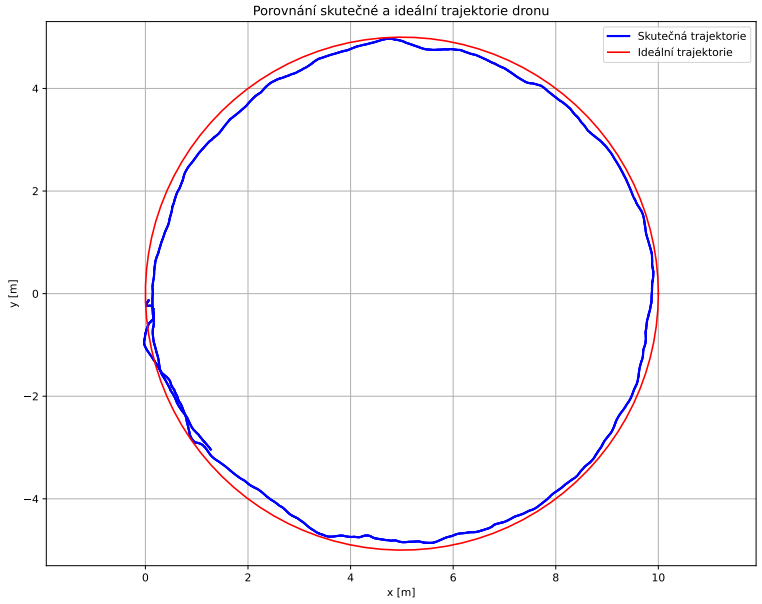


(a) Vývoj regulační odchylky polohy dronu v čase



(b) Vývoj výchylky nákladu v čase

Obrázek 6.3: Vývoj stavů systému v čase při sledování kruhové trajektorie



Obrázek 6.4: Porovnání skutečné a ideální kruhové trajektorie

Regulační odchylka není konstantně nulová kvůli prioritizaci stabilizace nákladu nad sledováním trajektorie, což je ilustrováno na obrázku 6.3a. Tato prioritizace způsobuje mírné zpoždění za trajektorií, což je podobný stav jako u lineární trajektorie 6.1.

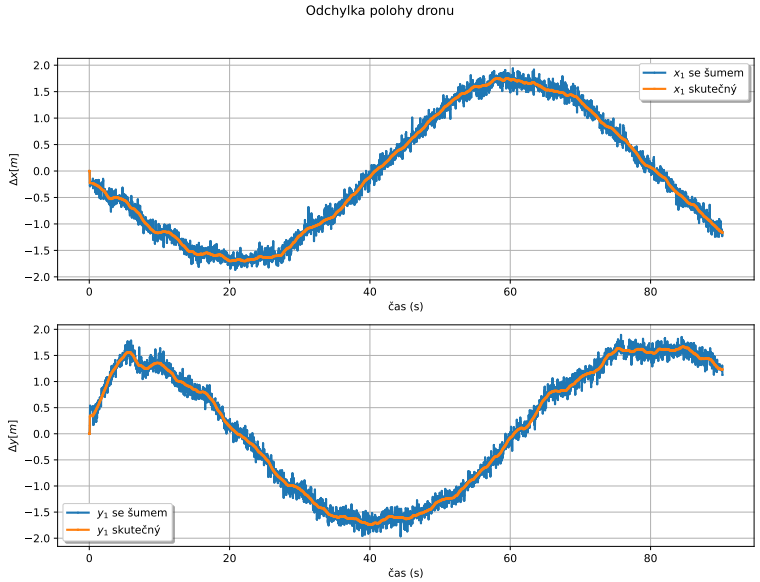
Stabilizace nákladu je rychlá a efektivní, což ukazuje obrázek 6.3b, kde dron stabilizuje náklad z počáteční výchylky 6 stupňů na maximální amplitudu ±1,5 stupně do 20 sekund. Tato účinná regulace je založena na kompromisu mezi rychlostí odezvy a potlačením šumu, který by mohl být významně zesílen, pokud by byly požadavky na rychlejší regulaci vyšší. Tato problematika je podrobněji rozpracována v sekci 6.3.

Obrázek 6.4 zobrazuje schopnost dronu sledovat kruhovou trajektorii s minimálními odchylkami, přičemž největší odchylky jsou zaznamenány na počátku trajektorie, kde řídící algoritmus upřednostňuje stabilizaci nákladu.

Závěrem, dron úspěšně demonstruje schopnost sledovat kruhovou trajektorii a efektivně akivně tlumit kmity nákladu, což současně potvrzuje správné návrhové předpoklady řídícího algoritmu pro prostorový pohyb.

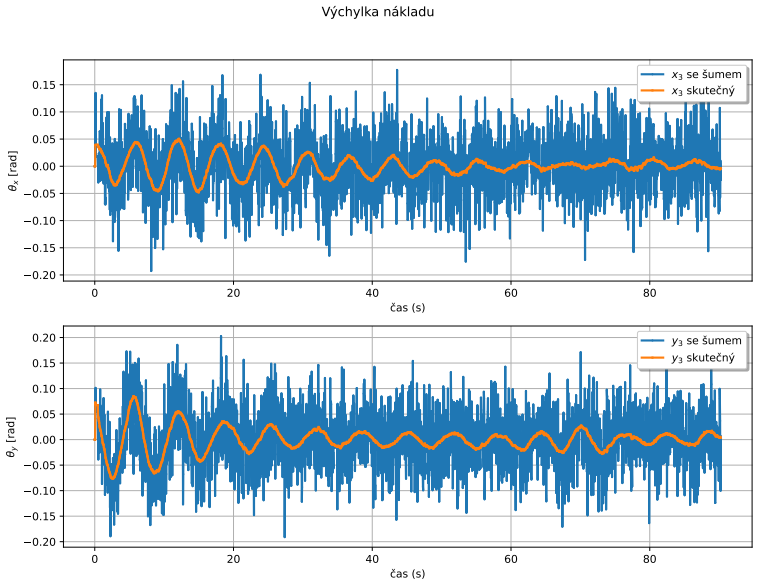
6.3 Analýza odolnosti řídícího algoritmu na senzorický šum

Tato sekce je věnována testování řídícího algoritmu na odolnost vůči chybám. Experiment je tedy prováděn tak, že je na jednotlivé stavy přidán šum stejným způsobem, jako tomu bylo u planárního modelu 3.3.2. Dron má za cíl sledovat kruhovou trajektorii a současně stabilizovat náklad tak, aby nedocházelo k nežádoucím oscilacím. Nastavení samotného šumu je uvažováno identické, jako tomu bylo u planárního modelu 3.3.2. Znázorněna je pouze regulační odchylka polohy dronu, výchylka nákladu a trajektorie letu dronu.



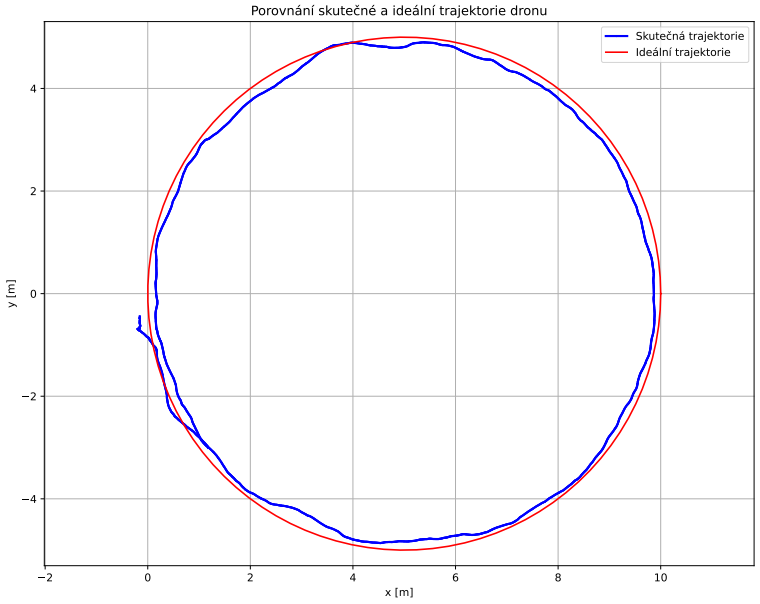
Obrázek 6.5: Vývoj regulační odchylky polohy dronu s vlivem šumu měření

Vliv šumu na regulační odchylku polohy dronu není nijak signifikantní, jak je znázorněno na obrázku 6.5. To je způsobeno tím, že uvažované zesílení v matici $\boldsymbol{K}$ je nízké, a tedy šum není výrazně zesilován.



Obrázek 6.6: Vývoj výchylky nákladu s vlivem šumu měření

Na druhou stranu regulátor je nastaven tak, aby prioritizoval stabilizaci nákladu před přesným sledováním trajektorie, což způsobuje významné zesílení šumu v této oblasti, jak je ilustrováno na obrázku 6.6. Přes značný vliv šumu, který může v extrémních případech dosahovat hodnot daleko převyšujících reálnou výchylku, tak je algoritmus schopen potlačit oscilace a udržet maximální amplitudu oscilací do rozmezí ±3 stupňů, což lze považovat za naprosto zanedbatelné oscilace. To poukazuje na vysokou míru robustnosti algoritmu, což je nezbytné pro zajištění stabilního chování systému. Porovnáním tohoto případu se situací, kdy šum nebyl uvažován, jak je znázorněno na obrázku 6.3b, je zřejmé, že šum negativně ovlivňuje dobu stabilizace nákladu spolu s amplitudou ustálených kmitů.



Obrázek 6.7: Porovnání skutečné a ideální kruhové trajektorie s aditivním šumem

Obrázek 6.7 ilustruje vliv šumu na schopnost dronu sledovat trajektorii, zejména na počátku letu, kdy bylo zaznamenáno největší vychýlení z počáteční pozice. Dron však přesto úspěšně sledoval trajektorii s minimálními odchylkami během zbytku letu. Závěrem tohoto experimentu lze shrnout, že ačkoliv byl vliv šumu velmi výrazný, především na výchylce nákladu, tak byl dron schopen náklad stabilizovat a potlačit kmity. Také byl velmi úspěšný ve sledování požadované trajektorie, což tedy značí, že navržený řídící algoritmus je odolný vůči chybám a je schopen jejich účinné kompenzace. Tato schopnost je zásadní pro potlačení nežádoucích dynamických odezev, které mohou vyplývat z nepřesnosti modelu či vlivu prostředí.

Závěr 7

Tato práce se zaměřila na návrh řídícího systému pro dron s nákladem zavěšeným volně pod dronem. Hlavním cílem bylo vyvinout robustní řídící algoritmus schopný efektivně řešit výzvy spojené s dynamikou a stabilizací přidaného nákladu.

Byla důkladně prostudována problematika, která úzce souvisí s tímto tématem a následně byl navržen matematický model této soustavy. Na tento matematický model byl navržen regulátor, konkrétně tedy LQ regulátor, který byl testován nejdříve přímo na matematickém modelu. Výsledkem tohoto návrhu vzešel velmi robustní regulátor, který je odolný vůči chybám v modelu a z teoretického hlediska je tak naprosto vhodný na nasazení na reálný systém.

Následně byl tento matematický návrh převeden na řídící algoritmus, který byl následně implementován přímo na řízený systém v simulačním prostředí, které bylo speciálně navrženo tak, aby věrně simulovalo reálné podmínky. Toto simulační prostředí zahrnuje detailní reprezentace fyzikálních charakteristik dronu, jako jsou hmotnost, aerodynamické vlastnosti a chování motorů. Díky tomu poskytuje cenné předpoklady pro ověření algoritmu před jeho nasazením na reálný systém, což je v této oblasti velmi zásadní.

Po implementaci řídícího systému byla pozornost zaměřena především na nejrůznější testování tohoto řídícího algoritmu. Bylo testováno sledování lineární a kruhové trajektorie, kde byla testována především schopnost řídícího algoritmu aktivně tlumit oscilace nákladu. Toto testování proběhlo velmi úspěšně a regulátor byl schopen splnit očekávané požadavky. Poslední fází testování bylo přidání šumu k jednotlivým stavům a tím testovat robustnost navrženého řízení. Závěrem tohoto experimentu je, že navržený algoritmus byl schopen aktivně tlumit oscilace nákladu i přes velmi významný vliv šumu.

Jako potenciální vylepšení tohoto řídícího algoritmu by bylo vhodné jej rozšířit pro uvažování nejen samotného letu, ale také uvažování vzletu či přistání, což je také velmi složitá problematika, která byla naznačena v úvodu této práce. Dalším krokem ke zvýšení efektivity algoritmu by mohla být implementace uzlu umožňujícího úpravu parametrů regulátoru v reálném čase. Tento uzel by odhadoval hmotnost nákladu a na základě tohoto odhadu by dynamicky upravoval parametry regulátoru.

Bibliografie

[Erm18] ERMAKOV, Vladimir. *MAVROS* [online]. 2018. [cit. 2024-03-13]. Dostupné z: http://wiki.ros.org/mavros.

[Est+24] ESTEVEZ, Julian; GARATE, Gorka; LOPEZ-GUEDE, Jose Manuel; LARREA, Mikel. Review of Aerial Transportation of Suspended-Cable Payloads with Quadrotors. *Drones*. 2024, roč. 8, č. 2. ISSN 2504-446X. Dostupné z DOI: 10.3390/drones8020035.

[24a] *Gazebo Garden* [online]. Open Robotics, 2024. [cit. 2024-05-13]. Dostupné z: https://gazebosim.org/docs/garden/getstarted.

[GM19] GOUBEJ, Martin; MELICHAR, Jiří. *Lineární systémy 2*. 2019. Učební text. Naposledy navštíveno 3.2. 2024.

[24b] *Holybro X500 V2 Kit* [online]. PX4 Autopilot User Documentation, 2024. [cit. 2024-04-13]. Dostupné z: https://docs.px4.io/main/en/frames_multicopter/holybro_x500V2_pixhawk5x.html.

[KFJ17] KLAUSEN, Kristian; FOSSEN, Thor I.; JOHANSEN, Tor Arne. Nonlinear Control with Swing Damping of a Multirotor UAV with Suspended Load. *Journal of Intelligent Robotic Systems*. 2017, roč. 88, č. 2, s. 379–394. Dostupné z DOI: 10.1007/s10846-017-0509-6.

[Mac+22] MACENSKI, Steven; FOOTE, Tully; GERKEY, Brian; LALANCETTE, Chris; WOODALL, William. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*. 2022, roč. 7, č. 66, eabm6074. Dostupné z DOI: 10.1126/scirobotics.abm6074.

[24c] *PX4 Autopilot User Documentation* [online]. PX4 Development Team, 2024. [cit. 2024-03-13]. Dostupné z: https://docs.px4.io/main/en/.

[SSL24] SAUNDERS, Jack; SAEEDI, Sajad; LI, Wenbin. Autonomous aerial robotics for package delivery: A technical review. *Journal of Field Robotics*. 2024, roč. 41, č. 1, s. 3–49. Dostupné z DOI: https://doi.org/10.1002/rob.22231.

[Sch16] SCHLEGEL, Miloš. *Systémy a modely*. 2016. Učební text. Naposledy navštíveno 10.1. 2024.

[VBS20] VILLA, Daniel K. D.; BRANDÃO, Alexandre S.; SARCINELLI-FILHO, Mário. A Survey on Load Transportation Using Multirotor UAVs. *Journal of Intelligent & Robotic Systems*. 2020, roč. 98, č. 2, s. 267–296. ISSN 1573-0409. Dostupné z DOI: 10.1007/s10846-019-01088-w.

Seznam obrázků