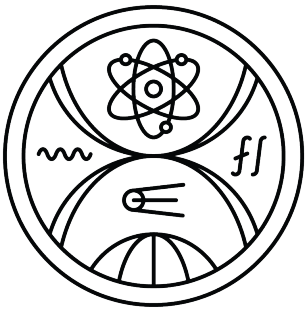
Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

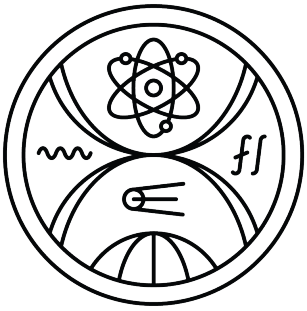


ADAPTÍVNE RIADENIE KROKU ŠTVORNOHÉHO ROBOTA

Diplomová práca

2024 Bc. Zuzana Mačicová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky



ADAPTÍVNE RIADENIE KROKU ŠTVORNOHÉHO ROBOTA

Diplomová práca

Študijný program:	aplikovaná informatika
Študijný odbor:	informatika
Školiace pracovisko:	Katedra aplikovanej informatiky
Školiteľ:	prof. RNDr. Roman Ďurikovič, PhD.
Konzultant:	Mgr. Rastislav Marko

Bratislava, 2024 Bc. Zuzana Mačicová

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Zuzana Mačicová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Adaptívne riadenie kroku štvornohého robota
Adaptive step control of a quadruped robot

Anotácia: Adaptívne riadenie je v súčasnosti jedným z najpopulárnejších prístupov pre ovládanie
kroku štvornohých robotov. V porovnaní s konvenčnejšími metódami dokáže generovať vysoko
dynamické pohyby a zároveň zabezpečiť stabilitu a robustnosť kroku. Napriek neustále zvyšujúcemu sa výkonu mikropočítačov a vývoju stále efektívnejších optimalizačných riešičov je však problém kráčania stále komplexný a jeho úspešné formulácie sa spoliehajú na mnohé zjednodušenia a zanedbania skutočných vlastností riadeného systému. Práve efektívne formulácie zohľadňujúce čo možno najviac relevantných vplyvov na krok sú v súčasnosti predmetom výskumu.

Cieľ: Cieľom práce je analyzovať a navrhnúť model adaptívneho riadenia pre štvornohého robota operujúceho v zložitom prostredí. Táto téma bude vypracovaná v spolupráci s firmou Panza Robotics s.r.o.

1. Naštudujte problematiku riadenia kroku kráčajúcich robotov.
2. Analyzujte možnosti matematických formulácii problému optimálneho riadenia kroku.
3. Navrhnite a implementujte adaptívne riadenie pre štvornohého robota.
4. Overte navrhnuté riešenie pri rôznych podmienkach v simulačnom prostredí.

Literatúra:
Ding, Yanran & Pandala, Abhishek & Park, Hae-Won. (2019). Real-time Model Predictive
Control for Versatile Dynamic Motions in Quadrupedal Robots. 10.1109/ICRA.2019.8793669.
Grandia, R., Taylor, A. J., Ames, A. D., & Hutter, M. (2021, May). Multi-layered safety for
legged robots via control barrier functions and model predictive control. In 2021 IEEE
International Conference on Robotics and Automation (ICRA) (pp. 8352-8358). IEEE.
Carlo, Jared & Wensing, Patrick & Katz, Benjamin & Bledt, Gerardo & Kim, Sangbae. (2018).
Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control. 1-9.

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

10.1109/IROS.2018.8594448.

Vedúci: prof. RNDr. Roman Ďurikovič, PhD.
Konzultant: Mgr. Rastislav Marko
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 29.11.2022

Dátum schválenia: 29.11.2022 prof. RNDr. Roman Ďurikovič, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Čestne prehlasujem, že túto diplomovú prácu som vypracovala samostatne len s použitím uvedenej literatúry a za pomoci konzultácií u môjho školiteľa a konzultanta.

. . . . . .

Bratislava, 2024 Bc. Zuzana Mačicová

Poďakovanie

Ďakujem mojej rodine, predovšetkým rodičom a starým rodičom, ktorí ma podporovali počas celého štúdia a môjmu školiteľovi prof. RNDr. Romanovi Ďurikovičovi, PhD. za odborné vedenie a konzultantovi Mgr. Rastislavovi Markovi.

Abstrakt

Artaban je štvornohý robot, so špecifickou stavbou nôh. Jeho zadné nohy sú zložitejšie v porovnaní s inými štvornohými robotmi, ktorí majú väčšinou všetky štyri nohy rovnaké, skladajúce sa z dvoch segmentov. V tejto práci sa venujeme adaptívnemu riadeniu takéhoto robota za pomoci knižnice Quad SDK. Odvodili sme inverznú dynamiku, implementovali riadenie a nakoniec sme riadenie otestovali aj v simulácií aj na hardvéri a porovnali s inými typmi riadenia. Vyskúšali sme rôzne typy pohybov - vpred, vzad, otáčanie sa a kráčanie vbok. Taktiež sme overili schopnosť robota zvládnuť neočakávané postrčenia a udržať si rovnováhu.

Kľúčové slová: štvornohý robot, inverzná dynamika, adaptívne riadenie

Abstract

Artaban is a quadruped robot with a specific leg structure. Its rear legs are more complex compared to other quadruped robots, which typically have all four legs identical and consisting of two segments. In this work, we focus on the adaptive control of such a robot using the Quad SDK library. We derived inverse dynamics, implemented control, and finally tested the control both in simulation and on hardware, comparing it with other types of control. We explored different types of movements: forward, backward, turning, and side-stepping. We also verified the robot's ability to handle unexpected pushes and maintain balance.

Keywords: quadruped robot, inverse dynamics, adaptive control

Obsah

Zoznam obrázkov

Zoznam tabuliek

Použitá terminológia a skratky

Terminológia

- **Vysoko úrovňové riadenie (High-level control)**
  Časť riadiaceho procesu robota, ktorej výsledky sa neposielajú hardvéru ale nižším vrstvám riadenia.
- **Nízko úrovňové riadenie (Low-level control)**
  Časť riadiaceho procesu robota, ktorá na základe príkazov z vyššej vrstvy posiela konkrétne príkazy hardvéru.
- **Model Predictive Control**
  Metóda používaná na výpočet optimálnej akcie pri splnení daných podmienok a minimalizovaní cenovej funkcie.
- **Quad SDK**
  Softvérový rámec pre riadenie robotov.
- **Gazebo**
  Fyzikálny simulátor pre testovanie robotov.

Skratky

- **CROCODDYL** - Contact RObot COntrol by Differential DYnamic Library.
- **MIT** - Massachusetts Institute of Technology.
- **MPC** - Model Predictive Control.
- **ROS** - Robotic Operating System.
- **RBDL** - Rigid Body Dynamics Library.
- **URDF** - Unified Robot Description Format.
- **SDF** - Simulation Description Format.
- **XML** - eXtensible Markup Language.
- **FD** - Priama dynamika (Forward dynamics).

- **ID** - Inverzná dynamika (Inverse dynamics).
- **Ipopt** - Interior Point Optimizer.
- **PID regulátor** - Proporcionálny, integračný a derivačný regulátor.
- **PD regulátor** - Proporcionálny a derivačný regulátor.

Úvod

Oblasť robotiky v posledných rokoch zažíva veľký rozmach. Či už v oblasti priemyselnej výroby, zdravotníctva, pri prieskume oblastí nebezpečných pre človeka alebo v domácnosti. Veľa z týchto robotov vykonáva monotónne úlohy, avšak pri kráčajúcich robotoch sú ich úlohy menej opakujúce sa a je nevyhnutnosťou aby sa robot vedel prispôsobiť rôznym situáciám, v ktorých sa môže ocitnúť. Tu vzniká potreba adaptívneho riadenia takýchto robotov.

Pri chodiacich robotoch sa venuje veľa pozornosti hlavne štvornohým robotom, ktorých výhodou je napríklad väčšia stabilita oproti dvojnohým robotom. Tieto roboty umožňujú pohyb po členitom teréne, či už sú to schody vo výrobnej hale, nerovný terén v lese alebo prekážka ktorú je potrebné prekročiť na inak rovnej ploche. Pre roboty na pásoch či kolesách je takáto prekážka neprekonateľná aj keď na rovnom povrchu majú veľa výhod ako nenáročné riadenie a jednoduchšiu konštrukciu.

Riadenie štvornohých robotov zahŕňa veľa aspektov. Najprv musí robot poznať svoje okolie. To zahŕňa spracovanie signálu z kamery alebo kamier, kedy si vytvorí vnútornú reprezentáciu svojho okolia [21] [4] [13]. Ukazuje sa, že roboty, ktorých riadiaci proces nezahŕňa okolité prostredie, nezvládajú chôdzu v teréne kde sú napríklad diery v podložke [27]. Ďalej musí poznať svoj stav - kde sa nachádza v rámci prostredia, ako je natočený a kde sa nachádzajú jeho nohy. To typicky vieme vypočítať pomocou priamej kinematiky z natočenia motorov, ktoré získame zo senzorov. Niektoré roboty majú aj senzor v spodnej časti nohy, ktorá sa dotýka podložky, či je noha v kontakte alebo nie [14] [12]. Taktiež musí robot vedieť ktorým smerom má ísť. Tento údaj môže pochádzať z už dopredu naplánovanej cesty alebo priamo ako príkaz od používateľa, ktorý robota ovláda.

Na základe všetkých týchto dát sa počíta aký pohyb má robot vykonávať, napríklad či má ísť dopredu, otáčať sa a akou rýchlosťou. Ďalej sa vypočítajú trajektórie nôh, prípadne sily, ktorými majú jednotlivé nohy pôsobiť. Z nich sa väčšinou vypočítajú krútiace momenty na motoroch a tie sa pošlú do ovládačov jednotlivých motorov. Celý proces riadenia musí počítať aj s ďalšími obmedzeniami ako sú napríklad rozsahy nôh a to aby reakčné sily koncových častí nôh sa nachádzali v kuželi trenia a predišlo sa šmýkaniu nôh robota [6].

V tejto práci sa venujeme hlavne riadeniu robota Artabana, štvornohého robota,

vyvíjaného firmou Panza Robotics, s.r.o. Tak ako aj pri ostatných moderných robotoch je kladený dôraz na to, aby sa robot vedel prispôsobiť terénu a zvládnuť rôzne rozruchy z prostredia, ako napríklad, že doň niekto strčí. Tento robot sa však od väčšiny štvornohých robotov líši stavbou svojich nôh, čo spôsobuje komplikovanejší spôsob riadenia. Jeho nohy obsahujú kinematické slučky a taktiež ich hmotnosť tvorí viac ako 30% celkovej hmotnosti robota, čo je výrazne viac ako pri iných robotoch.

Kapitola 1

Prehľad problematiky

1.1 Riadenie u iných robotov

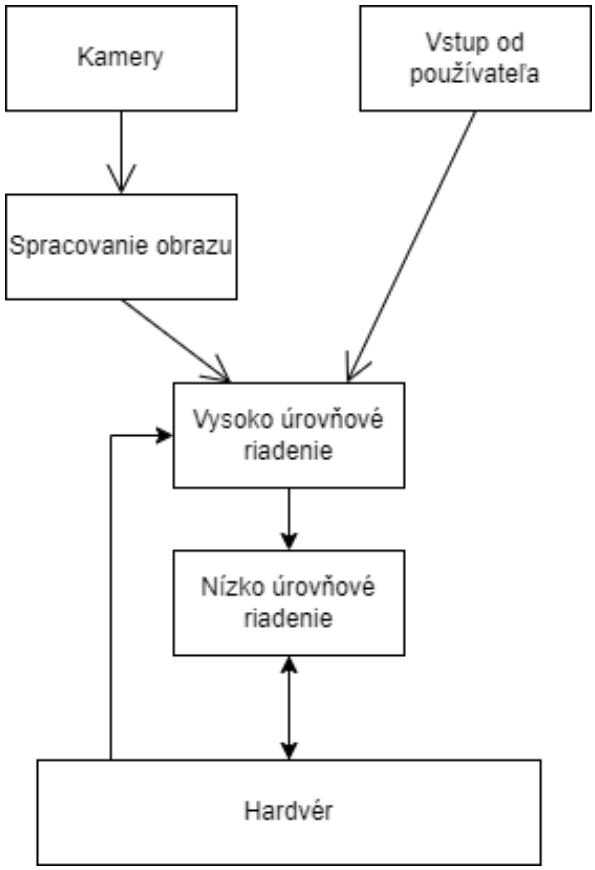
Na Obr. 1.1 vidíme zovšeobecnený postup riadenia robotov. Tento diagram bol vytvorený na základe viacerých článkov [12] [27] [21] a predstavuje zovšeobecnene riadiaceho procesu štvornohých robotov. Jednotlivé časti tohto procesu bežia s frekvenciou približne od 10Hz pre spracovanie prostredia až po 400Hz pre komunikáciu s hardvérom. Je nevyhnuté aby bol proces riadenia dostatočne rýchly, aby vedel reagovať na zmeny a zároveň efektívny aby mohol bežať priamo na počítači v tele robota.

Takéto adaptívne riadenie je nevyhnutnosťou pri robotoch s nohami, keďže ich okolie sa môže neustále meniť. Napríklad sa na naplánovanej ceste zjaví prekážka alebo je miesto kam robot stúpil nestabilné.

Ďalším problémom pri riadení robotov sú rozdiely medzi modelom v simulácií a skutočným hardvérom. Hardvérové súčiastky môžu mať nepresné rozmery, inú hmotnosť, opotrebením sa môžu deformovať, napríklad teplom pri používaní prevodov a motorov, prípadne nie sú dostatočné pevné a ohýbajú sa. V systéme môžu vznikať rôzne trenia, s ktorými sme nepočítali, robot môže mať nedostatočný príkon a iné problémy. Toto spôsobuje potrebu validácie riadenia aj na hardvéri a často môžeme dostať iné výsledky ako v simulácií.

Tieto rozdiely sú zodpovedné aj za náročný a dlhý proces ladenia a hľadania chýb, keďže je náročné lokalizovať, kde sa daná chyba nachádza. Taktiež do výslednej chôdze vstupuje veľa parametrov, ako dĺžka a výška kroku, výška tela pri chôdzi a rýchlosť chôdze, ktoré je potrebné optimálne nastaviť. Väčšinou tieto parametre vieme nastaviť iba empiricky, skúšaním rôznych kombinácií.

Dôležitý je aj typ chôdze. Štvornohé roboty majú inšpiráciu v prírode vo štvornohých zvieratách, to znamená, že aj typ chôdze je im podobný. Typ chôdze vie ovplyvniť plynulosť kroku robota, ale aj stabilitu či rýchlosť. Poznáme niekoľko typov chôdze. Medzi najznámejšie patrí napríklad chôdza krokom, klus a cval.

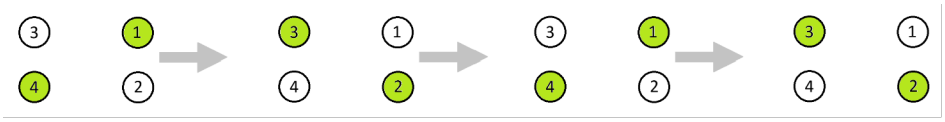


Obr. 1.1: Všeobecný postup riadenia robotov.

Pri chôdzi krokom, viď. Obr. 1.2, má robot vždy aspoň 3 nohy na podložke, tento typ chôdze ja najstabilnejší ale aj väčšinou pomalší oproti zvyšným. Pri kluse, viď. Obr. 1.3 , sa striedajú vždy dve a dve protiľahlé nohy v kontakte s podložkou. Medzi vystriedaním nôh môže byť ešte fáza letu, kedy nie je žiadna noha v kontakte s podložkou alebo fáza, kedy robota stabilizujeme, kedy sú 3 alebo 4 nohy v kontakte s podložkou. Tento typ chôdze je veľmi populárny pri štvornohých robotoch. Cval je v prírode typicky najrýchlejší typ chôdze. Tesne po sebe sa striedajú predné dve a zadné nohy dve nohy, viď Obr. 1.4. Pri testovaní kroku na robotovi sme sa venovali hlavne chôdzi a klusu.

Obr. 1.2: Typ kroku štvornohých zvierat - chôdza krokom. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Nohy v kontakte sú vyznačené zelenou farbou, bielou je vyznačená noha, ktorá sa podložky nedotýka. Značenie nôh je nasledovné: 1 - ľavá predná, 2 - pravá predná, 3 - ľavá zadná, 4 - pravá zadná.

Existuje viacero štvornohých robotov, ktoré sú schopné stabilnej chôdze. Niektoré sú vyvíjané univerzitami a sú verejne dostupné spôsoby a metódy akými sú riadené,



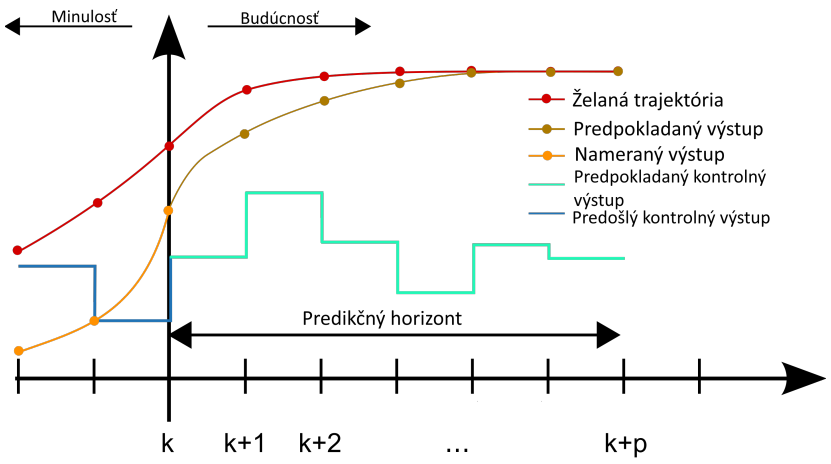
Obr. 1.3: Typ kroku štvornohých zvierat - klus. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na Obr. 1.2.

Obr. 1.4: Typ kroku štvornohých zvierat - cval. Postupnosť v akej sú jednotlivé nohy v kontakte s podložkou. Značenie je rovnaké ako na Obr. 1.2.

veľa robotov je však vyvíjaných súkromnými firmami, ktoré svoj výskum v tejto oblasti nepublikujú.

Pravdepodobne najznámejším štvornohých robotom je Spot od Boston Dynamics [1]. Z videí je vidno, že tento robot má dobrú stabilitu a vyladenú chôdzu a zvláda aj zložitý terén ako napríklad schody, avšak nie sú verejne dostupné podrobnejšie informácie o tom, ako funguje jeho riadenie.

Princípy riadenia niektorých iných štvornohých robotov, ako napríklad ANYmal a Cheetah, však vererejné sú. Základom pre väčšinu je MPC [6], [7], [17]. Vo všeobecnosti MPC je metóda optimálneho riadenia, kedy sa počíta optimálna akcia, ktorá minimalizuje cenovú funkciu za splnenia vopred zadefinovaných podmienok pre konečný časový horizont danej veľkosti. Zároveň však prijíma spätnú väzbu zo sytému a plán prispôsobuje [29]. Schému MPC môžeme vidieť na Obr. 1.5.



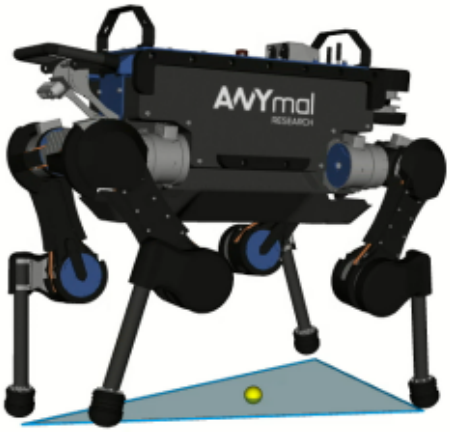
Obr. 1.5: Schéma MPC. Na obrázku vidíme ako v danom časovom horizonte sa mení kontrolný signál na základe nameraného stavu a predikovaného stavu .

V robotike sa MPC využíva hlavne na plánovanie cesty, ktorá je pre robota schodná. Výstupom môžu byť priamo krútiace momenty motorov, trajektórie nôh, trajektória tela, sily akými majú jednotlivé nohy pôsobiť alebo kombinácia z už spomenutých. Do

MPC väčšinou vstupuje vhodná reprezentácia prostredia, v ktorom sa robot pohybuje, cieľ robota alebo smer akým má ísť a jeho aktuálny stav. Pod stavom robota väčšinou rozumieme polohu a natočenie tela, natočenie motorov, prípadne rýchlosť motorov. Vstupom môže byť aj informácia zo senzorov v koncovej časti nohy, či je noha robota na podložke alebo nie, takýto senzor však nie každý robot má.

Ďalej si rozoberieme niekoľko robotov a spôsoby ich riadenia, keďže každý robot má iné senzory a stavbu tela, existujú aj rôzne prístupy k ich riadeniu.

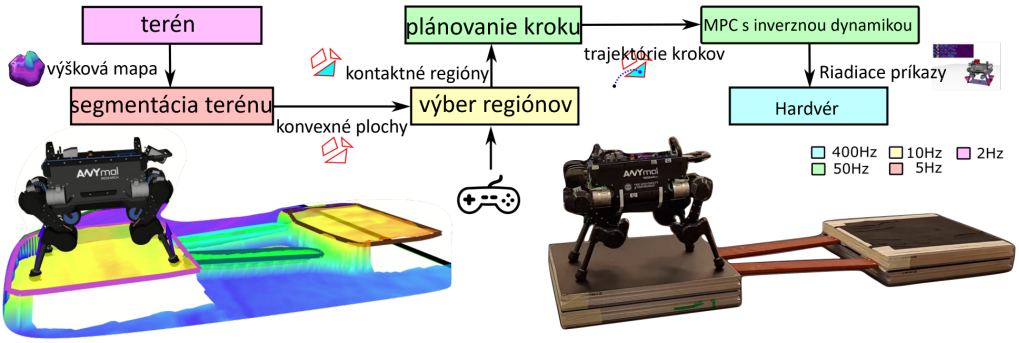
1.1.1 ANYmal



Obr. 1.6: ANYmal [4].

Tento štvornohý robot je vyvíjaný firmou ANYbiotics, Obr. 1.6. Vie sa pohybovať aj v zložitejšom teréne ako sú schody alebo terén, kde sú diery v podložke a rôzne iné nerovnosti [21].

Na Obr. 1.7 je znázornený proces jeho riadenia.



Obr. 1.7: Riadiaci proces ANYmal. Schéma procesu riadenia robota ANYmal [21].

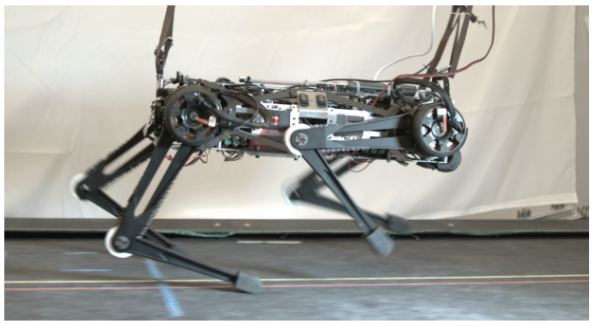
Najprv sa spracuje vstup z kamier, prípadne iných senzorov, v tele robota. Vytvorí sa mapa terénu a z nej sa vypočítajú konvexné plochy, na ktoré môže robot bezpečne

stúpiť. Bezpečne stúpiť pre robota znamená, že daná plocha nie je príliš naklonená a zároveň je dostatočne veľká pre koncovú časť nohy. Tieto dáta sa pošlú ďalej do MPC, spolu s pokynom od používateľa ktorým smerom má robot ísť a jeho aktuálnym stavom (uhly motorov, natočenie tela,...). MPC vypočíta optimálnu trajektóriu ťažiska a trajektórie nôh, ktoré nie sú v kolízií so žiadnym okolitým objektom [21].

MPC využíva knižnicu CROCODDYL. Je to knižnica pre optimálne riadenie robotov, ktorú je možné použiť nielen pre štvornohé roboty [20]. Slúži pre efektívne riadenie robotov pri viacerých bodoch kontaktu robota s prostredím, napríklad pri štvornohých robotoch je viacero nôh v kontakte s podložkou. Výhodou tejto knižnice je jej verejná dostupnosť a dokumentácia.

Riadiaci softvér ANYmal-a je kompatibilný s ROS, avšak nieje zverejnený v plnom rozsahu v jednotnom framework-u. Sú zverejné iba časti riadenia v podobe niektorých algoritmov [9] [23] [8].

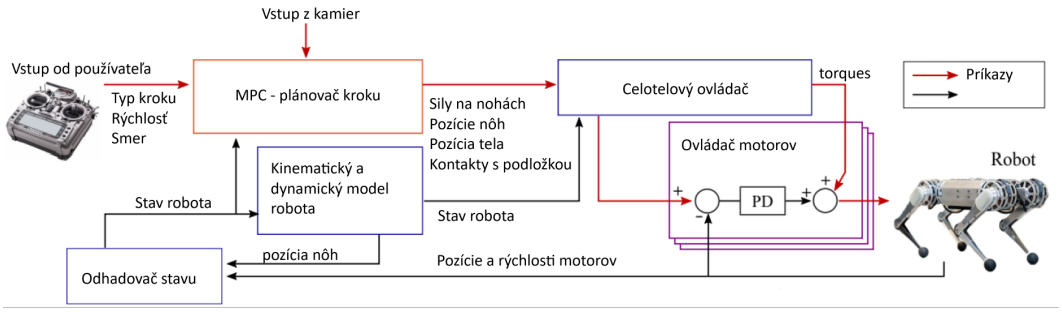
1.1.2 Cheetah



Obr. 1.8: MIT Cheetah 3 [6].

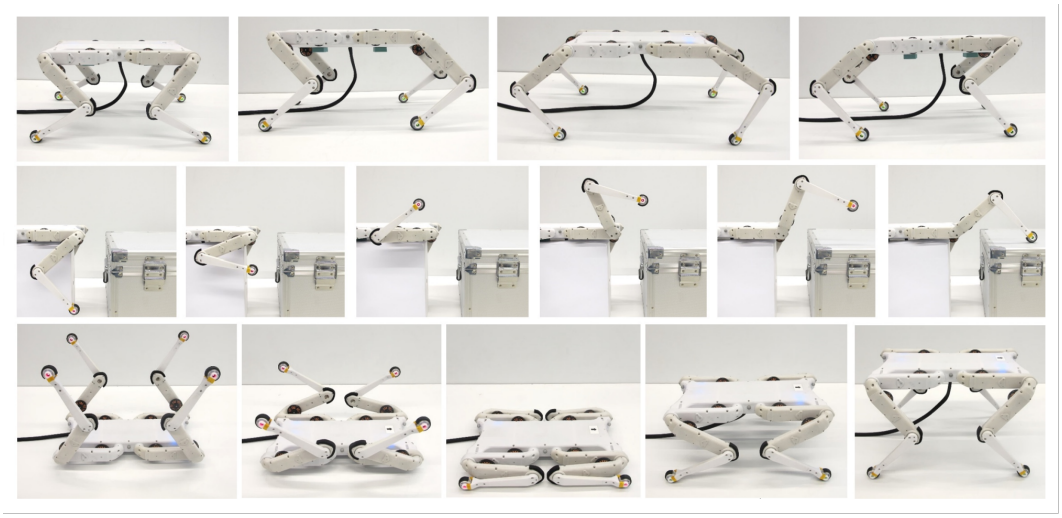
Cheetah 3 je robot vyvíjaný na MIT. Ako prvý štvornohý robot zvládol salto dozadu [3]. Má konštrukciu tela typickú pre veľa štvornohých robotov, nohy sa skladajú z 2 segmentov a každá noha má 3 motory. Väčšina jeho váhy je koncentrovaná v tele, keďže má jednoduché a ľahké nohy. Je ovládaný pomocou krútiaceho momentu. Pri riadení tohto robota sa využíva MPC, ktoré určí reakčné sily od podložky pre jednotlivé nohy, viď. Obr. 1.9. Do MPC vstupuje zjednodušený model robota, požadovaná rýchlosť, pozícia a natočenie robota. Chýba však spracovanie prostredia, keďže v [6] sa nespomínajú žiadne senzory, ktoré by snímali okolie robota.

Všetok software je verejne prístupný [18]. Využívajú Linux a je písaný v C++ [5]. Nepoužívajú však ROS [26], čo je veľká nevýhoda keďže pri našom robotovi už máme väčšinu kódu organizovaná v ROS balíčkoch a vyžadovalo by to veľké dodatočné úpravy.



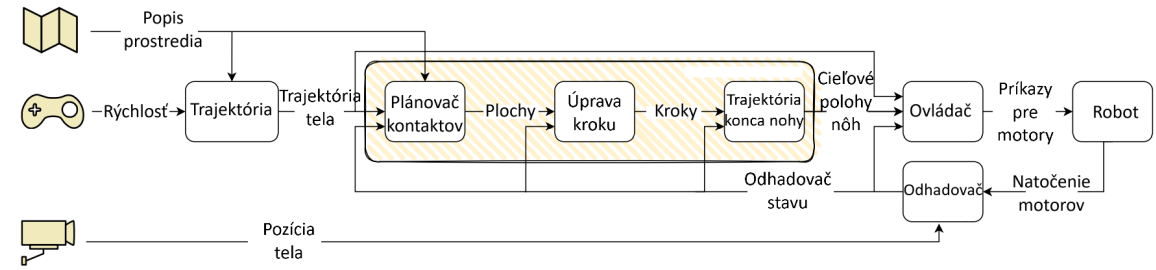
Obr. 1.9: Schéma riadenia robota Cheetah [17].

1.1.3 Solo



Obr. 1.10: Robot Solo [14]. Robot Solo má schopnosť dostať sa do rôznych konfigurácií so svojimi nohami, vďaka tomu, že má veľký rozsah kĺbov.

Robot Solo, Obr. 1.10, je výsledkom projektu Open Dynamic Robot Initiative, ktorého cieľom bolo vytvoriť neveľmi drahého a jednoduchého štvornohého robota na zostrojenie. Jeho telo sa skladá zo súčiastok, ktoré boli vytlačené na 3D tlačiarni alebo sú bežne dostupné, aj vďaka tomu váži len 2.2 kg. Patrí k robotom, ktoré sú riadené krútiacim momentom. V koncovej časti nôh má senzor, či je noha v kontakte s podložkou [14].

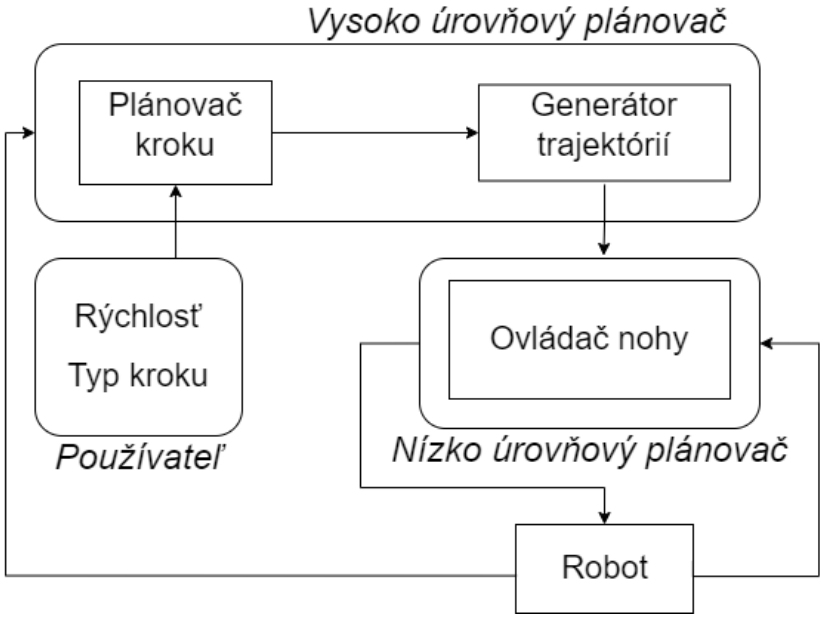


Obr. 1.11: Proces riadenia robota Solo [27].

Na Obr. 1.11 vidíme proces riadenia robota Solo. Najprv sa z údajov o prostredí vytvoria konvexné plochy a naplánujú sa kontakty s podložkou, teda kde je vhodné pre robota stúpiť. Tento plán počíta aj s aktuálnym stavom robota a požadovaným smerom robota ako príkazom od operátora. Potom sa vypočítajú trajektórie jednotlivých nôh, reprezentované ako spliny, s ohľadom na vyhnutie sa kolíziám. Časť Control na základe trajektórií jednotlivých nôh pomocou inverznej kinematiky vypočíta požadované natočenie jednotlivých kĺbov. Low-level control nakoniec pomocou PD-controllera a krútiacich momentov ovláda jednotlivé motory. Z hardvéru dostaváme spätnú odozvu o stave robota vo forme natočenia jednotlivých motorov a natočenia a pozície robota [27].

1.1.4 CHAMP

CHAMP je verejne dostupná knižnica pre riadenie štvornohých robotov [2]. Využíva ROS, je písaný v C++ a podporuje beh simulácie v Gazebe.



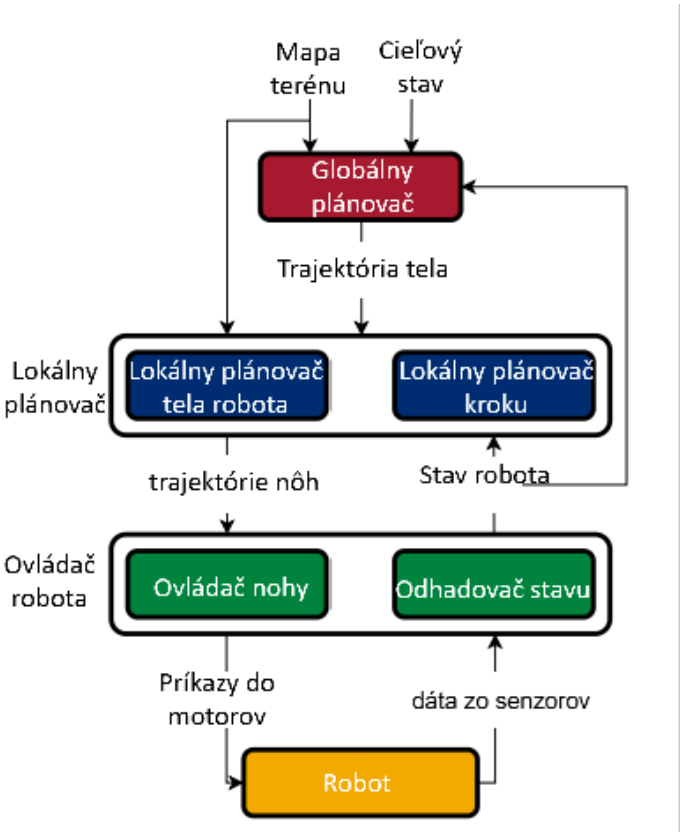
Obr. 1.12: Proces riadenia robota v CHAMP[19].

Ako vidíme na Obr. 1.12, riadenie robota má dva hlavné komponenty a to vysoko a nízko úrovňové riadenie.Vysoko úrovňové riadenie je zodpovedné za generovanie trajektórií pre jednotlivé nohy na základe požadovanej rýchlosti robota od operátora a typu chôdze. Nízko úrovňové riadenie posiela kontrolné signály motorom na základe trajektórií nôh. Z hardvéru sa ako odozva posielajú naspäť do vysoko úrovňového riadenia informácie o kontakte nohy s podložkou a do ovládača nohy zas údaje o stave motorov robota. Chýba však spracovanie prostredia, v ktorom sa robot pohybuje.

1.1.5 Quad SDK

Je verejne dostupný rámec [15], vytvorený špeciálne pre štvornohé roboty. Využíva ROS a je písaný predovšetkým v C++ a Pythone. Nie je urobený na mieru pre špecifického robota, ale všeobecne pre štvornohé roboty. Podporuje ovládanie a beh na hardvéri ale aj simuláciu v softvéri Gazebo.

Na Obr. 1.13 vidíme štruktúru Quad SDK. Globálny plánovač je zodpovedný za výpočet trajektórií tela robota bez kolízií tak, aby sa robot z aktuálneho stavu dostal do želaného. Vstupom je 2.5D reprezentácia okolitého terénu a cieľový stav robota. Cieľovým stavom môže byť aj príkaz od operátora z ovládača. Taktiež doň vstupuje aktuálny stav robota a parameter toho, aký je želaný krok robota [24].



Obr. 1.13: Štruktúra Quad SDK. Hierarchická štruktúra Quad SDK, ktorá má 3 hlavné časti: globálny plánovač, lokálny plánovač a ovládač robota [24].

Lokálny plánovač na základe okolitého terénu, výstupu z globálneho plávača a stavu robota vypočíta kedy a kde majú byť nohy v kontakte s podložkou a akou silou majú pôsobiť nohy na podložku. Má dve časti a to lokálny plánovač tela a lokálny plánovač kroku. Lokálny plánovač tela naplánuje najlepšie reakčné sily nôh od podložky, tak aby robot sledoval požadovanú trajektóriu tela. Lokálny plánovač kroku plánuje trajektórie jednotlivých nôh.

Tretia časť - ovládač robota je rozhranie medzi plánom a hardvérom, prípadne simuláciou. Ovládač nohy je zodpovedný za výpočet správnych krútiacich momentov na

motoroch na základe reakčných síl a trajektórií nôh. Odhadovač stavu zbiera informácie o robotovi zo senzorov za účelom zostrojenia čo najlepšieho obrazu o stave robota. To zahŕňa pozíciu a natočenia tela robota, ale aj uhol natočenia jednotlivých motorov. V prípade, že máme spustenú iba simuláciu tieto informácie poskytuje Gazebo.

Quad SDK okrem častí opísaných vyššie obsahuje pomocné triedy, kde je napríklad kinematika a dynamika robota a jeho celkový model. Na to využíva knižnicu RBDL [11]. Táto knižnica si načíta model robota zo súboru typu URDF. Tento typ súboru je štandarným formátom pre fyzický opis robota.

Táto knižnica bola testovaná nielen na robotoch v simulácií ale aj na hardvéri na rôznych robotoch ako napríklad Ghost Robotics Spirit alebo Unitree Robotics A1.

1.2 Diskusia

Riadenie štvornohých robotov je netriviálny proces, ktorý musí zohľadňovať veľa aspektov naraz. Pri adaptívnom riadení je dôležité mať spätnú väzbu z robota, ale ukazuje sa ako nevyhnutnosť poznať prostredie, v ktorom sa robot nachádza. To vyžaduje hardvér, ktorý s týmito potrebami ráta a má potrebné senzory. Od softvéru požadujeme aby vedel dáta zo senzorov spracovať v reálnom čase a taktiež v reálnom čase sa na ich základe rozhodnúť ako optimálne riadiť robota. Z vyššie popísaných možností riadenia najlepšie tieto potreby spĺňa Quad SDK. Tento framework ráta s dátami so senzorov, beží v reálnom čase a je možné ho adaptovať na rôznych robotov. Ráta však s tým, že väčšina robotov ma podobné nohy, ktoré sa skladajú z horného a spodného segmentu. Náš robot má o niečo komplikovanejšiu konštrukciu nôh, preto žiadne z vyššie spomínaných riešení nie je možné využiť bez väčších úprav.

Quad SDk taktiež využíva podobné technológie aké sme pri vývoja robota už použili a všetok kód je verejne dostupný, čo napríklad pri ANYmalovi nie je. CHAMP, tak ako aj robot Cheetah, neráta pri výpočte optimálneho riadenia robota s prostredím, v ktorom sa robot nachádza. Robot Solo má verejne dostupný kód a aj sníma svoje okolie, oproti Artabanovi je však veľmi ľahký a na každej nohe má o 1 motor menej. Na základe už vyššie spomenutého sme sa rozhodli použiť knižnicu Quad SDK pre riadenie robota Artabana.

Kapitola 2

Popis nášho robota Artabana

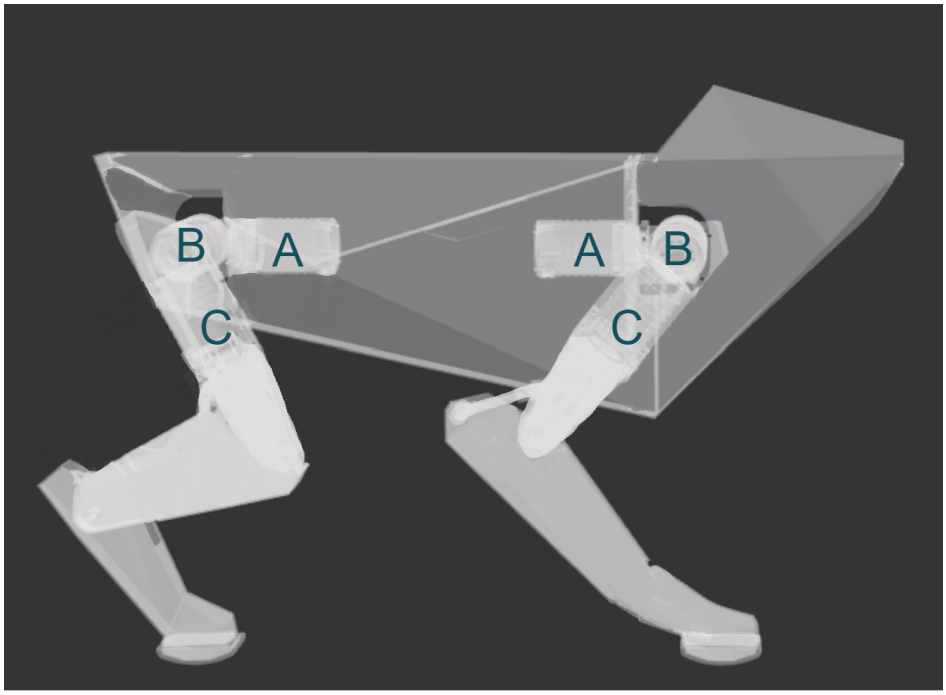
Obr. 2.1: Robot Artaban.

Predtým než začneme implementovať riadenie je nevyhnutné poznať niekoľko charakteristík nášho robota Artabana, ktorého vidíme na Obr. 2.1. To zahŕňa tvar jeho nôh, ktoré kĺby sú poháňané, priama aj inverzná kinematika, jakobiány pre výpočet krútiacich momentov zo síl pôsobiacich na jednotlivé nohy a inverznú dynamiku.

2.1 Nohy Artabana

Artaban je štvornohý robot, pričom každá noha je ovládaná 3 motormi. Motory vieme ovládať buď pozične alebo krútiacim momentom. Pri adaptívnom riadení robotov sa

využíva hlavne ovládanie krútiacim momentom. Všetkých motorov má robot spolu 12. Predné dve nohy sú si navzájom zrkadlovo symetrické, tak ako aj zadné dve.

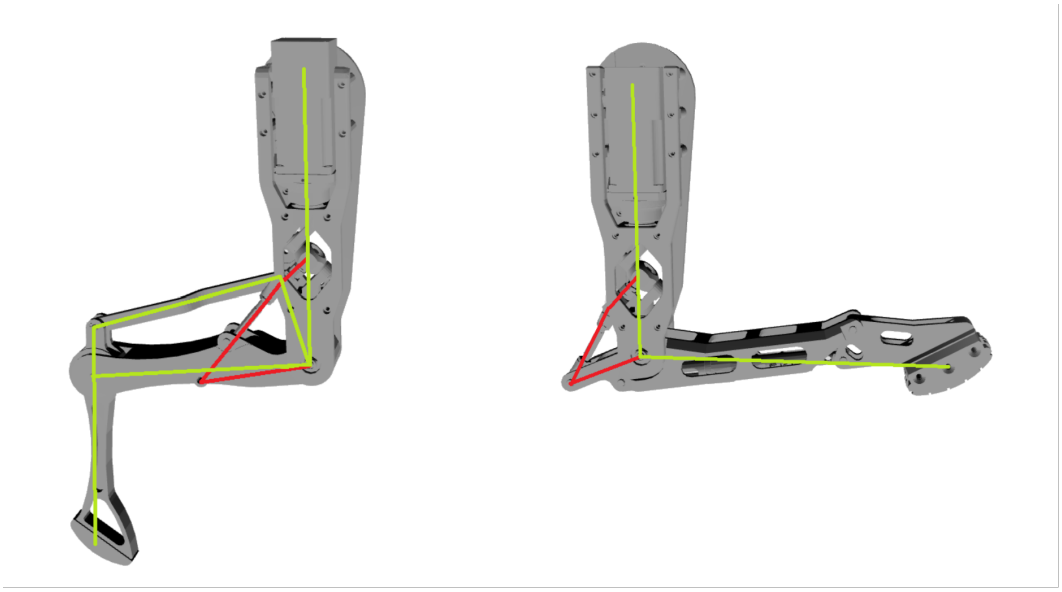


Obr. 2.2: Motory A, B, C zobrazené v tele robota pre prednú a zadnú nohu.

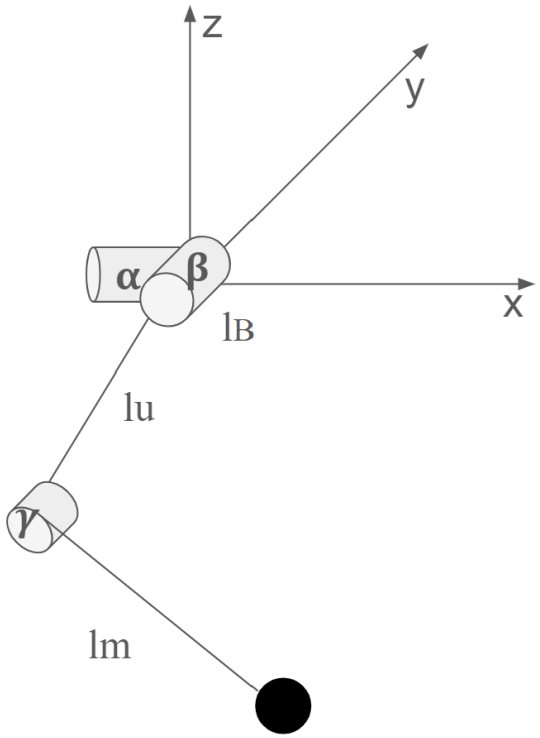
Na Obr. 2.2 vidíme znázornené jednotlivé motory. Motor A natáča nohu do strany, je dôležitý najmä pre otáčavé pohyby robota na mieste a pohyb do boku. Motor B ovláda celú nohu, hlavne pri švihu nohy pri kroku. Pre motor A aj B platí, že priamo otáčajú nohu. Pohyb motora C sa prenáša na nohu cez kardanový mechanizmus a následne cez 4-tyčový mechanizmus. Ovláda koleno robota, teda krčí a vystiera jednotlivé nohy. Nachádza sa v hornej časti nohy. Vzťah polohy, rýchlosti, zrýchlenia a krútiaceho momentu kolena je 1:1 k motoru C, preto pri ovládaní počítame s nohou akoby sme mali priamo motorom ovládané koleno. Údaje z motora o pozícií a rýchlosti prepočítame na koleno, prevedieme celý výpočet ako sa má noha ďalej hýbať a nakoniec výsledný krútiaci prepočítame z kolena na motor C. Toto zjednodušenie nôh nám umožní aj jednoduchší výpočet pre optimálne riadenie robota. Odteraz budeme pracovať ďalej so zjednodušenými nohami ako vidíme na Obr. 2.3.

2.2 Kinematika

Poznať kinematiku robota znamená, že poznáme výpočet priamej a inverznej kinematiky jeho nôh. Priama kinematika znamená výpočet polohy konca nohy na základe natočenia motorov. Inverzná kinematika predstavuje výpočet natočenia motorov zo súradníc konca nohy.



Obr. 2.3: Zjednodušenie nôh robota. Zelenou je znázornená zjednodušená reprezentácia nôh, červenou sú znázornené časti nohy, s ktorými počítame až pri spätnom prepočte z kolena na motor C.



Obr. 2.4: Značenie prednej nohy pri výpočte kinematiky. Uhly α, β, γ predstavujú natočenie motorov, dĺžky l_u, l_m predstavujú dĺžky častí nohy, l_B predstavuje dĺžku motora B.

Priamu kinematiku, pri značení znázornenom na Obr. 2.4, vypočítame nasledovne ako je uvedené v Rov. 2.1, 2.2, 2.3.

$$x = l_m sin(\beta + \gamma) + l_u sin(\beta) \tag{2.1}$$

$$y = l_B cos(\alpha) + l_m sin(\alpha) cos(\beta + \gamma) + l_u cos(\beta) sin(\alpha) \tag{2.2}$$

$$z = l_B sin(\alpha) - l_m cos(\alpha) cos(\beta + \gamma) - l_u cos(\alpha) cos(\beta) \tag{2.3}$$

Inverzná kinematika prednej nohy, tak ako ja inverzná kinematika zadnej sú popísané tu [22].

2.3 Jakobiány nôh

Pomocou jakobiánov nôh vieme vypočítať krútiace momenty jednotlivých motorov potrebné na to, aby sme na konci nohy vyvinuli požadovanú silu. Tento vzťah popisuje Rov. 2.4, kde J je jakobián danej nohy, τ sú krútiace momenty na pohonoch a F je sila, ktorá vplyvom τ vzniká na konci nohy.

$$\tau = J^T F \tag{2.4}$$

Jakobián vieme taktiež použiť na výpočet rýchlosti konca nohy z rýchlosti aktuátorov. Rov. 2.5 popisuje vzťah na výpočet rýchlosti konca nohy $\dot{v}$ z rýchlosti aktuátorov $\dot{q}$.

$$\dot{v} = J\dot{q} \tag{2.5}$$

Jakobián pre danú nohu získame ako deriváciu priamej kinematiky, Rov. 2.6, kde x, y, z predstavujú funkcie pre výpočet danej súradnice kona nohy z natočenia aktuátorov.

$$J = \begin{pmatrix} \frac{\partial x(\alpha,\beta,\gamma)}{\partial \alpha} & \frac{\partial x(\alpha,\beta,\gamma)}{\partial \beta} & \frac{\partial x(\alpha,\beta,\gamma)}{\partial \gamma} \\ \frac{\partial y(\alpha,\beta,\gamma)}{\partial \alpha} & \frac{\partial y(\alpha,\beta,\gamma)}{\partial \beta} & \frac{\partial y(\alpha,\beta,\gamma)}{\partial \gamma} \\ \frac{\partial z(\alpha,\beta,\gamma)}{\partial \alpha} & \frac{\partial z(\alpha,\beta,\gamma)}{\partial \beta} & \frac{\partial z(\alpha,\beta,\gamma)}{\partial \gamma} \end{pmatrix} \tag{2.6}$$

2.4 Inverzná dynamika robota

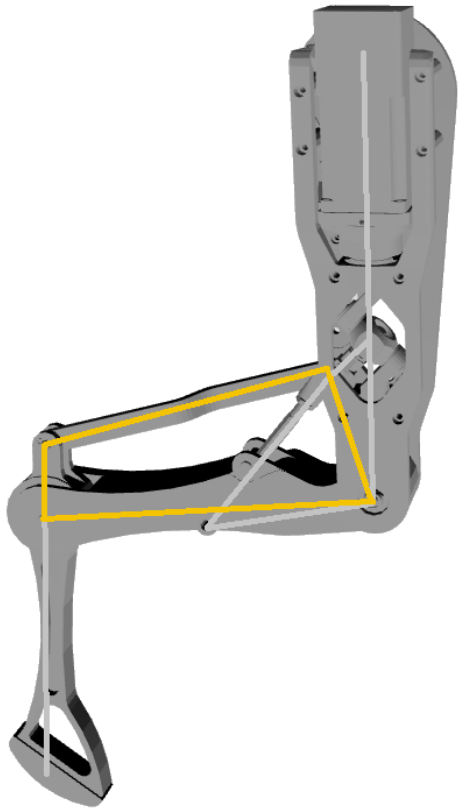
Poznáme priamu (FD) a inverznú dynamiku (ID) tuhých telies. Priama dynamika predstavuje výpočet zrýchlení v danom systéme na základe krútiacich momentov Rov. 2.7. Inverzná dynamika je výpočet síl alebo krútiacich momentov, ktoré musia vzniknúť v systéme aby sa vytvorilo požadované zrýchlenie [10], Rov. 2.8.

$$\ddot{q} = FD(model, q, \dot{q}, \tau) \tag{2.7}$$

$$\tau = ID(model, q, \dot{q}, \ddot{q}) \tag{2.8}$$

Pozíciu robota predstavuje q (pozícia ťažiska robota vo svete, natočenie jednotlivých kĺbov), $\dot{q}$ predstavuje rýchlosti, $\ddot{q}$ zrýchlenia a τ krútiace momenty. Reprezentácia robota *model* zahŕňa nielen rozmery jednotlivých častí robota a popis jednotlivých kĺbov ale aj hmotnosti všetkých častí.

Pri výpočte inverznej dynamiky štvornohého robota nás zaujíma τ, za podmienky, že poznáme $model, q, \dot{q}, \ddot{q}$. Najprv ukážeme ako sa rieši inverzná dynamika kinematického stromu a potom prejdeme ku kinematickým slučkám, keďže aj v zjednodušenom modeli Artabana sa nachádzajú v zadných nohách ako vidíme na Obr. 2.5.



Obr. 2.5: Kinematická slučka v zadnej nohe. Žltou farbou je zvýraznená kinematická slučka nachádzajúca sa v zadnej nohe robota.

Inverzná dynamika kinematického stromu

Výpočet inverznej dynamiky pre kinematický strom vychádza z pohybovej rovnice [10]:

$$\tau = H(q)\ddot{q} + C(q, \dot{q}, f_{ext}) \tag{2.9}$$

Kde H je matica zotrvačnosti, C je vektor síl odpovedajúci všetkým silám pôsobiacim v systém, vrátane gravitačnej sily, okrem tých, ktoré sú spôsobené τ.

Na výpočet τ sa využíva Newton-Eulerov algoritmus 1.

Algoritmus 1 Newton-Eulerov algoritmus pre výpočet inverznej dynamiky [10].

1: $v_0 = 0$
2: $a_0 = -a_g$
3: **for** $i = 1$ **to** N_B **do**
4: $\quad [X_J, S_i, v_J, c_J] = jcalc(jtype(i), q_i, \dot{q}_i)$
5: $\quad {}^{i}X_{\lambda(i)} = X_J X_T(i)$
6: $\quad$ **if** $\lambda(i) \neq 0$ **then**
7: $\quad\quad {}^{i}X_0 = {}^{i}X_{\lambda(i)} {}^{\lambda(i)}X_0$
8: $\quad$ **end if**
9: $\quad v_i = {}^{i}X_{\lambda(i)} v_{\lambda(i)} + v_J$
10: $\quad a_i = {}^{i}X_{\lambda(i)} a_{\lambda(i)} + S_i \ddot{q}_i + c_J + v_i \times v_J$
11: $\quad f_i = I_i a_i + v_i \times^* I_i v_i - {}^{i}X_0^* f_i^x$
12: **end for**
13: **for** $i = N_B$ **to** 1 **do**
14: $\quad \tau_i = S_i^T f_i$
15: $\quad$ **if** $\lambda(i) \neq 0$ **then**
16: $\quad\quad f_{\lambda(i)} = f_{\lambda(i)} + {}^{\lambda(i)}X_i^* f_i$
17: $\quad$ **end if**
18: **end for**

N_B predstavuje počet segmentov v kinematickom strome, pospájaných kĺbmi. Funkcie jtype a jcalc na riadku 4 popisujú jednotlivé kĺby. Funkcia jtype vracia typ kĺbu (otáčavý, posuvný, sférický, ...). Funkcia jcalc vracia všetky vlastnosti kĺbu daného typu, ktoré potrebujeme vedieť pre výpočet ID. Sú to X_j (transformácia medzi jednotlivými segmentami), S_i (pohybový podpriestor), v_J (rýchlosť kĺbu). S_i hovorí v našom prípade okolo ktorej osi sa otáča kĺb, keďže máme otáčavé kĺby. $\lambda(i)$ je funkcia, ktorá vracia rodiča i-teho stromu. ${}^{i}X_j$ predstavuje transformáciu medzi ľubovoľnými dvomi segmentmi stromu.

V prvom cykle na riadkoch 3 - 12, s využitím Newtonovej metódy, sa vypočítajú rýchlosti v_i a zrýchlenia a_i jednotlivých segmentov. Následne sa vypočíta sila f_i pôsobiaca na i-ty segment. I_i je inercia i-teho segmentu. Keď už poznáme f_i, vieme vypočítať krútiace momenty pre jednotlivé kĺby. To sa deje v druhou cykle na riadkoch 13 - 18, kde sa využíva Eulerova metóda.

Inverzná dynamika kinematického stromu je ako funkcia implementovaná vo viacerých knižniciach, napríklad aj v RBDL [11].

Inverzná dynamika Artabana

Zadné nohy nášho robota obsahujú kinematické slučky a preto nevieme priamo použiť hotové riešenia v podobe jedného volania funkcie knižnice alebo riešenie implementované v Quad SDK. Vo všeobecnosti je inverzná dynamika systému s kinematickými slučkami netriviálny problém. Inverznú dynamiku systému s kinematickými slučkami opisuje Rov. 2.10 [10], kde τ^c predstavuje sily spôsobené kinematickou slučkou a τ^a predstavuje aktívne sily spôsobené kinematickou slučkou (napríklad ďalší pohon v kĺbe, ktorý uzatvára slučku).

$$\tau + \tau^c + \tau^a = H(q)\ddot{q} + C(q, \dot{q}, f_{ext}) \tag{2.10}$$

V našom prípade však žiadne aktívne sily v mieste slučky nevznikajú, $\tau^a = 0$.

V kinematickom strome je medzi τ a $\ddot{q}$ vzťah 1:1, avšak ak sú v systéme kinematické slučky, tak nekonečne veľa rôznych hodnôt τ vyprodukuje rovnaké $\ddot{q}$ [10]. To platí ak sú všetky kĺby v systéme poháňané. Teda ak má systém menej stupňov voľnosti ako pohonov. Taktiež môže nastať ešte prípad kedy je viac stupňov voľnosti ako pohonov, teda nevieme ich všetky ovládať. Ideálny je však prípad, kedy sa rovnajú, čo je väčšinou cieľom pri návrhu robota a platí to aj pri Artabanovi.

Ďalším dôležitým predpokladom je, že na základe polohy poháňaných kĺbov vieme určiť polohu celého systému. Teda pri zadných nohách potrebujeme vedieť vypočítať zvyšné uhly v 4-tyčovom mechanizme, na základe uhlu kolena. Vo všeobecnosti pre 4-tyčový mechanizmus platí, že ak poznáme jeden uhol, tak celý mechanizmus môže byť v 2 polohách. Zadné nohy sú však limitované rozsahom jednotlivých kĺbov. Vďaka týmto limitom vieme určiť presnú polohu mechanizmu na základe jedného uhlu, keďže druhá poloha neprichádza do úvahy.

Ak tieto predpoklady platia, tak platí Rov. 2.11. Pričom τ_{ID} predstavuje výsledok ID pre príslušný kinematický strom a $y, \dot{y}, \ddot{y}$ v Rov. 2.12 predstavujú polohu, rýchlosť a zrýchlenie jednotlivých pohonov.

$$G_u^T \tau = G^T \tau_{ID} \tag{2.11}$$

$$\tau_{ID} = ID(\gamma(y), G\dot{y}, G\ddot{y} + g) \tag{2.12}$$

Funkcie γ, G, G_u, g sú funkciami kinematických slučiek, definované nasledovne:

- $q = \gamma(y)$, teda prepočet uhlov natočenia motorov na všetky kĺby v systéme
- $G = \frac{\partial \gamma}{\partial y}$
- G_u obsahuje riadky z G, pre poháňané kĺby

- $g = \dot{G}\dot{y}$

ID Artabana vypočítame nasledovne:

1. Z uhlu natočenia motorov y, rýchlosti motorov $\dot{y}$ a zrýchlenia motorov $\ddot{y}$ vypočítame $q, \dot{q}, \ddot{q}$.

2. Vypočítame τ_{ID} pre príslušný kinematický strom.

3. Z τ_{ID} vypočítame τ.

V 1.kroku potrebujeme poznať γ, G, g. Funkcia γ má na vstupe y - vektor dĺžky 18, ktorý predstavuje stav robota, teda pozíciu v prostredí, trojicu (x, y, z), natočenie robota okolo jednotlivých os (r, p, y) a natočenie jednotlivých motorov. Každá noha má 3 motory a vo vektore y idú v poradí ľavá predná, ľavá zadná, pravá predná, pravá zadná. Na výstupe funkcie γ chceme mať vektor dĺžky 22, kde sú zahrnuté natočenia všetkých kĺbov zadných nôh v príslušnom kinematickom strome. Funkciu γ vidíme na Rov. 2.13.

$$\gamma(y) = q = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \\ \gamma_1(y_{12}) \\ \gamma_2(y_{12}) \\ y_{13} \\ y_{14} \\ y_{15} \\ y_{16} \\ y_{17} \\ y_{18} \\ \gamma_1(y_{18}) \\ \gamma_2(y_{18}) \end{bmatrix} \tag{2.13}$$

Funkcie γ_1, γ_2 sú pomocné funkcie, ktoré nám vracajú príslušné uhly v 4-tyčovom mechanizme, na základe natočenia kolena y_{12} alebo y_{18}. Keďže ľavá a pravá noha sú symetrické, tak sú tieto funkcie rovnaké pre obe nohy. Vychádzame v nich z rovníc pre 4tyčový mechanizmus [22]. Funkciu γ_1 vidíme na Rov. 2.17.

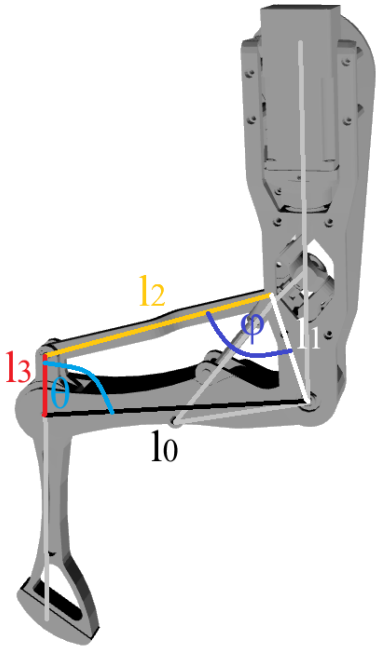
$$k_1 = -2l_2l_0\sin(y_{in}) \tag{2.14}$$

$$k_2 = -2l_2(l_3 + l_0cos(y_{in})) \tag{2.15}$$

$$k_3 = l_3^2 + l_2^2 - l_1^2 + l_0^2 + 2l_0l_3cos(y_{in}) \tag{2.16}$$

$$\phi = \gamma_1(y_{in}) = 2atan2(-k_1 + \sqrt{(k_1^2 + k_2^2 - k_3^2)}, k_3 - k_2) \tag{2.17}$$

Za y_{in} môžeme dosadiť y_{12} alebo y_{18} a dostaneme výsledok pre ľavú alebo pravú zadnú nohu. Funkciu γ_2 vidíme v Rov. 2.21. Použité značenie je na Obr. 2.6.



Obr. 2.6: Značenie 4-tyčového mechanizmu zadnej nohy.

$$k_1 = -2l_1l_3\sin(y_{in}) \tag{2.18}$$

$$k_2 = -2l_1(l_0 + l_3cos(y_{in})) \tag{2.19}$$

$$k_3 = l_0^2 + l_1^2 - l_2^2 + l_3^2 + 2l_0 l_3 cos(y_{in}) \tag{2.20}$$

$$\theta = \gamma_2(y_{in}) = 2atan2(-k_1 + \sqrt{(k_1^2 + k_2^2 - k_3^2)}, k_3 - k_2) \tag{2.21}$$

Ďalej potrebujeme poznať G, to získame ako:

$$G = \frac{\partial \gamma}{\partial y} = \begin{bmatrix} \frac{\partial y_1}{\partial y_1} & \frac{\partial y_1}{\partial y_2} & \cdots & \frac{\partial y_1}{\partial y_{12}} & \cdots & \frac{\partial y_1}{\partial y_{18}} \\ \frac{\partial y_2}{\partial y_1} & \frac{\partial y_2}{\partial y_2} & \cdots & \frac{\partial y_2}{\partial y_{12}} & \cdots & \frac{\partial y_2}{\partial y_{18}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial \gamma_1}{\partial y_1} & \frac{\partial \gamma_1}{\partial y_2} & \cdots & \frac{\partial \gamma_1}{\partial y_{12}} & \cdots & \frac{\partial \gamma_1}{\partial y_{18}} \\ \frac{\partial \gamma_2}{\partial y_1} & \frac{\partial \gamma_2}{\partial y_2} & \cdots & \frac{\partial \gamma_2}{\partial y_{12}} & \cdots & \frac{\partial \gamma_2}{\partial y_{18}} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial \gamma_3}{\partial y_1} & \frac{\partial \gamma_3}{\partial y_2} & \cdots & \frac{\partial \gamma_3}{\partial y_{12}} & \cdots & \frac{\partial \gamma_3}{\partial y_{18}} \\ \frac{\partial \gamma_3}{\partial y_1} & \frac{\partial \gamma_3}{\partial y_2} & \cdots & \frac{\partial \gamma_3}{\partial y_{12}} & \cdots & \frac{\partial \gamma_3}{\partial y_{18}} \end{bmatrix} =$$

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & G_{13,12} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & G_{14,12} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & G_{21,18} \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & G_{22,18} \end{bmatrix}$$

G predstavuje prepočet rýchlostí poháňaných kĺbov na všetky v systéme. Jediné nepoháňané kĺby v systéme sú tie v 4-tyčovom mechanizme. O rýchlostiach v 4-tyč mechanizme vieme nasledovné [28]:

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{l_3 sin(\pi - \theta - y_{in})}{l_2 sin(y_{in} + \phi + \theta - 2\pi)} \\ \frac{l_1 sin(\phi - \pi)}{l_3 sin(y_{in} + \phi + \theta - 2\pi)} \end{bmatrix} \dot{y}_{in} \tag{2.22}$$

Na základe Rov. 2.22 vieme odvodiť chýbajúce členy G:

$$G_{13,12} = \frac{l_3 sin(\pi - \theta_L - y_{12})}{l_2 sin(y_{12} + \phi_L + \theta_L - 2\pi)} \tag{2.23}$$

$$G_{14,12} = \frac{l_1 sin(\phi_L - \pi)}{l_3 sin(y_{12} + \phi_L + \theta_L - 2\pi)} \tag{2.24}$$

$$G_{21,18} = \frac{l_3 sin(\pi - \theta_P - y_{18})}{l_2 sin(y_{18} + \phi_P + \theta_P - 2\pi)} \tag{2.25}$$

$$G_{22,18} = \frac{l_1 sin(\phi_P - \pi)}{l_3 sin(y_{18} + \phi_P + \theta_P - 2\pi)} \tag{2.26}$$

Pričom θ_L, ϕ_L v Rov. 2.23 a 2.24 označujú hodnoty pre ľavú nohu a θ_P, ϕ_P v Rov. 2.25 a 2.26 pre pravú nohu.

Vektor g získame ako:

$$g = \dot{G}\dot{y} = \frac{\partial G}{\partial t}\dot{y} = \frac{\partial G}{\partial y}\frac{\partial y}{\partial t}\dot{y} = \frac{\partial G}{\partial y}\dot{y}^2 \tag{2.27}$$

Teda:

$$g = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g_{13} \\ g_{14} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ g_{21} \\ g_{22} \end{bmatrix}$$

Pričom platí, že:

$$g_{13} = \frac{\partial G_{13,12}}{\partial y_{12}} \dot{y}_{12}^2 = \left(\frac{l_3 \cos(y_{12} + \theta_L)}{l_2 \sin(y_{12} + \theta_L + \phi_L)} - \right. \\ \left. - \frac{l_3 \sin(y_{12} + \theta_L) \cos(y_{12} + \theta_L + \phi_L)}{l_2 \sin^2(y_{12} + \theta_L + \phi_L)} \right) \dot{y}_{12}^2 \tag{2.28}$$

$$g_{14} = \frac{\partial G_{14,12}}{\partial y_{12}} \dot{y}_{12}^2 = \frac{l_1 sin(\phi_L) cos(y_{12} + \theta_L + \phi_L)}{l_3 sin^2(y_{12} + \theta_L + \phi_L)} \dot{y}_{12}^2 \tag{2.29}$$

$$g_{21} = \frac{\partial G_{21,18}}{\partial y_{18}} \dot{y}_{18}^2 = \left(\frac{l_3 \cos(y_{18} + \theta_P)}{l_2 \sin(y_{18} + \theta_P + \phi_P)} - \right. \\ \left. - \frac{l_3 \sin(y_{18} + \theta_P) \cos(y_{18} + \theta_P + \phi_P)}{l_2 \sin^2(y_{18} + \theta_P + \phi_P)} \right) \dot{y}_{18}^2 \tag{2.30}$$

$$g_{22} = \frac{\partial G_{22,18}}{\partial y_{18}} \dot{y}_{18}^2 = \frac{l_1 sin(\phi_P) cos(y_{18} + \theta_P + \phi_P)}{l_3 sin^2(y_{18} + \theta_P + \phi_P)} \dot{y}_{18}^2 \tag{2.31}$$

Kapitola 3

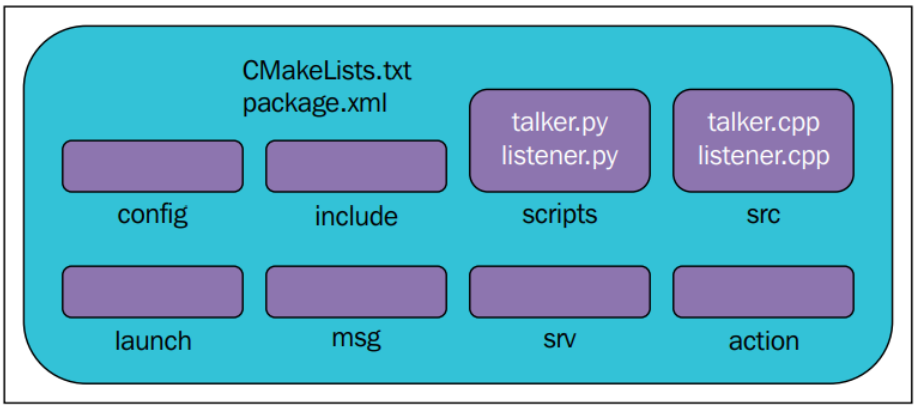
Implementácia

3.1 Prehľad technológií

Pre riadenie robota sme sa rozhodli využiť Quad SDK. Tento rámec je schopný adaptívneho riadenia štvornohých robotov, je verejne dostupný aj s dokumentáciou. Využíva nasledovné technológie: ROS, URDF, SDF, C++, Python, Gazebo. Väčšinu z nich sme pri riadení robota používali aj doteraz, čo je veľká výhoda.

ROS

ROS je verejne dostupný systém. Pozostáva z knižníc a nástrojov, ktoré majú pomôcť vývojárom vyvíjať štruktúrovaný a udržateľný kód pre robotov [26].



Obr. 3.1: Štruktúra ROS balíčku [16].

Funguje ako množina procesov, ktoré je možné jednotlivo púšťať, a komunikujú medzi sebou posielaním a prijímaním správ. Toto umožňuje zmeniť časť kódu bez zásahu alebo obmedzení na inej. Jednotlivé procesy je možné zoskupovať do balíčkov [16]. Typická štruktúra takéhoto balíčku je na Obr. 3.1. Tieto balíčky väčšinou potom predstavujú jednotlivé repositories na Githube pri spolupráci viacerých vývojárov.

ROS taktiež obsahuje viacero bežne používaných knižníc a funkcionalít pri vývoji robotov, napríklad low-level control. Podporuje viacero programovacích jazykov, okrem iných aj C++ a Python.

URDF

URDF je XML formát súboru, slúžiaci na opis robota v ROS-e. Základnými kameňmi tohto formátu sú segment (link) a kĺb (joint) [16].

Kĺb spája vždy dva segmenty. Poznáme viacero typov kĺbov a to napríklad otáčavý, s limitmi alebo bez limitov, posuvný a pevný - pevne spája dva segmenty. Dôležitým parametrom každého kĺbu je, kde sa vrámci segmentov, ktoré spája, nachádza. Taktiež každý kĺb má popísanú svoju os otáčania ako 3d vektor. Nižšie vidíme príklad kĺbu v URDF:

```
<joint name="joint1" type="continuous">
   <parent link="link1"/>
   <child link="link2"/>
   <axis xyz="1 0 0"/>
   <origin xyz="0 0 0.5"/>
 </joint>
```

Kĺb joint1 je otáčavý kĺb bez limitov a spája segment link1 a link2. Otáča sa okolo osi x.

Segment popisuje hmotnosť jednotlivých častí robota a ich inerčnú maticu. Nižšie vidíme príklad definície segmentu:

```
<link name="link1">
    <inertial>
      <mass value="5"/>
      <inertia
        ixx="0.3"
        ixy="0.0"
        ixz="0.0"
        iyy="0.4"
        iyz="0.0"
        izz="0.2"/>
    </inertial>
  </link>
```

URDF je schopné popísať kinematický strom, avšak nie je schopné popísať kinematickú slučku. Tá sa dá uzatvoriť dodatočne v niektorých softvéroch.

Vo formáte URDF máme popis celého Artaban so všetkými jeho časťami - segmentmi a kĺbmi.

SDF

SDF je ďalší XML formát súboru pre popis robota. Využíva ho najmä Gazebo. Je podobný URDF, tiež jeho základnom je segment (link) a kĺb (joint), ale umožňuje vytvárať aj kinematické slučky v systéme [16]. URDF je do SDF možné konvertovať automaticky.

Gazebo

Gazebo je fyzikálny 3D simulátor, určený primárne pre vývoj robotov. Umožňuje simuláciu robota aj s prostredím, v ktorom sa pohybuje. Vďaka tomu je možné si overiť funkčnosť kódu a vyladiť niektoré chyby ešte predtým ako sa otestuje na robotovi.

3.2 Návrh a Implementácia

Pri riadení robota využívame ROS, ktorý zabezpečuje dostupnosť funkcionalít jednotlivých balíčkov, komunikáciu medzi jednotlivými procesmi, zaznamenávanie dát pri behu programov na hardvéri robota ale aj podporu pri simulácii. Všetko softvér je rozdelený do ROS balíčkov. V nasledujúcich častiach si prejdeme niekoľko najdôležitejších pre riadenie robota.

Popis robota Artabana

Tento balíček obsahuje súbory popisujúceho robota vo formáte URDF aj SDF. Súbory popisujúce Artabana obsahujú informácie o všetkých jeho segmentoch a kĺboch. Pri segmentoch nás zaujíma kde sa nachádzajú, aké su ich rozmery a koľko vážia. Pri kĺboch je dôležité vedieť typ kĺbu, ktoré segmenty spája a v ktorom mieste a či je daný kĺb poháňaný (je tam motor) alebo nie. Formát URDF využíva Gazebo pri zobrazení robota v simulácii a knižnica RBDL. SDF formát je potrebný pre Quad SDK. Dôležité je aby tieto súbory čo najvernejšie popisovali skutočný hardvér robota.

Rozhranie pre hardvér

Pre komunikáciu s motormi máme zvlášť balíček, ktorý poskytuje nevyhnutnú funkcionalitu pre riadenie robota. To zahŕňa prijímanie riadiacich príkazov od iného balíčku, informáciu o požadovanej polohe jednotlivých motorov alebo krútiacich momentov, a poslanie riadiaceho príkazu motoru. Taktiež zabezpečuje tok informácií opačným smerom, teda poskytuje informáciu o polohe a rýchlosti motora iným balíčkom.

Kinematika Artabana

Všetky výpočty súvisiace s kinematikou Artabana sa nachádzajú tu. To zahŕňa funkcie priamej a inverznej kinematiky pre jednotlivé nohy, jakobiány a prepočty medzi zjednodušeným a úplným modelom nôh. Táto celá časť je písaná v C++ s dôrazom na rýchlosť výpočtov.

Quad SDK

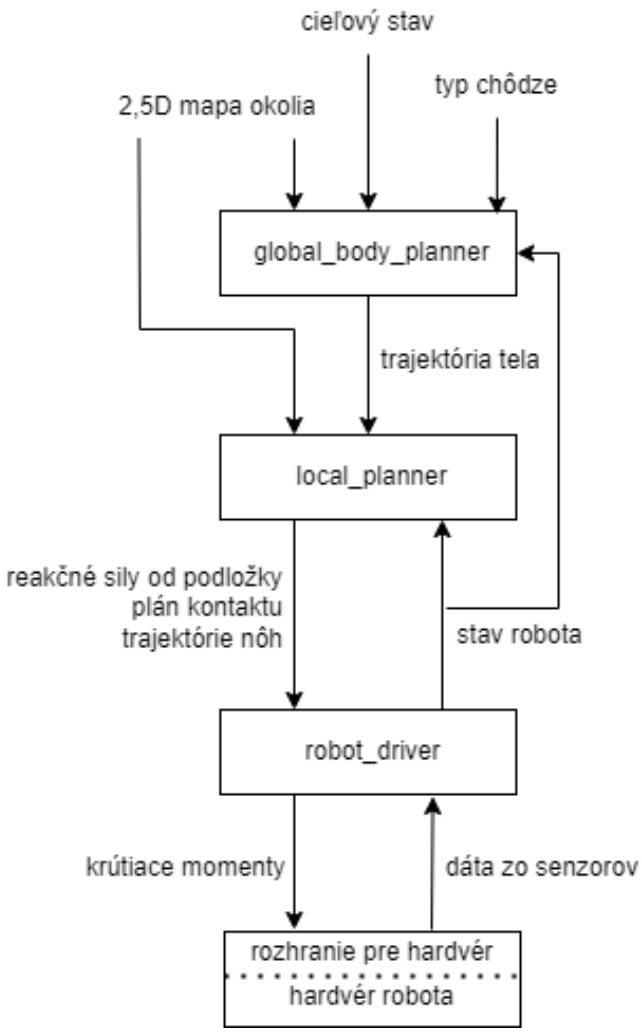
Quad SDK je zodpovedný za adaptívne riadenie robota. Plánuje trasu robota, trajektóriu a rýchlosť jednotlivých nôh a výstupom je požadovaný krútiaci moment na jednotlivých motoroch. Tieto úlohy vykonáva za pomoci vyššie spomenutých balíčkov. Quad SDK je možné prispôsobiť na rôznych štvornohých robotov, pri Artabanovi však bolo potrebené vykonať úpravy nad rámec krokov, popísaných v dokumentácií. Pre atypické nohy Artabana bolo potrebné mať vlastnú implementáciu kinematiky a aj inverznej dynamiky. Nastaviť správne parametre v konfiguračných súborov, napr. hmotnosť robota, výška tela, dĺžka kroku atď.. Taktiež Artaban má rôzne predné a zadné nohy, čo vyžadovalo ďalšie úpravy v rámci viacerých balíčkov.

Na Obr. 3.2 vidíme hlavné komponenty riadenia robota Artabana a s akými dátami pracujú. Nižšie si prejdeme zoznam všetkých častí knižnice Quad SDK a ich funkcií.

- **external** - Externé knižnice, ktoré využíva Quad SDK. Zahŕňa RBDL, využíva sa pri výpočtoch dynamiky a načítaní modelu robota z URDF súboru, Ipopt, knižnicu pre riešenie optimalizačných problémov, a teleop_twist_joy, ktorý slúži pre ovládanie pomocou pákového ovládaču.

- **global_body_planner** - Globálny plánovač, ktorý plánuje trajektóriu tela na základe mapy terénu, cieľového stavu a aktuálneho stavu robota. Pracuje iba so zjednodušeným modelom robota, zohľadňujúcim predovšetkým hmotnosť, ktorý je nezávislý od tvaru nôh robota.

- **local_planner** - Plánuje trajektórie jednotlivých nôh a sily pôsobiace medzi nohami a podložkou robota a čas kontaktu jednotlivých nôh s podložkou. Nachádza sa tu taktiež súbor s konfiguračnými parametrami, ako sú napríklad typ kroku robota, dĺžka trvania kroku a želaná výška tela robota nad podložkou. Typ kroku robota a dĺžka sú veľmi dôležité parametre, ktoré majú vplyv na stabilitu a plynulosť chôdze robota, tak ako aj výška, v ktorej sa drží telo. V prípade, že sa telo robota nachádza príliš vysoko, robot je nestabilný a náchylný k pádom. V opačnom prípade ak je robot príliš nízko, tak sa môže stať, že nevie dvíhať dostatočne vysoko nohy a tým pádom nasledovať naplánovanú trajektóriu nôh kvôli obmedzeným rozsahom jednotlivých kĺbov.



Obr. 3.2: Prehľad riadenia robota Artabana.

- **nmpc_controller** - Tu sa nachádza implementácia NMPC, ktorá vracia reakčné sily od podložky pre jednolitvé nohy a stav robota, také aby robot sledoval naplánovú trajektóriu svojho tela. Tento balíček sa využíva v local_planner.

- **quad_logger** - Slúži na zaznamenávanie a prácu s dátami, ktoré boli zozbierané počas jednotlivých behov programu, či už v simulácií alebo na hardvéri. Dáta sa zaznamenávajú do súborov vo formáte .bag. Pomáhajú nám vyhodnocovať, ktoré parametre kroku sú pre robota vhodnejšie, ale aj spätne nájsť chyby v riadení robota.

- **quad_msgs** - Balíček, ktorý obsahuje ako majú vyzerať správy pre popis stavu robota, riadiacich príkazov robota a plánu robota. Pomáha udržiavať jasnú štruktúru správ pri ich spracovaní v rôznych iných častiach kódu. Pomocou týchto správ komunikujú potom zvyšné balíčky, napríklad local_planner a robot_driver komunikajú medzi sebou pomocou týchto správ.

- **quad_simulator** - Obsahuje softvér potrebný pre spustenie a beh simulácie robota v prostredí Gazebo.

- **quad_utils** - Pomocné funkcie pre iné balíčky v rámci Quad SDK. Tu je implementované čítanie modelu robota z URDF za pomoci RBDL, inverzná dynamika ale poskytuje aj funkcie kinematiky robota, pri Artabanovi volá funkcie z nášho balíčka, kde sú implementované. Dôležitými pomocnými funkciami sú aj tie, ktoré robia transformácie medzi rôznymi súradnicovými systémami, napríklad medzi systémom sveta, v ktorom sa robot nachádza a súradnicovým systémom, ktorý má začiatok v ťažisku robota.

- **robot_driver** - Ovládač robota, ktorý sa skladá z 3 častí. Prvá sú ovládače, ktoré prijímajú informácie o trajektórií nôh, kedy sa majú nohy dotýkať podložky a reakčných síl od podložky ktoré majú pôsobiť v miestach kde sa noha dotýka podložky z local_planner. Na základe týchto dát sa počítajú optimálne krútiace momenty pre jednotlivé motory τ .

$$\tau = \tau_{ID} + \tau_{PD} \tag{3.1}$$

Rov. 3.1 slúži na výpočet τ, kde τ_{ID} je výsledok inverznej dynamiky a z quad_utils a τ_{PD} predstavuje spätnú väzbu z PD regulátora. PD regulátor je odvodený z PID regulátora vynechaním integračnej zložky. Krútiaci moment τ_{PD} získame nasledovne:

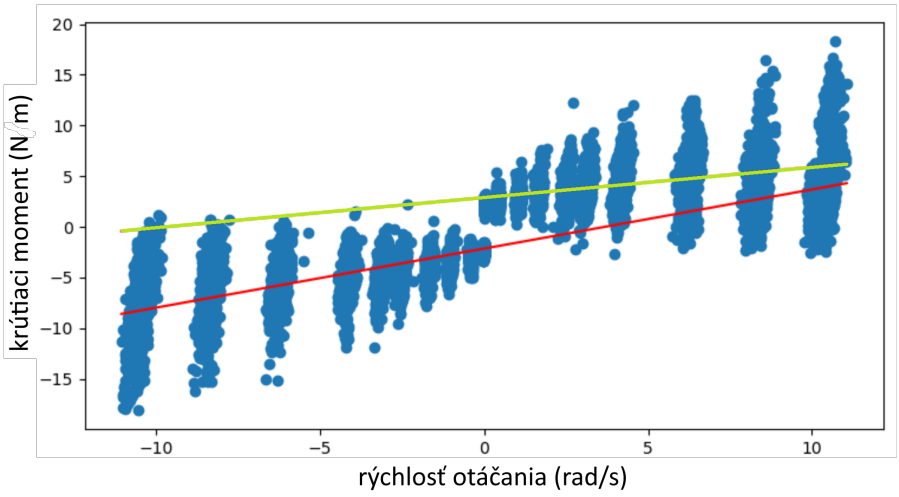
$$\tau_{PD} = P(q_{plan} - q) + D(\dot{q}_{plan} - \dot{q}) \tag{3.2}$$

Pričom $q_{plan}, \dot{q}_{plan}$ predstavujú požadované polohy a rýchlosti motorov na základe plánu z lokálneho plánovača a $q, \dot{q}$ sú skutočné polohy a rýchlosti získané zo senzorov. P je koeficient proporčnej zložky a D je koeficient derivačnej zložky. Hodnotu týchto dvoch koeficientov sme nastavili empiricky. Rov. 3.1 zobrazuje ako sa vykonáva výpočet τ pôvodne v Quad SDK a pri behu simulácie. Pri kráčaní s robotom Artabanom ešte dodatočne kompenzujeme trenia na motoroch a prevodovkách a ich zotrvačnosti. Keďže sme pri experimentoch na harvéri spozorovali veľký rozdiel medzi správaním robota v simulácii a hardvérom. Výsledné τ vypočítame na základe Rov. 3.3, kde τ_F predstavuje krútiaci moment potrebný na prekonanie trení v motoroch a prevodovkách a τ_I predstavuje krútiaci moment potrebný na prekonanie zotrvačností motorov a prevodoviek.

$$\tau = \tau_{ID} + \tau_{PD} + \tau_F + \tau_I \tag{3.3}$$

Rov. 3.4 zobrazuje výpočet τ_F na základe rýchlosti motora, $\dot{q}$, a^+, b^+, a^-, b^- sú koeficienty, získané pomocou lineárnej regresie z nameraných dát, ktoré vidíme na Obr. 3.3.

$$\tau_F = \begin{cases} a^+\dot{q} + b^+ & \text{if } \dot{q} > 0 \\ a^-\dot{q} + b^- & \text{otherwise} \end{cases} \tag{3.4}$$



Obr. 3.3: Namerané hodnoty krútiaceho momentu pre udržanie motoru v danej rýchlosti pre motor na kolene ľavej zadnej nohy. Na x-ovej osi vidíme rýchlosť otáčania motoru a na y-ovej je potrebný krútiaci moment aby si motor udržal danú rýchlosť. Modrou sú znázornené namerané hodnoty. Zelenou je znázornená závislosť pri otáčaní v kladnom smere, kde $a^+ = 0.2997, b^+ = 2.8929$. Červenou je znázornená závislosť v zápornom smere, kde $a^- = 0.5827, b^- = -2.1459$.

Výpočet τ_I vidíme v Rov. 3.5, kde I je zotrvačnosť daného motora a prevodovky a $\ddot{q}$ je zrýchlenie.

$$\tau_I = I\ddot{q} \tag{3.5}$$

Hodnota I je daná typom motora a prevodovky a poznáme ju od výrobcu. Zrýchlenie na motore nepoznáme, keďže motory nám dávajú iba spätnú väzbu o svojej polohe a rýchlosti. Preto je potrebné $\ddot{q}$ vypočítať, ako vidíme v Rov. 3.6.

$$\ddot{q} = \frac{\dot{q}_t - \dot{q}_{t-1}}{\Delta t} \tag{3.6}$$

Pre výpočet zrýchlenia použijeme aktuálnu hodnotu $\dot{q}_t$ a poslednú predošlú nameranú hodnotu $\dot{q}_{t-1}$, ich rozdiel vydelíme časom Δt ktorý prešiel medzi zaznamenaním týchto dvoch hodnôt.

Druhou časťou sú odhadovače stavu, ktoré poskytujú informácie o stave robota plánovačom na základe dát z ovládačov motorov z rozhrania s hardvérom. Poslednou časťou je trieda pre rozhranie s hardvérom. Tá je nevyhnutná pre rôzne špecifiká harvéru ako je napríklad iné poradie motorov pri posielaní správ ako v Quad SDK, ale aj kompenzácia trení a inercií motorov a prevodoviek, ktoré sú typické pre daný hardvér a nie sú simulované v Gazebe. Taktiež sa tu prepočítavajú hodnoty zo zjednodušeného modelu robota na skutočný, Obr. 2.3, teda z kolien robota na motor C.

Kapitola 4

Výsledky

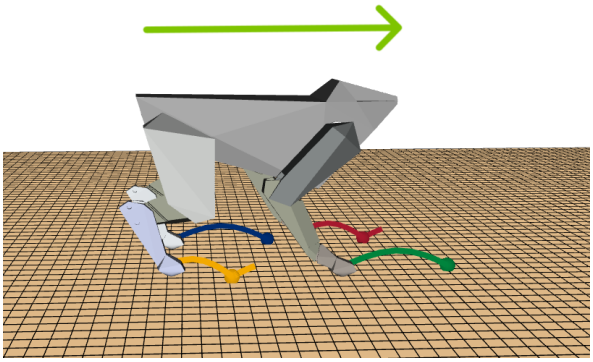
Chôdzu robota sme testovali viacerými spôsobmi riadenia. Pre porovnanie sme otestovali aj priame pozičné ovládanie a ovládanie pomocou krútiaceho momentu a PD-regulátora. Ďalej sme testovali ovládanie pomocou knižnice Quad SDK bez inverznej dynamiky, Quad SDK s inverznou dynamikou a nakoniec aj s kompenzáciou trení a zotrvačností na prevodovkách a motoroch.

V Quad SDK sme testovali rôzne typy chôdze - chôdzu krokom Obr. 1.2 a klusom Obr. 1.3. Menili sme aj parametre parametre kroku - konkrétne dĺžku kroku, tá sa nastavuje pomocou dĺžky periódy, teda čas za ktorý majú vykročiť všetky nohy. Testovali sme hlavne krátky krok, s trvaním periódy 0.6s, a dlhý krok s trvaním periódy 0.72s. Skúšali sme aj iné dĺžky periódy avšak pri kratších ani dlhších robot nezvládal vôbec stabilne kráčať. Výšku tela robota, v ktorej má robot kráčať sme nastavili na 0.45m. Pre každého robota môže byť táto výška iná, keďže každý robot má inak dlhé časti nôh. Túto výšku sme vypočítali pomocou kinematiky tak, aby kolená robota boli pri státí v strede svojho rozsahu, teda nohy mali priestor sa krčiť aj vystierať. Ak je táto výška nižšie robot nevie zdvíhať dostatočne nohy zo zeme, keďže je limitovaný rozsahmi kolena. Ak sa telo robota nachádza príliš vysoko, tak robot má problém robiť dostatočne dlhé kroky, keďže ho zastavia limity maximálneho vystretia kolien.

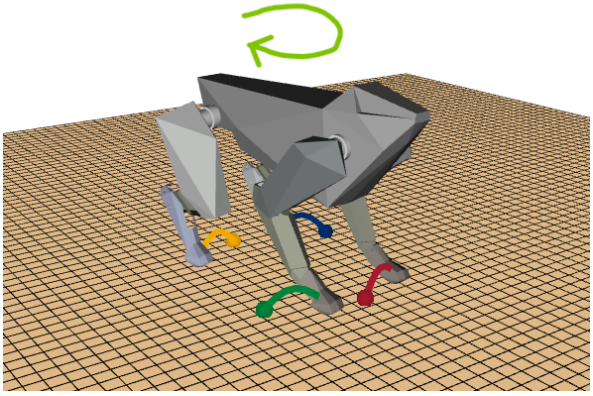
Nevyhnutnou súčasťou testovania bolo nastavenie správnych koeficientov PD-regulátora, ktoré sa môžu líšiť pre softvér a hardvér, ale aj hľadanie a opravovanie chýb v softvéri, či opravovanie hardvéru.

Pri kráčaní robota sme skúšali rôzne typy pohybov. Najprv chôdzu vpred, ktorú vidíme na Obr. 4.1, otáčanie sa na mieste Obr. 4.2, chôdza vzad Obr. 4.3 a kráčanie do strany Obr. 4.4. V simulácii sme otestovali aj chôdzu robota po schodoch. Na hardvéri sme overili schopnosť robota udržať si stabilitu pri postrčenia do robota z boku.

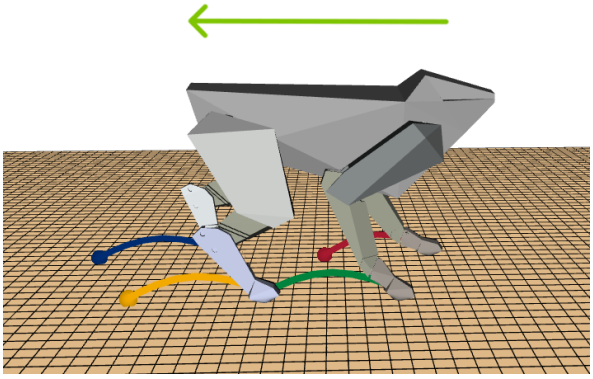
Na záver sme jednotlivé metódy vyhodnotili a porovnali, na základe toho aké pohyby robot zvládol robiť a či zvládol nečakané vplyvy z okolia.



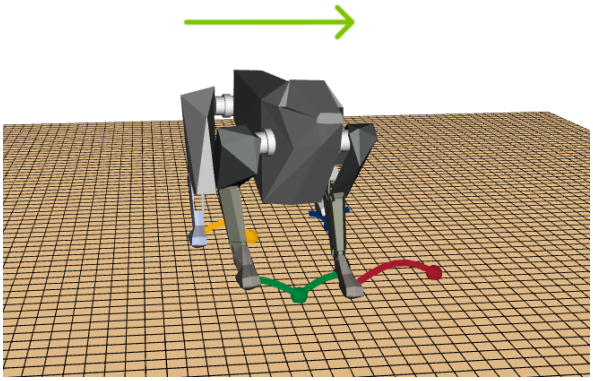
Obr. 4.1: Chôdza vpred s naplánovaným krivkami pre jednotlivé nohy. Kruhy na koncoch naplánovaných kriviek predstavujú bod kontaktu nohy s podložkou. Šípka nad robotom predstavuje smer pohybu.



Obr. 4.2: Otáčanie sa na mieste s naplánovaným krivkami pre jednotlivé nohy. Kruhy na koncoch naplánovaných kriviek predstavujú bod kontaktu nohy s podložkou. Šípka nad robotom predstavuje smer pohybu.



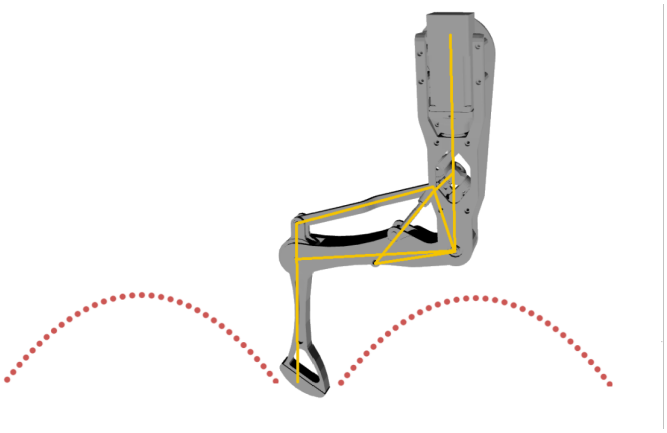
Obr. 4.3: Chôdza vzad s naplánovaným krivkami pre jednotlivé nohy. Kruhy na koncoch naplánovaných kriviek predstavujú bod kontaktu nohy s podložkou. Šípka nad robotom predstavuje smer pohybu.



Obr. 4.4: Chôdza vbok s naplánovaným krivkami pre jednotlivé nohy. Kruhy na koncoch naplánovaných kriviek predstavujú bod kontaktu nohy s podložkou. Šípka nad robotom predstavuje smer pohybu.

4.1 Priame pozičné ovládanie

Motory robota vieme ovládať priamo, kedy im nastavíme pozíciu natočenia. Máme funkciu, ktorá nám generuje súradnice bodu pre každú nohu v časovom bode, do ktorého sa má noha dostať. Tieto body sa nachádzajú na krivke po ktorej sa má noha hýbať pri chôdzi, viď. Obr. 4.5, pomocou inverznej kinematiky vypočítame uhly natočenia motorov, ktoré potom pošleme ovládaču motora. Na takéto ovládanie potrebujeme poznať inverznú kinematiku, tú však už pre Artabana poznáme poznáme. Túto metódu sme otestovali v simulácií a aj na hardvéri.



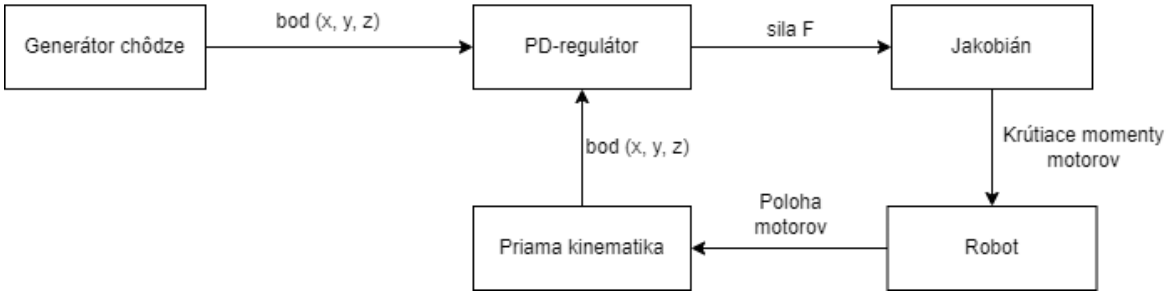
Obr. 4.5: Zobrazenie zadnej nohy a jej trajektórie.

Na hardvéri táto metóda nefungovala ideálne, keďže jej chýba akákoľvek spätná väzba a keď sa robot z nejakého dôvodu naklonil alebo inak vychýlil z plánu, čo sa vždy stalo, tak spadol. Taktiež náš model robota nie je dokonalý a aj to malo vplyv na jeho chôdzu. Keď bol robot zavesený na lanách tak sme týmto spôsobom validovali aspoň kinematiku, rozsah nôh a funkčnosť a schopnosť hardvéru sa hýbať po želaných trajektóriách. Taktiež sme videli potrebu riadenia robota so spätnou väzbou o stave

robota a následnou úpravou plánu pohybu robota na základe meraní zo senzorov o stave robota.

Netestovali sme túto metódu ďalej, ale pozičné ovládanie nie je ideálny spôsob riadenia robota a všeobecne sa využíva ovládanie pomocou krútiaceho momentu. Častokrát potrebujeme aby nohy robota rovnomerne pôsobili na podložku nejakou silou, napríklad keď robot stojí, a nestačí len správna poloha nôh.

4.2 Ovládanie pomocou krútiaceho momentu a PD-regulátora



Obr. 4.6: Priebeh riadenia robota pomocou PD-regulátora a krútiacich momentov motorov.

Druhý spôsob ovládania je znázornený na Obr. 4.6. Znova sme generovali krivky, po ktorých sa má noha hýbať. Jednotlivé body z týchto kriviek sme ďalej posielali do PD-regulátora spolu s aktuálnom pozíciou koncov nôh. Tie sme získali z priamej kinematiky a pozícií motorov, ktoré získavame z ovládačov motorov. Výstup z PD-regulátora predstavoval silu, ktorou musí daná noha pôsobiť. Z tejto sily pomocou jakobiánu vieme vypočítať krútiace momenty pre jednotlivé motory, ktoré sme poslali rozhraniu pred hardvér.

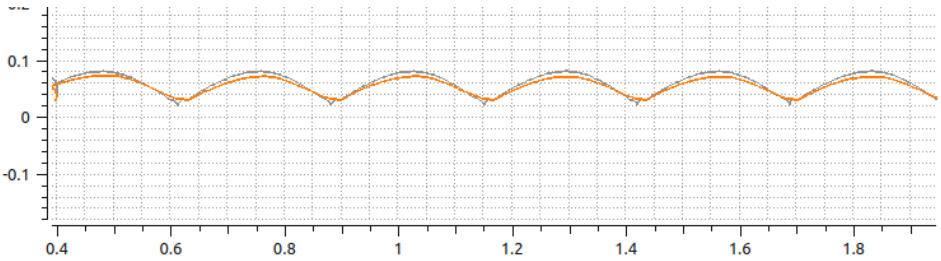
Tento spôsob riadenia je o niečo lepší ako priame pozičné ovládanie. Dôležité je správne nastavenie koeficientov PD-regulátora, bez nich robot nie je schopný ani stáť. Po ich nastavení robot zvládol samostatne stáť a pohybovať sa dopredu aj dozadu. Tie sme sa najprv snažili nastaviť v simulácií a potom preniesť na hardvér, avšak kvôli všetkým treniam, nepresnostiam pri výrobe a iným špecifikám hardvéru hodnoty PD-regulátora v simulácií a na hardvéri sa líšili násobne.

Keďže je tu určitý typ spätnej väzby, tak aj rozdiely medzi skutočným hardvérom a modelom robota sme zvládali preklenúť. Stále však šlo o nestabilný typ chôdze. Pri generovaní kriviek sa nezohľadňuje stav robota a ani žiadne vonkajšie vplyvy, preto v konečnom dôsledku by robot žiadne postrčenie ani nerovnosť nezvládol a spadol by. Taktiež robot nenasledoval naplánované krivky ideálne a bol schopný iba veľmi malých a krátkych krokov.

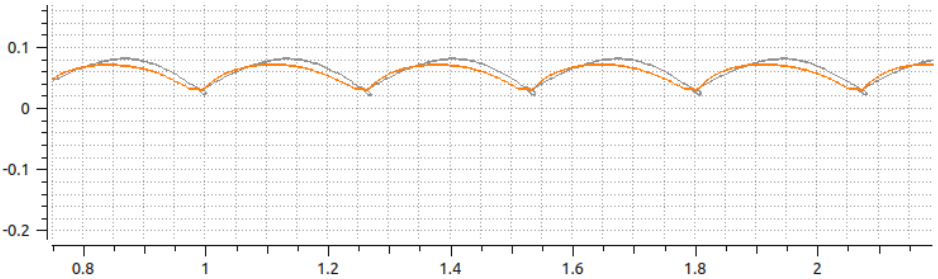
4.3 Quad SDK bez inverznej dynamiky

Toto riešenie sme validovali aj v simulácií aj na hardvéri. Slúžilo hlavne na overenie, či je možné Quad SDK upraviť na Artabana a využiť pre adaptívne riadenie predtým než sme začali odvodzovať a implementovať inverznú dynamiku. V simulácií sme najprv vyladili chôdzu, teda nastavili optimálnu výšku tela a dĺžku kroku, a potom sme ju skúšali na hardvéri. Testovali sme chôdzu vpred, otáčanie sa na mieste, kráčanie vbok a chôdzu vzad.

Najprv sme testovali chôdzu klusom. Robot mal v kontakte s podložkou 2 nohy, zatiaľ čo zvyšnými dvomi robil pohyb vpred. Pri krátkom kroku robot robí kratšie ale rýchlejšie kroky, pri dlhom zas dlhšie a pomalšie. Na Obr. 4.7 vidíme ako nohy robota sledovali naplánované trajektórie pri chôdzi vpred klusom pri krátkom kroku. Na obrázku sú zobrazené krivky nôh pri chôdzi vpred a pohľade zboku, tak ako vidíme na Obr. 4.1. číselné údaje na súradnicových osiach sú v metroch. Toto platí aj pre všetky nadchádzajúce porovnanie plánovaných a skutočných kriviek nôh. Robot zvládol v simulácii pre tento typ chôdze stabilne všetky testované pohyby a aj si držal stabilnú polohu tela v požadovanej výške.



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

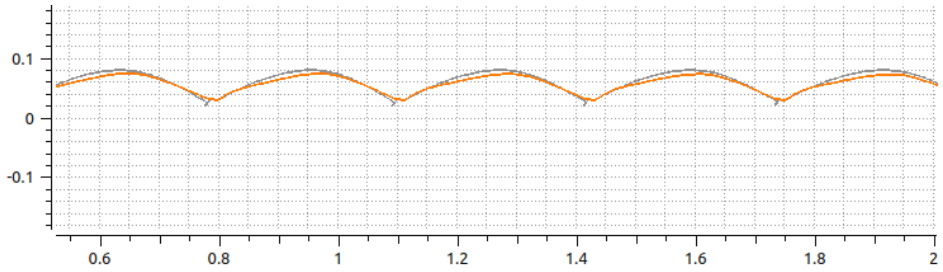


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

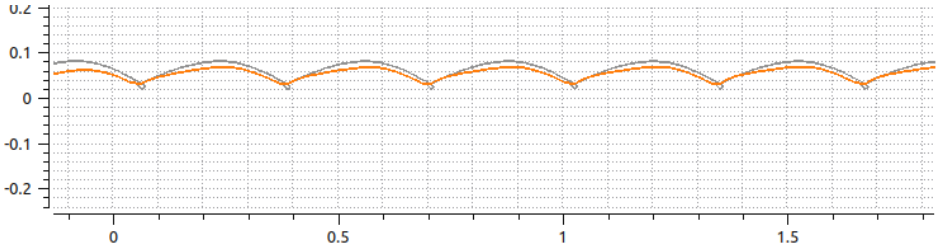
Obr. 4.7: Porovnanie plánovej a skutočnej trajektórie v simulácii pre nohy pri chôdzi vpred klusom pri kratšom kroku.

Na Obr. 4.8 vidíme ako sa nohy správali pri dlhom kroku. Robot sa pohyboval stabilne a podľa vypočítaného plánu pri všetkých typoch pohybu.

Následne sme skúsili chôdzu krokom, teda v kontakte s podložkou boli vždy aspoň



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.



(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.8: Porovnanie plánovej a skutočnej trajektórie v simulácii pre nohy pri chôdzi vpred klusom pri dlhšom kroku.

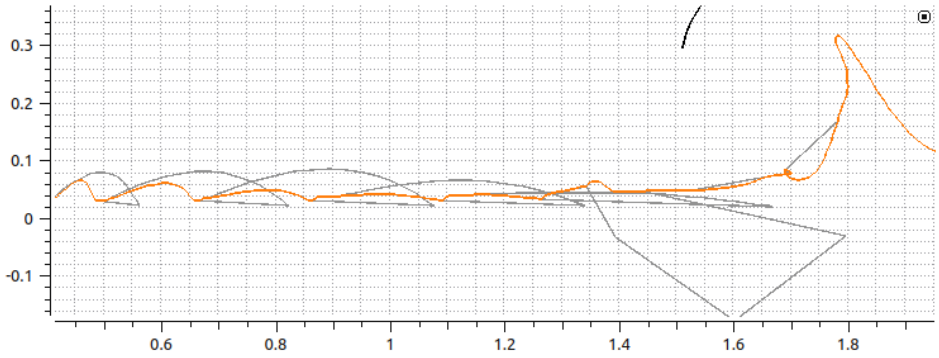
3 nohy. Na Obr. 4.9 vidíme, že robotov mal problémy ísť dopredu a spadol. Nestíhal sledovať naplánované krivky a zvládol iba prvých pár krokov. Pri ostatných typoch pohybov nezvládal robot ani pár krokov.

Chôdzu krokom sme skúsili aj s dlhším krokom. ako vidíme na Obr. 4.10. Robot zvládol o pár krokov viac predtým než spadol, ale stále to nie je stabilná chôdza, ktorú by bolo bezpečné testovať na hardvéri. Dlhší krok sme už neskúšali keďže vidíme, že robot má problém robiť také veľké kroky.

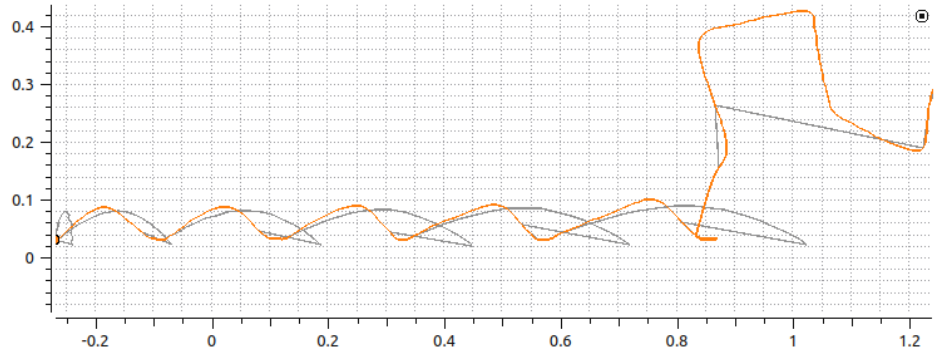
Nakoniec sme v simulácii otestovali schopnosť robota ísť cez prekážky konkrétne schody ako vidíme na Obr. 4.11. Nastavili sme mu chôdzu klusom a krátky krok, keďže pri týchto parametroch zvládal rôzne typy pohybov najlepšie na rovnej ploche. Robot zvládol vyjsť po hore po schodoch smerom dolu však spadol, ako môžeme vidieť na Obr. 4.12.

V simulácií robot chodil relatívne pekne a stabilne, avšak už tu sme pozorovali, že napríklad pri dlhom kroku robot strácal stabilitu a padal.

Na hardvéri sme skúšali iba chôdzu, ktorá bola v simulácii stabilná, teda chôdzu klusom. Z Obr. 4.13 vidíme, že robot zvládal chodiť a zvládol aj iné základné pohyby. Môžeme však vidieť výrazné rozdiely medzi hardvérom a softvérom. Tie sú spôsobené rozdielmi v hmotnosti jednotlivých častí robota medzi modelom a realitou. Ďalej rozdielnymi rozmermi jednotlivých častí robota, ktoré vznikli pri výrobe alebo opotrebovaním. Najväčším faktorom sú však pravdepodobne trenia vznikajúce v nohách robota,



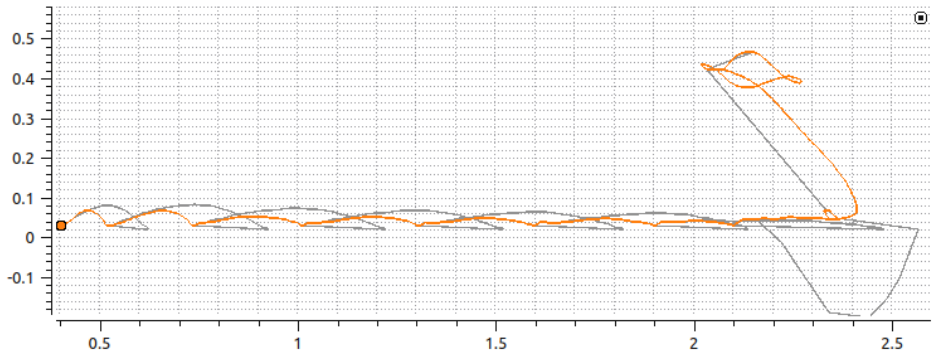
(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.



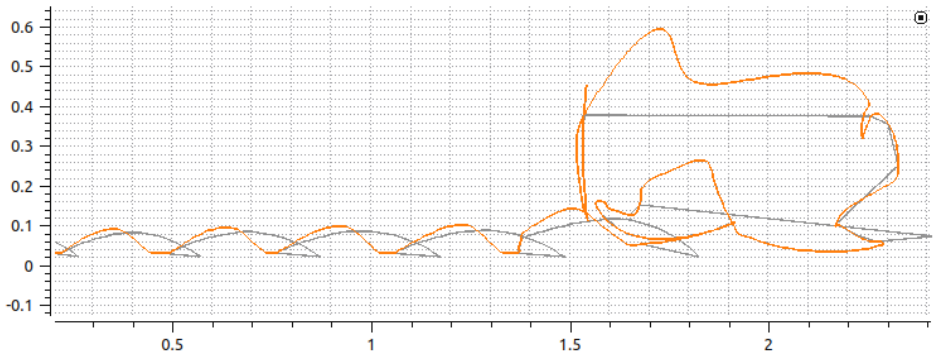
(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.9: Porovnanie plánovej a skutočnej trajektórie v simulácii pre nohy pri chôdzi vpred krokom pri kratšom kroku.

s ktorými v simulácia nepočíta. Oproti simulácii bolo potrebné zmeniť aj koeficienty PD-regulátora, keďže iba pomocou výstupu z neho sa prekonávajú trenia.

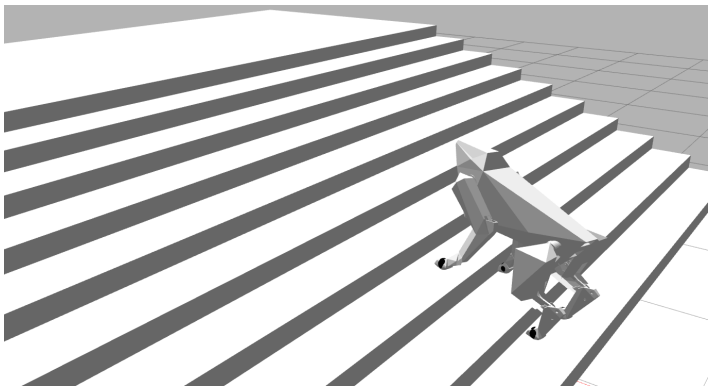


(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

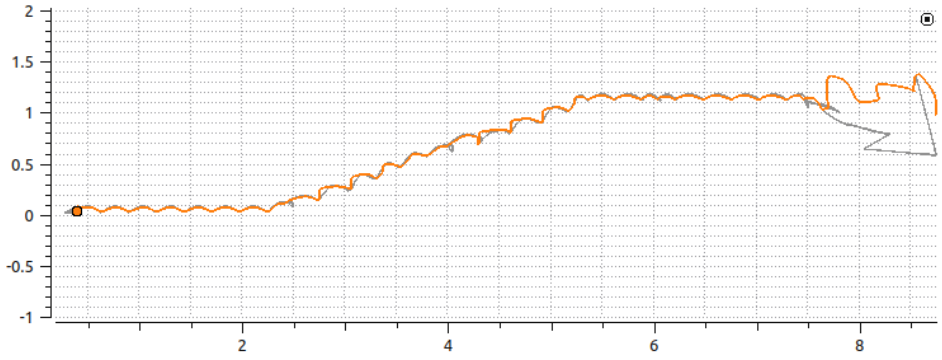


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

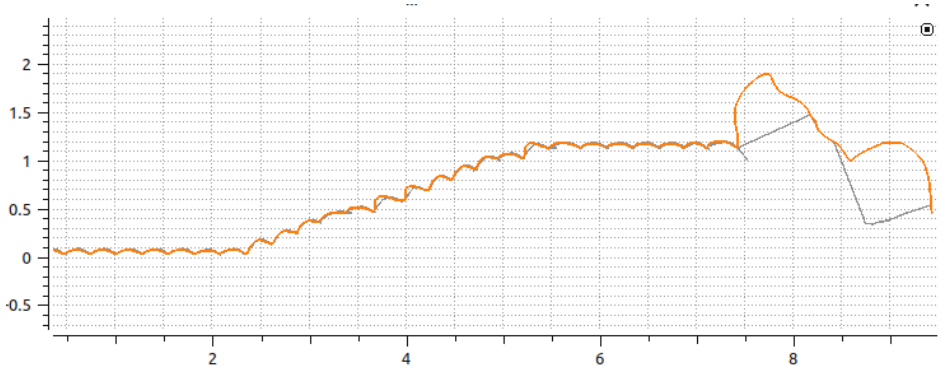
Obr. 4.10: Porovnanie plánovej a skutočnej trajektórie v simulácii pre nohy pri chôdzi vpred krokom pri dlhšom kroku.



Obr. 4.11: Robot Artaban v simulácii na schodoch bez inverznej dynamiky.

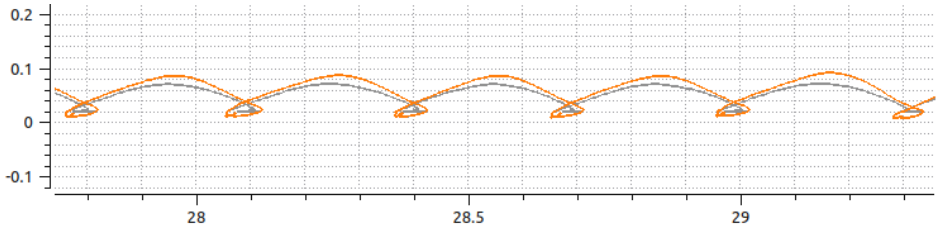


(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

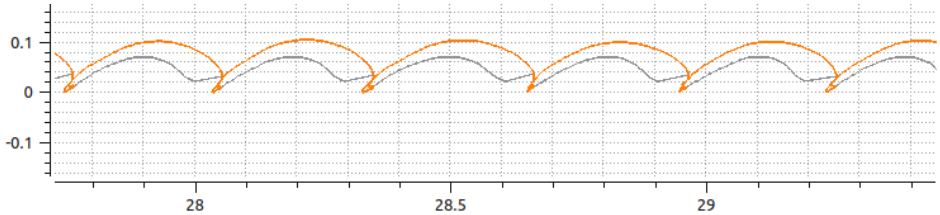


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.12: Porovnanie plánovej a skutočnej trajektórie v simulácii pre nohy pri chôdzi po schodoch.



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.



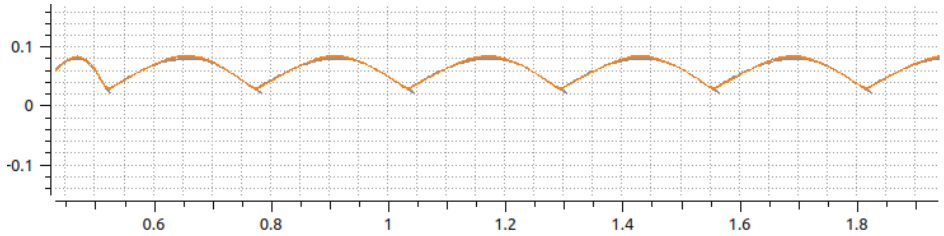
(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.13: Porovnanie plánovej a skutočnej trajektórie na hardvéri pre nohy pri chôdzi.

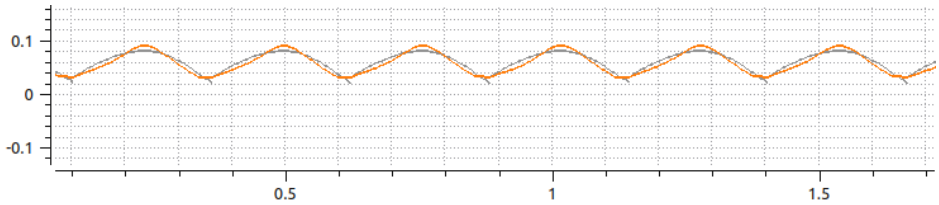
4.4 Quad SDK s inverznou dynamikou všetkých nôh bez kompenzácie trení a zotrvačností

Po validovaní funkčnosti a použiteľnosti knižnice Quad SDK pre robota Artabana, sme odvodili a implementovali inverznú dynamiku a ďalej sme testovali s inverznou dynamikou.

Najprv sme v simulácii otestovali chôdzu klusom pre kratší krok. Na Obr. 4.14 vidíme isté zlepšenie, ale nie je také výrazné, keďže aj bez inverznej dynamiky tento typ chôdze robot zvládal.



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.



(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.14: Porovnanie plánovej a skutočnej trajektórie v simulácii pri chôdzi klusom s inverznou dynamikou a krátkym krokom.

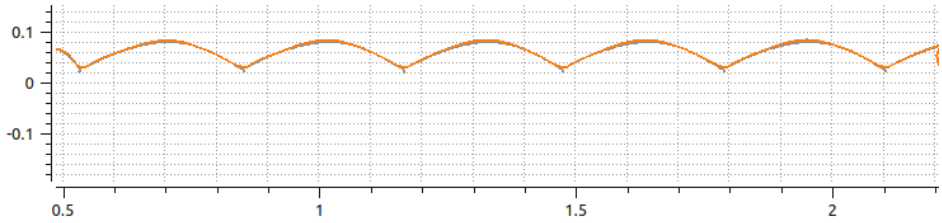
Skúsili sme aj chôdzu klusom pre dlhý krok, Obr. 4.15, robot znova zvládol všetky typy pohybov stabilne.

Zaujímavejšie výsledky sme videli pri chôdzi krokom. Na Obr. 4.16 vidíme, že aj keď robot nesledoval trajektórie nôh ideálne, už zvládal kráčať vpred bez pádu. Zvládal taktiež cúvať ale pri otáčavých pohyboch a pohyboch do boku už strácal stabilitu.

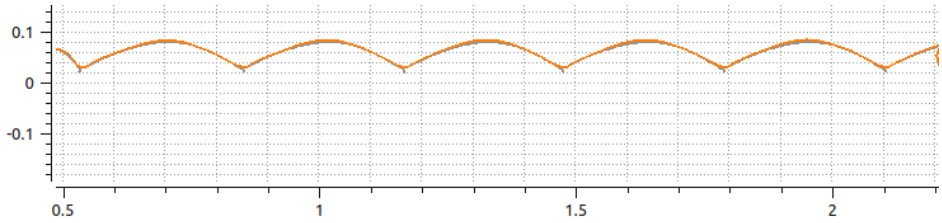
Na Obr. 4.17 vidíme krivky nôh pri dlhom kroku chôdzou. Nohy robota nenasledovali naplánované krivky ideálne, ale robot aj tak zvládol všetky typy pohybov. Tu vidíme najväčšie zlepšenie oproti chôdzi bez inverznej dynamiky v simulácii.

Nakoniec sme skúsili aj či robot vie vyjsť po schodoch. Na Obr. 4.18 vidíme robota na schodoch aj s naplánovanými trajektóriami pre jednotlivé nohy.

Smerom hore to zvládol veľmi dobre, ako vidíme na Obr.4.19. Síce v strede schodov zakopol ale zvládol sa z toho spamätať. Dolu schodmi však stabilne nezvládol ísť. Pre

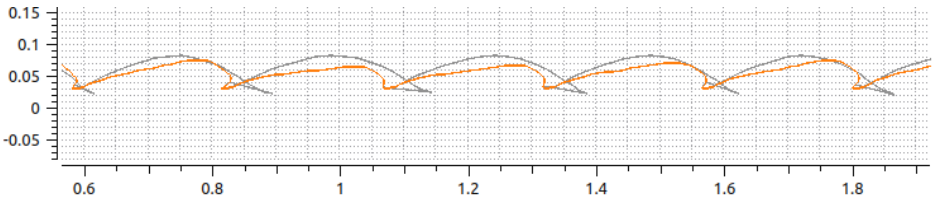


(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

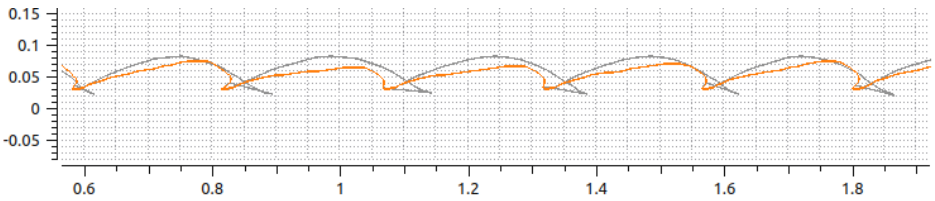


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.15: Porovnanie plánovej a skutočnej trajektórie v simulácii pri chôdzi klusom s inverznou dynamikou a dlhým krokom.



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

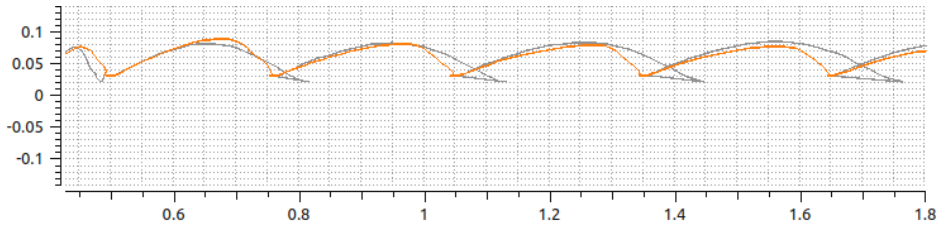


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

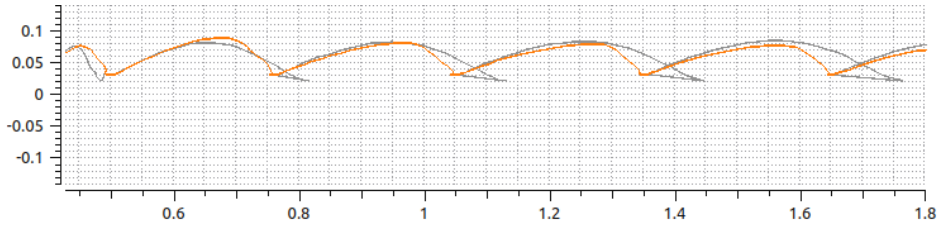
Obr. 4.16: Porovnanie plánovej a skutočnej trajektórie v simulácii pri chôdzi krokom s inverznou dynamikou a krátkym krokom.

chôdzu dolu schodmi je ešte potrebné nájsť ideálne parametre kroku, aby robot zvládol ísť po schodoch smerom dolu. Prípadne otestovať, či robot zvláda zísť dolu po schodoch cúvaním, keďže to by mohla byť preňho stabilnejšia poloha.

Chôdzu klusom sme znova otestovali aj na hardvéri. Videli sme zlepšenie, Obr. 4.20, predné nohy nasledujú naplánovanú trajektóriu veľmi dobre, pri zadných je isté zlepšenie ale stále nenasledujú naplánovanú trajektóriu tak ako predné nohy.

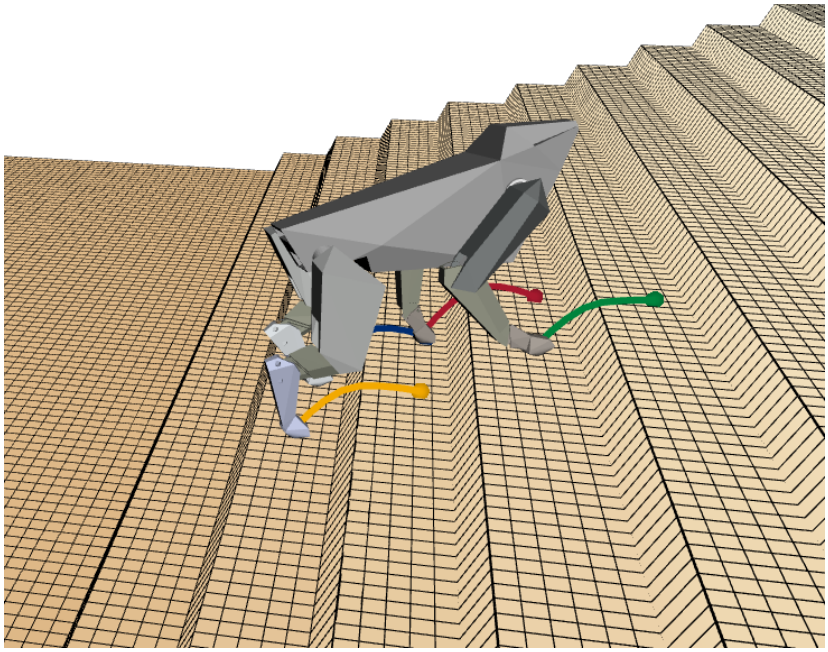


(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

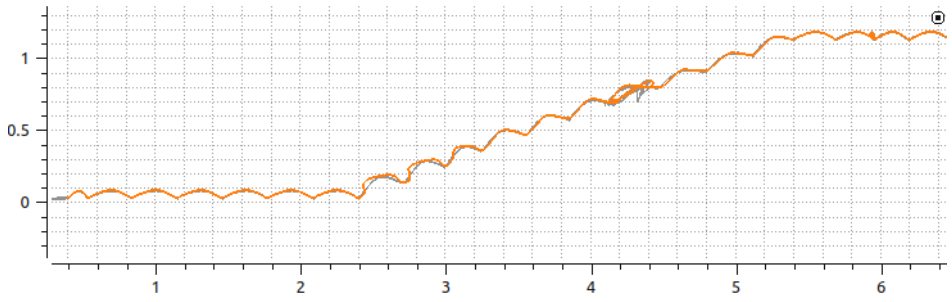


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

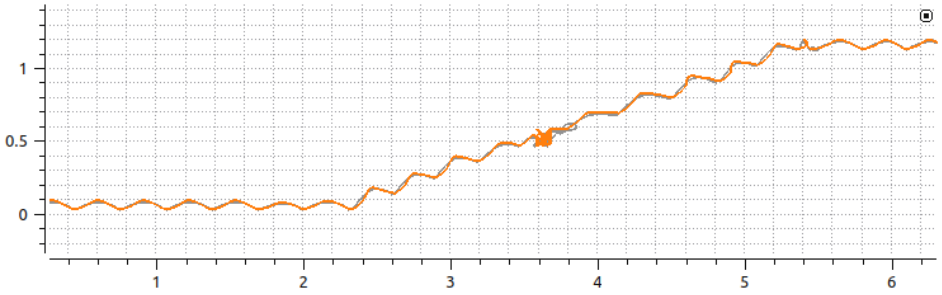
Obr. 4.17: Porovnanie plánovej a skutočnej trajektórie v simulácii pri chôdzi krokom s inverznou dynamikou a dlhým krokom.



Obr. 4.18: Robot Artaban zobrazený na schodoch aj s naplánovanými trajektóriami nôh.

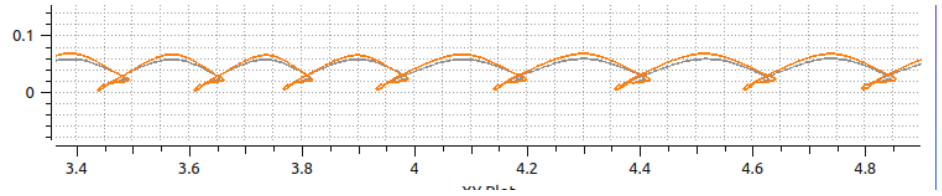


(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

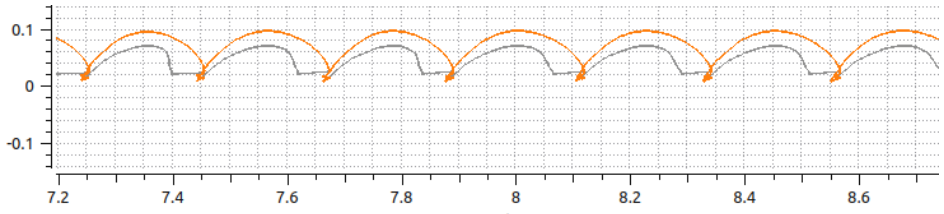


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.19: Porovnanie plánovej a skutočnej trajektórie v simulácii pri chôdzi inverznou dynamikou hore schodmi. Vidíme ako robot postupne stúpal po schodoch s jedným zakopnutím v strede.



(a) Predná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

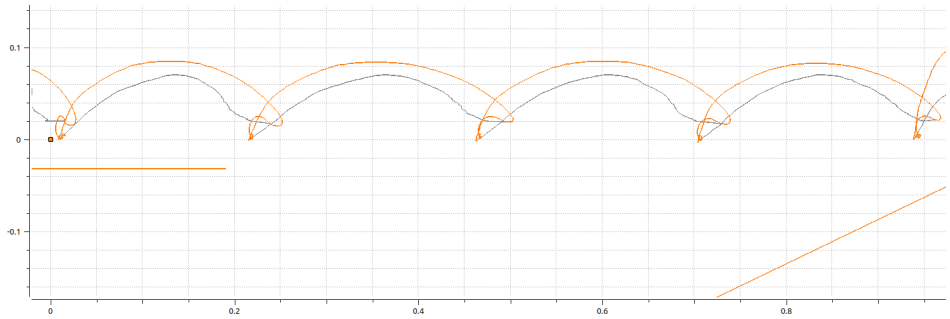


(b) Zadná noha, sivou je znázornená plánovaná trajektória a oranžovou skutočná.

Obr. 4.20: Porovnanie plánovej a skutočnej trajektórie na hardvéri pri chôdzi s inverznou dynamikou.

4.5 Quad SDK s inverznou dynamikou všetkých nôh s kompenzáciou trení a zotrvačností

Všimli sme si, že aj napriek pridanej inverznej dynamike, stále sú veľké rozdiely medzi hardvérom a simuláciou. Preto sme sa rozhodli kompenzovať trenia a zotrvačnosti na motoroch a prevodovkách. Samozrejme trenia vznikajú aj v kardanovom mechanizme a v kĺboch, ktoré nie sú poháňané, avšak tie nevieme zatiaľ merať a teda ani nijako modelovať pri riadení.



Obr. 4.21: Porovnanie plánovej a skutočnej trajektórie na hardvéri pre zadnú pri chôdzi s inverznou dynamikou a kompenzáciou trení a zotrvačností.

Kompenzovať trenia a zotrvačnosti motorov a prevodoviek sa ukázalo ako veľmi dobrý krok, porovnanie vidíme na Obr.4.21 a preto chceme v budúcnosti tento model rozšíriť aj o ďalšie parametre ako napríklad teplota, či efektivita prevodoviek v záťaži.

Nakoniec sme sa rozhodli otestovať aj schopnosť robota zvládnuť nečakané postrčenia. Na Obr. 4.22 vidíme ako sa robot zvládol aj z takéhoto nečakaného postrčenia spamätať a nadobudnúť znova rovnováhu.

Obr. 4.22: Priebeh postrčenia do robota.

4.6 Diskusia

Adaptívne riadenie nám pomohlo vykonávať pohyby, ktoré robot pri jednoduchších spôsoboch riadenia nezvládal. Nižšie vidíme porovnanie v Tab. 4.1, kde sme jednotlivé spôsoby porovnali pre dosiahnuté výsledky na hardvéri. Riadenie pomocou Quad SDK s implementovanou inverznou dynamikou sa osvedčil najviac. Spätná väzba je pri adaptívnom riadení veľmi dôležitá a bez nej nevieme dosiahnuť tak dobré výsledky.

	vpred	**otáčanie sa**	**vzad**	**vbok**	**postrčenie**
Pozičné ovládanie	×	×	×	×	×
PD-regulátor	✓	×	✓	×	×
Quad SDK bez ID	✓	✓	✓	✓	×
Quad SDK s ID	✓	✓	✓	✓	✓

Tabuľka 4.1: Porovnanie jednotlivých riadiacich metód.

Záver

V tejto práci sme sa venovali adaptívnemu riadeniu robota Artabana. Naším cieľom bolo implementovať a otestovať takýto spôsob riadenia, čo sa nám aj podarilo. Základné pohyby robota sme otestovali aj v simulácii aj na hardvéri.

Začali sme tým, že sme sa pozreli na spôsoby riadenia iných robotov a aké sú dostupné technológie. Nakoniec sme sa rozhodli využiť na riadenia Artabana knižnicu Quad SDK. Je verejne dostupná, dobre zdokumentovaná a nie je písaná pre jedného konkrétneho robota. Taktiež je dôležité, že do riadiaceho procesu vstupuje aj okolitý terén a robot sa mu vie prispôsobiť.

Ďalším krokom bolo odvodenie a implementovanie funkcií, ktoré sú špecifické pre Artabana, kvôli jeho unikátnej stavbe zadných nôh. Potrebovali sme poznať priamu a inverznú kinematiku, ktorú sme však odvodili už v bakalárskej práci. Z priamej kinematiky sme odvodili jakobiány. Ďalej sme odvodili inverznú dynamiku, čo je netriviálna úloha pre systémy s kinematickými slučkami.

Po implementácii potrebných pomocných funkcií sme začali s testovaním v simulácii. Vyladili sme potrebné parametre, ako koeficienty PD-regulátora, dĺžka a typ kroku.

Keď sme overili funkčnosť riadenia v simulácii začali sme s testovaním na hardvéri. Oproti simulácii robot chodil horšie, teda menej dodržiaval naplánovanú trajektóriu nôh. To je spôsobené rôznymi špecifikami hardvéru. Napríklad treniami v rôznych častiach robota, určitou efektivitou prevodoviek a zotrvačnosťou motorov a prevodoviek. Taktiež hardvér má určité nepresnosti, napríklad v dĺžkach jednotlivých segmentov nohy. Konkrétne pár milimetrov na segmente, ktorý je súčasťou 4-tyčového mechanizmu môže spôsobiť v krajnom prípade až niekoľko centimetrov na nohe. Tieto rozdiely vedia spôsobiť aj jav, kedy robot nedostúpi na zem podľa plánu. Existujú metódy ako odhadovať, či je noha v kontakte s podložkou alebo nie, na ktoré by sme sa chceli v budúcnosti určite pozrieť.

Keď sme prekonali tieto rozdiely medzi hardvérom a softvérom, robot bol schopný stabilnej chôdze vpred, vedel sa otáčať, cúvať a aj kráčať do boku. Taktiež vedel ustáť postrčenia a znovunadobudnúť rovnováhu. Úspešne sa nám podarilo implementovať a aj otestovať adaptívne riadenie robota, nielen v simulácii ale aj na hardvéri, čo vždy predstavuje dodatočné výzvy.

V budúcnosti chceme toto riadenie ďalej vylepšovať, napríklad o odhad či je noha

v kontakte s podložkou alebo nie, vyladiť ešte viac parametre chôdze a začať testovať aj hardvér robota na nerovnom teréne. Taktiež by sme sa radi pozreli aj na spôsoby riadenia pomocou učenia formou odmeňovania (Reinforcement learning), prípadne ako kombináciu s aktuálnym riadením.

Literatúra

[1] Boston Dynamics. Spot, 2023. [Citované 2023-11-20] Dostupné z `https://bostondynamics.com/products/spot/`.

[2] Champ. champ. [Citované 2023-11-23] Dostupné z `https://github.com/chvmp/champ`.

[3] Jennifer Chu. Mini cheetah is the first four-legged robot to do a backflip. MIT, 2019. [Citované 2023-11-22] Dostupné z `https://news.mit.edu/2019/mit-mini-cheetah-first-four-legged-robot-to-backflip-0304`.

[4] Thomas Corbères, Carlos Mastalli, Wolfgang Merkt, Ioannis Havoutis, Maurice Fallon, Nicolas Mansard, Thomas Flayols, Sethu Vijayakumar, and Steve Tonneau. Perceptive locomotion through whole-body mpc and optimal region selection. *arXiv preprint arXiv:2305.08926*, 2023.

[5] Jared Di Carlo. Software and control design for the mit cheetah quadruped robots. Master's thesis, Massachusetts Institute of Technology, 2020.

[6] Jared Di Carlo, Patrick M. Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018.

[7] Yanran Ding, Abhishek Pandala, and Hae-Won Park. Real-time model predictive control for versatile dynamic motions in quadrupedal robots. In *International Conference on Robotics and Automation (ICRA)*, pages 8484–8490. IEEE, 2019.

[8] Péter Fankhauser and Marco Hutter. A Universal Grid Map Library: Implementation and Use Case for Rough Terrain Navigation. In Anis Koubaa, editor, *Robot Operating System (ROS) – The Complete Reference (Volume 1)*, chapter 5. Springer, 2016.

[9] Farbod Farshidian. Ocs2. Github. [Citované 2023-11-23] Dostupné z `https://github.com/leggedrobotics/ocs2/tree/main`.

[10] Roy Featherstone. *Rigid Body Dynamics Algorithms*, pages 161–165. Springer New York, NY, 2007.

[11] Martin L. Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. In *Autonomous Robots*, pages 1–17. Springer Netherlands, 2016.

[12] Ruben Grandia, Fabian Jenelten, Shaohui Yang, Farbod Farshidian, and Marco Hutter. Perceptive locomotion through nonlinear model predictive control. ETH Zurich, 08 2022.

[13] Ruben Grandia, Andrew J. Taylor, Aaron D. Ames, and Marco Hutter. Multi-layered safety for legged robots via control barrier functions and model predictive control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8352–8358. IEEE, 2021.

[14] Felix Grimminger, Avadesh Meduri, Majid Khadiv, Julian Viereck, Manuel Wüthrich, Maximilien Naveau, Vincent Berenz, Steve Heim, Felix Widmaier, Thomas Flayols, Jonathan Fiene, Alexander Badri-Spröwitz, and Ludovic Righetti. An Open Torque-Controlled Modular robot architecture for legged locomotion research. volume 5, pages 3650–3657. IEEE, 4 2020.

[15] Joe Norby, Yanhao Yang, Ardalan Tajbakhsh, Jiming Ren, Justin K. Yim, Alexandra Stutt, Qishun Yu, Nikolai Flowers, and Aaron M. Johnson. Quad sdk. [Citované 2023-11-23] Dostupné z `https://github.com/robomechanics/quad-sdk`.

[16] Lentin Joseph. *Mastering ROS for robotics programming : design, build, and simulate complex robots using robot operating system and master its out-of-the-box functionalities*, page 7. Packt Publishing, 2015.

[17] D. Kim, Jared Di Carlo, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Highly dynamic quadruped locomotion via Whole-Body impulse control and model predictive control. *arXiv (Cornell University)*, 2019.

[18] MIT Biomimetic Robotics Lab. Cheetah-software. [Citované 2023-11-23] Dostupné z `https://github.com/mit-biomimetics/Cheetah-Software`.

[19] Jongwoo Lee. Hierarchical controller for highly dynamic locomotion utilizing pattern modulation and impedance control : implementation on the mit cheetah robot. Master's thesis, Massachusetts Institute of Technology, 2013.

[20] C. Mastalli, R. Budhiraja, W. Merkt, G. Saurel, B. Hammoud, M. Naveau, J. Carpentier, L. Righetti, S. Vijayakumar, and N. Mansard. Crocoddyl: An Efficient

and Versatile Framework for Multi-Contact Optimal Control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2536–2542. IEEE, 2020.

[21] Carlos Mastalli, Saroj Chhatoi, Thomas Corberes, Steve Tonneau, and Sethu Vijayakumar. Inverse-dynamics mpc via nullspace resolution. In *IEEE Transactions on Robotics*, pages 3222–3241. IEEE, 2022.

[22] Zuzana Mačicová. Implementácia aproximácie priamej a inverznej kinematiky paralelného mechanizmu nôh robota artaban. Bachelor thesis, Comenius University in Bratislava, 2022.

[23] Takahiro Miki, Lorenz Wellhausen, Ruben Grandia, Fabian Jenelten, Timon Homberger, and Marco Hutter. Elevation mapping for locomotion and navigation using gpu. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2273–2280. IEEE, 2022.

[24] Joseph Norby, Yanhao Yang, Ardalan Tajbakhsh, Jiming Ren, Justin K. Yim, Alexandra Stutt, Qishun Yu, Nikolai Flowers, and Aaron M. Johnson. Quad-SDK: Full stack software framework for agile quadrupedal locomotion. In *ICRA Workshop on Legged Robots*. IEEE, May 2022.

[25] Open Robotics. Gazebo. [Citované 2023-11-23] Dostupné z `https://gazebosim.org/doc`.

[26] Open Robotics. ROS - ROS Wiki. [Citované 2023-11-22] Dostupné z `https://wiki.ros.org/ROS`.

[27] Fanny Risbourg, Thomas Corbères, Pierre-Alexandre Léziart, Thomas Flayols, Nicolas Mansard, and Steve Tonneau. Real-time footstep planning and control of the solo quadruped robot in 3D environments. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022.

[28] Chin Pei Tang. Lagrangian dynamic formulation of a four-bar mechanism with minimal coordinates. March, 2006.

[29] The MathWorks, Inc. What is Model Predictive Control? - MATLAB and Simulink. [Citované 2023-11-20] Dostupné z `https://www.mathworks.com/help/mpc/gs/what-is-mpc.html`.