České Vysoké Učení Technické v Praze
Fakulta strojní
Ústav mechaniky, biomechaniky a mechatroniky
Obor: Robotika a výrobní technika.

Simulace kráčení robota po vesmírné stanici

DIPLOMOVÁ PRÁCE

Vypracoval: Bc. Jan Černý
Vedoucí práce: prof. Ing. Michael Valášek, DrSc.
Rok: 2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Černý** Jméno: **Jan** Osobní číslo: **483267**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Robotika a výrobní technika**

Specializace: **Robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Simulace kráčení robota po vesmírné stanici

Název diplomové práce anglicky:

Simulation of a robot walking around a space station

Pokyny pro vypracování:

Vytvoření simulace pohybu robota po vesmírné stanici kráčením a optimalizace jeho pohybu
Cíle:
1. Proveďte rešerši robotů ve vesmíru
2. Vytvořte koncept kráčejícího robota na vesmírné stanici
3. Vytvořte simulační model kráčejícího robota
4. Vyřešte inverzní kinematickou úlohu robota při kráčení
5. Vyřešte inverzní dynamickou úlohu robota při kráčení
6. Vyřešte simulaci pohybu robota se stabilizací pohybu
7. Optimalizujte velikost potřebných sil pro pohyb robota

Seznam doporučené literatury:

- Sciavicco, L., Siciliano, B.: Modeling and control of robot manipulators, McGraw Hill, 1996
- Stejskal, V., Valasek, M.: Kinematics and dynamics of machinery, Marcel Dekker, New York, 1996

Jméno a pracoviště vedoucí(ho) diplomové práce:

prof. Ing. Michael Valášek, DrSc. centrum leteckého a kosmického výzkumu FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **27.04.2023** Termín odevzdání diplomové práce: **14.08.2023**

Platnost zadání diplomové práce: _____

prof. Ing. Michael Valášek, DrSc. podpis vedoucí(ho) práce	prof. Ing. Michael Valášek, DrSc. podpis vedoucí(ho) ústavu/katedry	doc. Ing. Miroslav Španiel, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____ Podpis studenta

Datum převzetí zadání

Prohlášení

Prohlašuji, že jsem svou diplomovou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne .....

.....
Bc. Jan Černý

Poděkování

Děkuji prof. Ing. Michaelu Valáškovi, DrSc. za veškerý věnovaný čas, užitečné rady a návrhy. Dále bych chtěl poděkovat své rodině za trpělivost a velkou podporu.

Bc. Jan Černý

Název práce:
Simulace kráčení robota po vesmírné stanici

Autor: Bc. Jan Černý

Druh práce: Diplomová práce

Vedoucí práce: prof. Ing. Michael Valášek, DrSc.

Abstrakt: Tato práce se zabývá simulací kráčení robota na vesmírné stanici. V simulaci je řešena inverzní kinematika robota, následně ze známých pohybů je vypočítána inverzní dynamika. Bylo navrženo řízení pohonů přes kaskádní regulaci a poté proběhla optimalizace momentů robota na určený pohyb.

Klíčová slova: Robot, Inverzní kinematika, Inverzní dynamika, Řízení

Title:
Simulation of a walking robot on space station

Author: Bc. Jan Černý

Druh práce: Masters's thesis

Supervisor: prof. Ing. Michael Valášek, DrSc.

Abstract: This paper deals with the simulation of a robot walking on a space station. In the simulation, the inverse kinematics of the robot is solved, then the inverse dynamics is calculated from the known motions. Control of the actuators via cascade control has been proposed and then optimization of the robot moments for the specified motion has been performed.

Key words: Robot, Inverse kinematics, Inverse dynamics, Control

Obsah

Úvod

408 kilometrů nad Zemí se lidem obtížně pohybuje a je potřeba dbát na veliké nebezpečí vzduchoprázda. Protože ochranné skafandry a veškeré bezpečností zajištění astronautů jsou velice drahé a náročné na údržbu, je zde snaha o zavedení robotiky. Robot uvedený v této práci má za úkol pohybovat se po vesmírné stanici a přemísťovat věci na určená místa.

Pro předem určený pohyb celého robota je nutné dopočítat pomocí inverzní kinematiky natočení jeho jednotlivých pohonů a následně vypočítat příslušné síly v těchto pohonech tak, aby daný pohyb byl za daných podmínek proveditelný.

Motivace

Motivací této práce je návrh kráčejícího robota na vesmírné stanici, aby pomohl s přídavnými pracemi na stanici. Jeho návrh a simulace jsou provedeny v prostředí Matlab/Simulink s využitím knihovny Simscape Multibody.

Cíle práce

Vlastní práce se zabývá naplněním následujících cílů:

1. Proveďte rešerši robotů ve vesmíru.
2. Vytvořte koncept kráčejícího robota na vesmírné stanici.
3. Vytvořte simulační model kráčejícího robota.
4. Vyřešte inverzní kinematickou úlohu robota při kráčení.
5. Vyřešte inverzní dynamickou úlohu robota při kráčení.
6. Vyřešte simulaci pohybu robota se stabilizací pohybu.
7. Optimalizujte velikost potřebných sil pro pohyb robota.

Kapitola 1

Roboti ve vesmíru

Tato kapitola se zabývá roboty na vesmírné stanici ISS.

1.1 Canadarm2

Canadarm2 je robotické rameno sloužící k přepravování těžkých součástí na vesmírné stanici nebo pro zachytávání objektů, jako jsou družice nebo raketoplány s posádkou, Je to robot se sedmi stupni volnosti, jehož klouby jsou poháněny elektrickými motory o výkonu 1,2 kW. Jeho hlavní výhodou jsou úchopné hlavice na obou koncích ramene ,které lze použít pro manipulaci s předměty. Způsob přesouvání ramene spočívá v řízeném pohybu volné úchopné hlavice do místa, kde dojde k jejímu následnému přichycení ke stanici. Poté se uvolní aktuální přichycení a volný konec ramene se přesune do dalšího místa úchopu. Spolehlivý úchop je umožněn pomocí kabelů, které se přichytí ke stanici nebo k zachycovanému předmětu. Rameno měří 17 metrů a váží 1,8 tuny [1].

Na obr. 1.1 je zobrazeno robotické rameno Canadarm2. Stanice se v tomto okamžiku nachází nad Mongolskem.

Canadarm2 má však následovníka s označením Canadarm3. Bude sloužit na stanici Gateway, která má obíhat kolem Měsíce a vykonávat podobné úkony jako stanice ISS. Předpokládaný termín dokončení stanice Gateway je rok 2026 a Canadarm3 bude připojen o rok později. Třetí generace robotického ramene bude ovládána pomocí umělé inteligence. Jeho předností je schopnost opravit sám sebe a popřípadě opravit i Canadarm2. Canadarm3 disponuje hlavním 8.5 metrů dlouhým ramenem, vedlejším menším pohyblivějším ramenem a sadou vyměnitelných nástrojů [2].

1.2 Dextre

Název Dextre pochází z anglického Special Purpose Dexterous Manipulator, ve významu obratný manipulátor pro speciální účely. Robot má dvě ramena, která mají sedm stupňů volnosti. Jsou konstrukčně podobná menším ramenům Canadarm2.

Obrázek 1.1: Rameno Canadarm2 [1].

Robot dokáže zastat různé práce, jako jsou inspekce povrchu stanice, její oprava, instalace nových zařízení a výměna starších dílů nebo baterií. Podstatnou výhodou tohoto robota je možnost spolupráce s ramenem Canadarm2, na které se dokáže připojit a využívat jeho velký dosah. Obr. 1.2 ukazuje Dextre robota přichyceného k robotickému ramenu Canadarm2[1].

Obrázek 1.2: Manipulátor Dextre [3].

Kapitola 2

Koncept kráčení robota

V této kapitole bude popsán návrh na chůzi robota.

2.1 Základ chůze

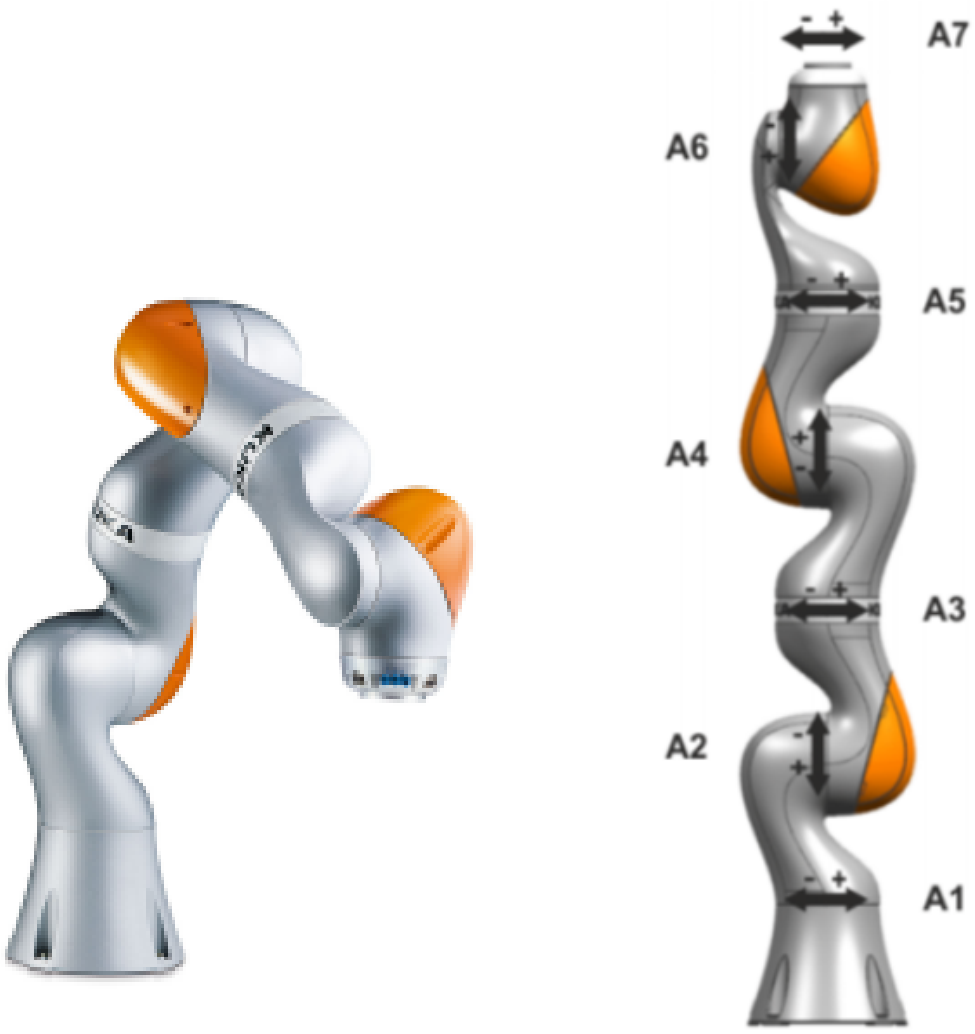
Hlavním faktorem, který napomáhá řešení našeho problému robotů na obíhajících satelitech, je rovnováha gravitačních a odstředivých sil, což má za následek nulové zrychlení satelitů (a na nich umístěných robotů) na oběžných drahách. Při normální chůzi na Zemi nám napomáhá gravitace, kdy noha, na které stojíme, je společně s hmotností našeho tělo přitahována k Zemi. Na vesmírné stanici, by se při "tradiční chůzi"nebo při jiném nezabezpečeném pohybu robot a stanice oddělily. Ve vesmírné praxi tak budeme potřebovat robota ke stanici vhodně přichytit. V této práci budeme proto předpokládat, že robot bude chycený ke stanici speciálním gripperem uzpůsobeným ke spolehlivému přichycení.

Koncept kráčejícího robota je v principu obdobný přemísťování ramene Canadarm2, avšak využívá více stupňů volnosti, které mu zajistí vyšší obratnost (dexteritu). Kráčející robot bude mobilním robotem. Roboty, které jsou pevně připevněné k Zemi, mají stálý zdroj elektrické energie. Mobilní robot si tento luxus dovolit nemůže a zatím jedinou alternativou je energie z baterií. Z hlediska jeho nezávislosti by tyto baterie měl mít robot s sebou a optimálním místem pro jejich umístění je co nejblíže řídícímu systému robota.

Ke kráčení potřebujeme minimálně dvě robotická ramena. Navržený mechanismus vesmírného robota pro chůzi staví na inspiraci pavoukem. V podmínkách zemské gravitace nejjednodušší mechanismus pavouka potřebuje šest nohou pro zajištění stability. Pro vesmírnou aplikaci stačí pouze dvě nohy, které však musí mít grippery.

2.2 Robotická ramena

Nyní musíme vybrat mechanismus, který bude zajišťovat náš kráčející pohyb. Nejjednodušší řešení nabízí samo lidské tělo, kde můžeme vzít lidský trup coby tělo



Obrázek 2.1: *KUKA LBR iiwa 14 R820* [6].

robota a lidské nohy jako základ jeho pohybového mechanismu. S výhodou místo nohou můžeme využít robotických ramen, která mají více než šesti stupni volnosti, (např. roboti od firmy KUKA). Sedmá osa robotického ramena umožňuje vyhnout se případným překážkám v pracovním prostoru robota. Společnost KUKA zde nabízí ideální volbu v podobě robota KUKA LBR iiwa 14 R820.

Robot LBR iiwa je kolaborativní robot, což znamená, že člověk a robot sdílejí pracovní prostor. Zatímco běžné industriální roboty musí být obvykle zavřené v kleci, kam se člověk nedostane, kolaborativní robot nesmí v žádném případě mít možnost zranit člověka. Tuto ochranu dosahuje několika způsoby:

- Senzory momentů (sil) na každé ose robota zajišťují okamžité zastavení robota pomocí brzd, pokud je překročena stanovená mez momentu.
- Omezení rychlosti motorů napomáhají včasnému zastavení robota.
- Robot je "oblečený"do pláště, který při nechtěném dotyku zastaví robota.

Označení LBR pohází z německého slova "Leichtbauroboter", česky robot lehké konstrukce. Robot váží pouhých 29.5 kg. Označení iiwa poté pochází z angličtiny "intelligent industrial work assistant", v překladu inteligentní průmyslový pracovní asistent. Robot s nulovým nástrojem měří něco málo přes 1.3 metru. Pro dosažení maximální pohyblivosti je však jeho pracovní prostor omezen na 820 mm. LBR iiwa se programuje v programovacím jazyce Java, který nabízí mnoho přídavných knihoven, které nám velice zjednoduší programování výsledné trajektorie robota [6].

2.3 Kocept kráčejícího robota

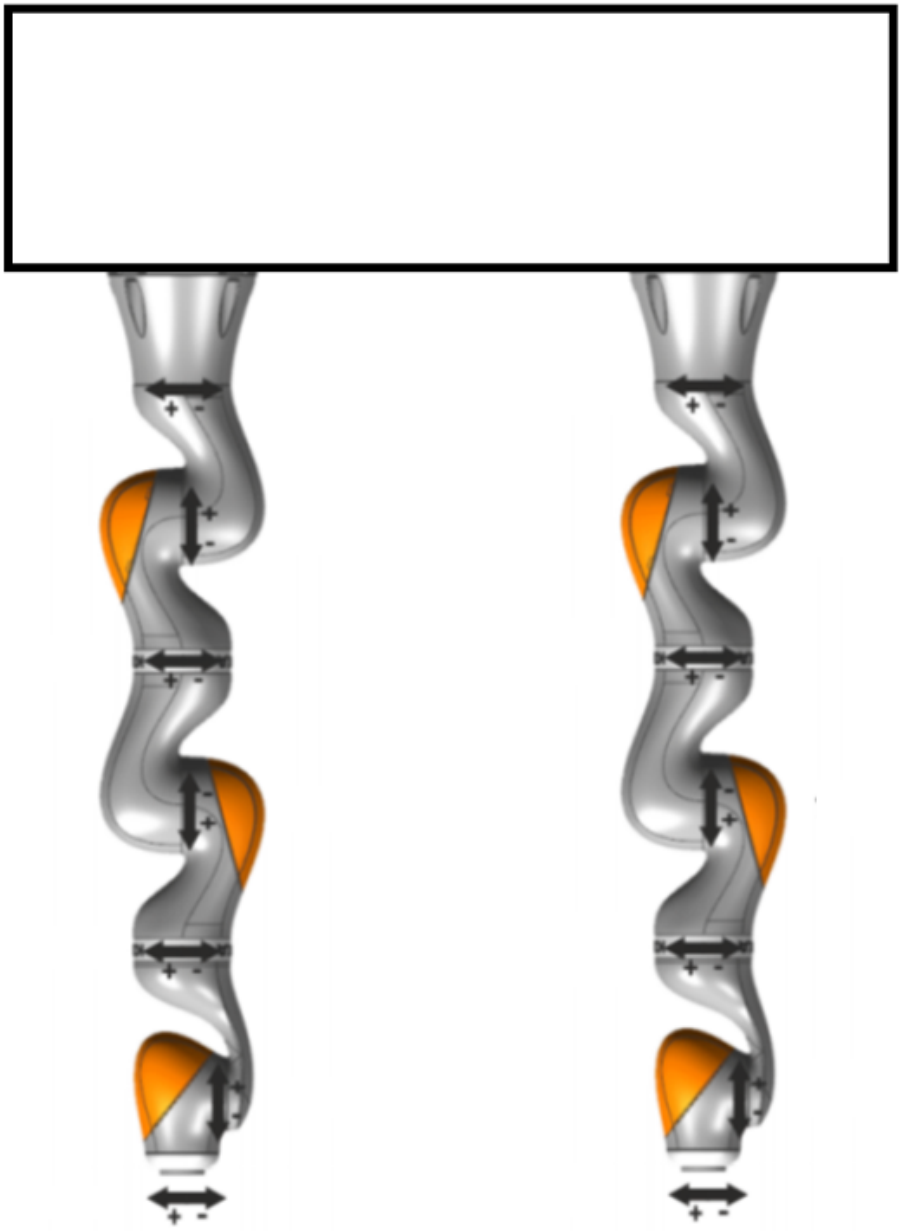
Nyní vezmeme naše vybrané rameno a připojíme ho k hlavnímu tělu robota, kde bude uložen řídící systém a baterie. Tvar těla není pro naše zadání významný, naopak jeho hmotnost by měla být co nejmenší. K tělu připevníme dvě vybraná robotická ramena a vznikne kráčející mechanismus.

Obr. 2.2 zobrazuje náčrt kráčejícího robota. Robot připomíná tvarem spodní část lidského těla, ve kterém bude uložen řídící systém a baterie a ke kterému jsou připojena dvě robotická ramena KUKA. Toto uspořádání robota umožňuje vykonávat kráčející pohyb.

2.4 Princip chůze

Aby se robot a stanice při pohybu (chůzi) robota od sebe neoddělily, pohyb robota se realizuje prostřednictvím speciálních úchopů, ke kterým se ramena robota průběžně chytají. Chůzi lze rozdělit do tří fází, přitom v každé fázi se bude pohybovat jiná část robota. Z výchozí pozice robota se pohne jeho první rameno, které se přichytí k následujícímu úchopu, k němu se poté se přesune tělo robota a následuje přemístění druhého ramena na další úchop.

Navržený pohyb není plynulý, ale takovéto zjednodušení nám pomůže při vytváření modelu robota a jeho simulace.



Obrázek 2.2: Koncept kráčejícího robota

Kapitola 3

Simulační model kráčejícího robota

V této kapitole bude popsán simulační model kráčejícího robota.

3.0.1 Simulační prostředí/Simscape

Simulační model je řešen v programu Matlab/Simulink s využitím rozšíření Simscape [4]. V Simscape je vytvořeno prostředí pro simulaci soustav více těles (Multibody) [5].

3.1 Podrobný model

3.1.1 KUKA.Sim

Abychom zjistili správné rozměry pro kráčejícího robota a následně mohli analyzovat jeho pohybové vlastnosti, využijeme oficiální software KUKA.Sim od společnosti KUKA. Jedná se o simulační program, sloužící k off-line programování robotů [11]. Do programu lze nahrát všechny simulační modely robotů a jejich periferií od této společnosti. Do tohoto prostředí je možné nahrát i našeho robota LBR iiwa. Prostřednictvím programu lze volit jednotlivé části robotických ramen a exportovat je do CAD formátu, který poté zobrazíme pomocí "File Solid"bloku z knihovny Simscape.

3.1.2 Implementace

Abychom mohli napojovat jednotlivé části robota, resp. tělesa na sebe, potřebujeme na nich vytvořit "frame". Frame je souřadný systém, přes který lze napojovat klouby a tělesa. Každé těleso má základní frame, který se nachází v geometrickém středu tělesa. Tento bod není pro nás užitečný, neboť se všechna tělesa v našem robotu napojují přes jejich vnější plochu. Proto je zde možnost frame přemístit na správné místo tělesa.

Tuto možnost lze, bohužel, použít pouze pro nativní tělesa Simscape nebo na určité formáty CAD souborů. Při exportování jednotlivých těles z programu KUKA.Sim máme několik možných formátů 3D souborů. Základním typem byl soubor s příponou .stl, ale po jeho importu do Simscape jej nešlo zobrazit. Proto byl tento formát vložen do CAD programu Autodesk Inventor, v němž bylo zjištěno, že se těleso zobrazuje pouze jako prázdná síť. Takto byly vyzkoušeny veškeré nabízené formáty .step, .stp, .igs, .obj, ale všechny se chovaly při importu totožně. Nejlepší z nich byl formát .step, který se alespoň dokázal zobrazit v Průzkumníku modelu. Na žádný formát však nešlo připojit nové souřadné systémy.

Vzhledem k tomu, že nešlo vytvořit nové souřadné systémy, model musel být manuálně složen z těles, na kterých vytvořit souřadné systémy šly. Tato tělesa byla poté propojena pomocí bloku 3D transformace. Obr. 3.1 ukazuje již hotový model kráčejícího robota. Jsou zde vidět dva roboty LBR iiwa, které jsou připojeny ke společnému tělu. Pro představu byly přidány úchyty, ke kterým se robot bude při pohybu chytat. Důležité je připomenout, že robot má od výrobce 7 stupňů volnosti. Pro praktické účely to znamená, že rameno se dokáže vyhnout překážkám v pracovním prostoru.

Na rozdíl od řízení robota od společnosti KUKA si Matalb/Simulink volí výchozí natočení sedmé osy. Vzhledem k tomu, že se nacházíme v beztíži, rameno se při simulaci začne chovat nepředvídavě. Pokud předpokládáme, že nebudou při pohybu robota na ISS žádné překážky, můžeme osu A7 viz. obr. 2.1 zastavit nebo vypnout.

Takový model robota je však pro simulaci nepoužitelný a slouží pouze k jeho vizualizaci. Jelikož zde není přímá návaznost na jednotlivá tělesa, program by nedokázal spočítat potřebné hodnoty momentů pro náš pohyb.

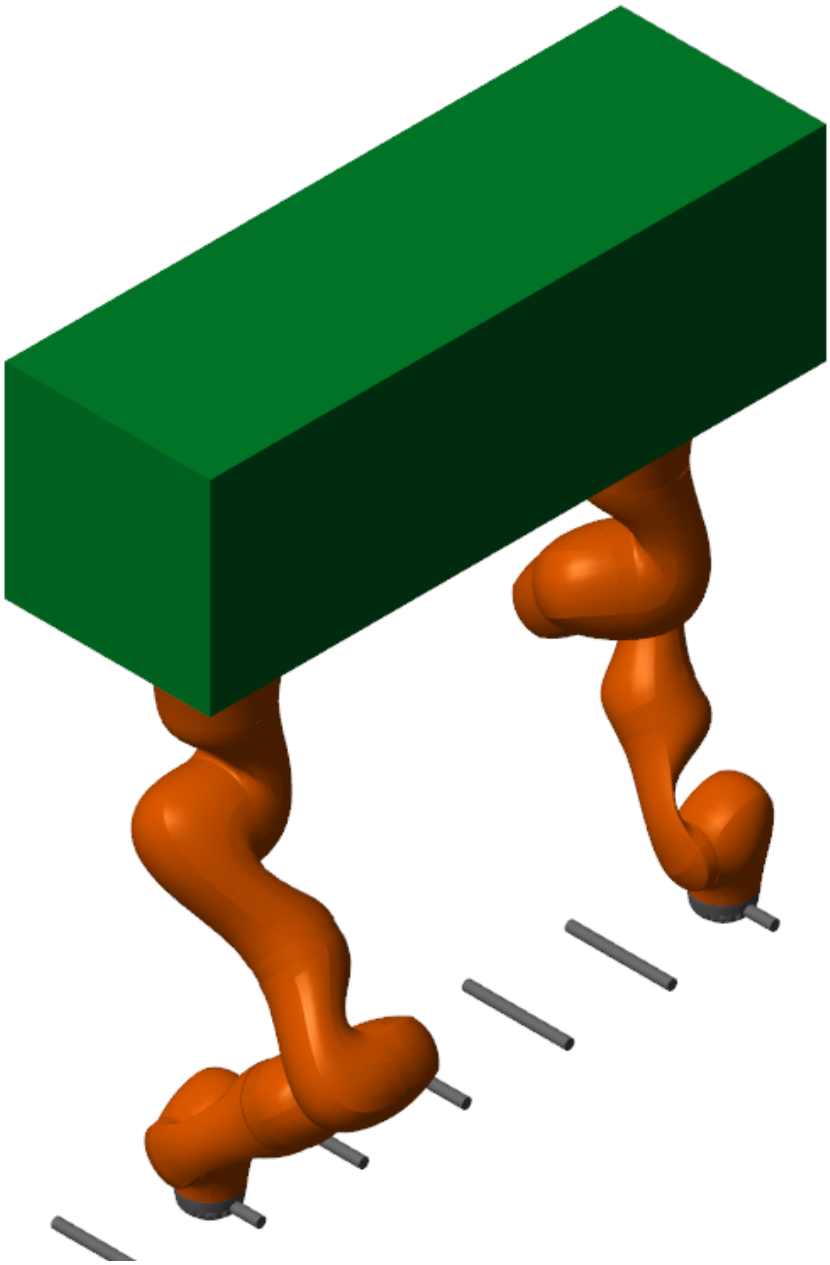
3.2 Zjednodušený model

Vymodelovat tělesa, která by korektně odpovídala reálnému robotu, je zbytečné, a proto se spokojíme s modely, které pouze zobrazují podstatu mechanismu.

3.2.1 Označení těles a kloubů

Jak bylo již uvedeno, pro zjednodušení simulace zastavíme jednu osu robotického ramene. Tímto zredukujeme mechanismus robotického ramene na 6 DOF se sedmi tělesy. Obr. 3.2 zobrazuje popis mechanismu, na který bude odkazováno v následujícím textu. Jelikož jsou nejvýše umístěná tělesa robotických ramen KUKA pevně připojená k společnému tělu, tak je budeme považovat za jedno těleso, označené jako těleso 8. Dále popíšeme krajní, spodní body mechanismu jako bod $G1$ a $G2$.

Poté popíšeme jednotlivé klouby robota jako rotaci mezi dvěma tělesy a získáme jednotlivá natočení φ_{23}, φ_{34}, ..., φ_{1314}. Obdobně úhlové rychlosti ω_{23}, ..., ω_{1314} a úhlové zrychlení ε_{23}, ..., ε_{1314}.

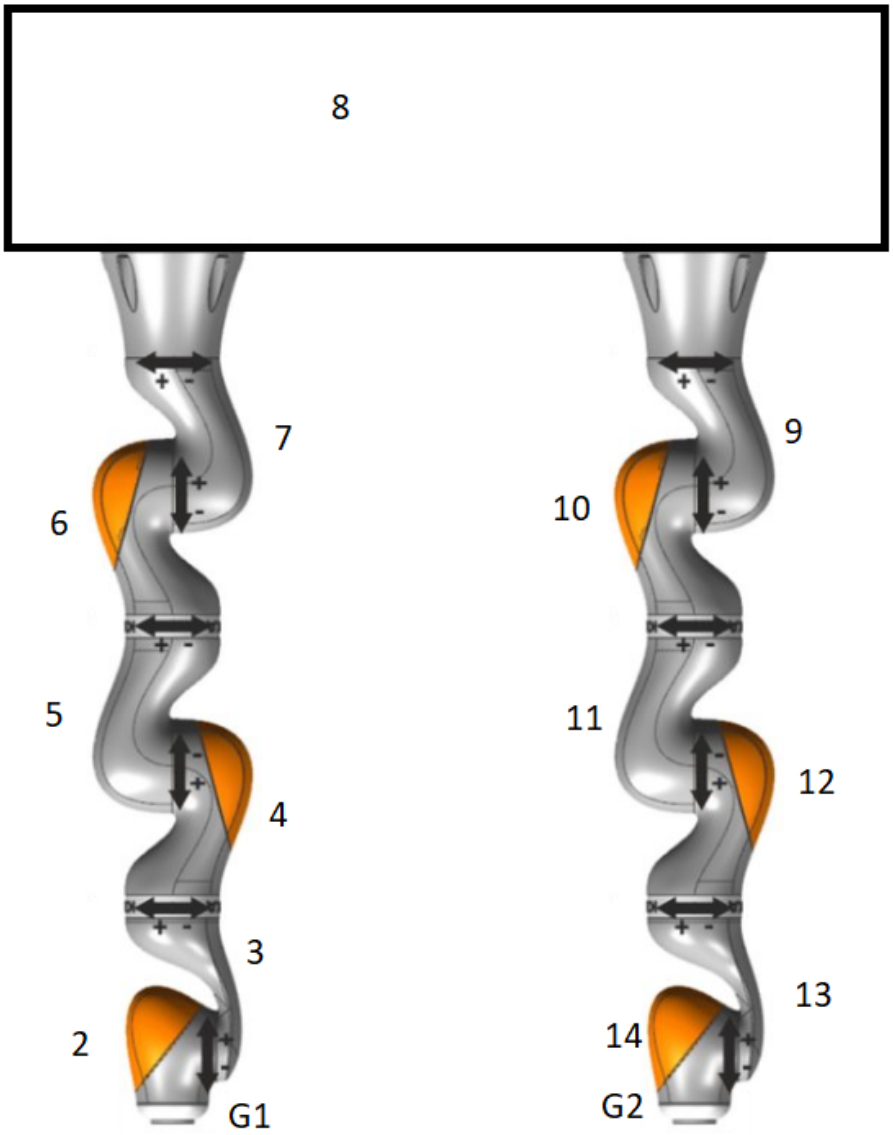


Obrázek 3.1: Model kráčejícího robota

3.2.2 Nové prvky

Pro zjednodušený model použijeme bloky, které jsou přímo nabízeny v rozšíření Simscape. Všechna ramena nahradíme válci a tělo robota necháme jako kvádr. Délky ramen určíme podle obr. 3.3 a jsou uvedeny v tab. 3.1. Průměry válců odvodíme podle rozměrů naměřených z exportovaných součástí robota KUKA na hodnotu 0.075 metrů. Takto jsou ramena dostatečně viditelná v prohlížeči modelu a zároveň v další simulaci dostatečně vystihnou setrvačné síly při pohybu.

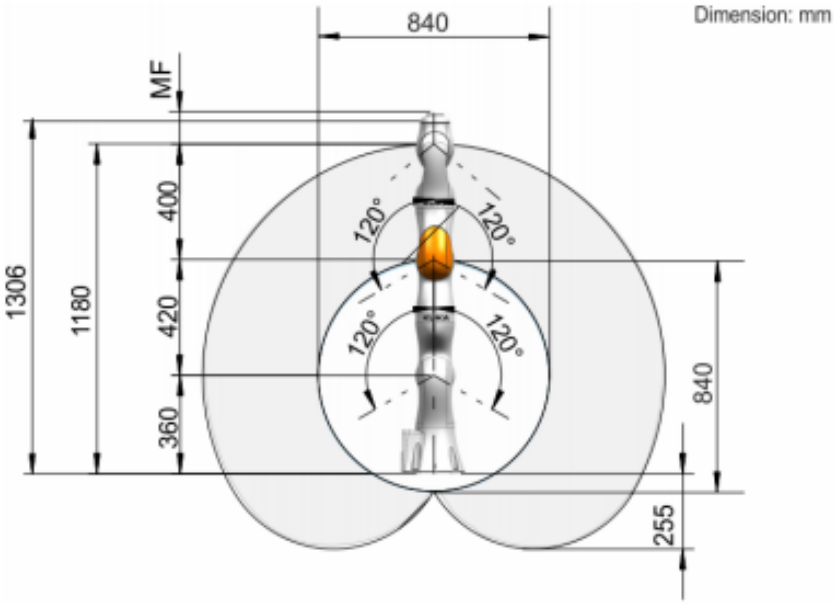
Rozměry a napojení tělesa 8 jsou zobrazeny na obr. 3.4. Hlavní společné těleso má tvar hranolu s rozměry 0.4 x 1 x 0.4 metrů.



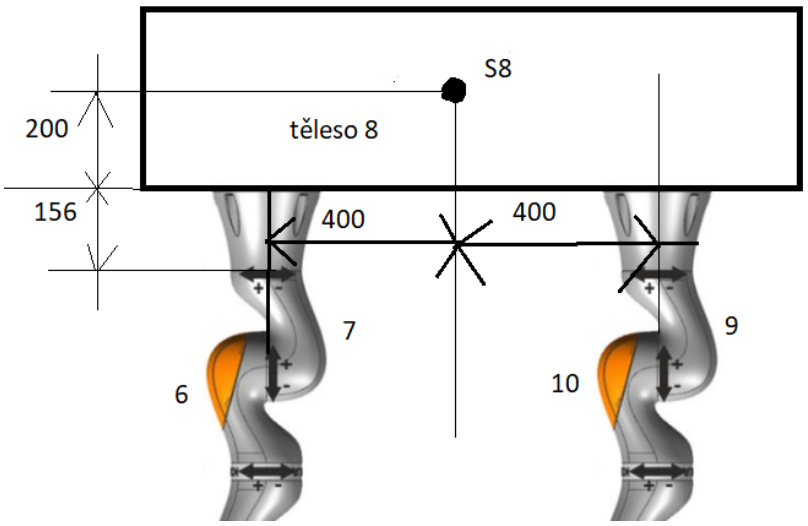
Obrázek 3.2: Označení robota

Rameno	Délka [mm]
2, 14	126
3, 13	215
4, 12	185
5, 11	216
6, 10	204
7, 9	204

Tabulka 3.1: Délky jednotlivých částí robotického ramena KUKA.



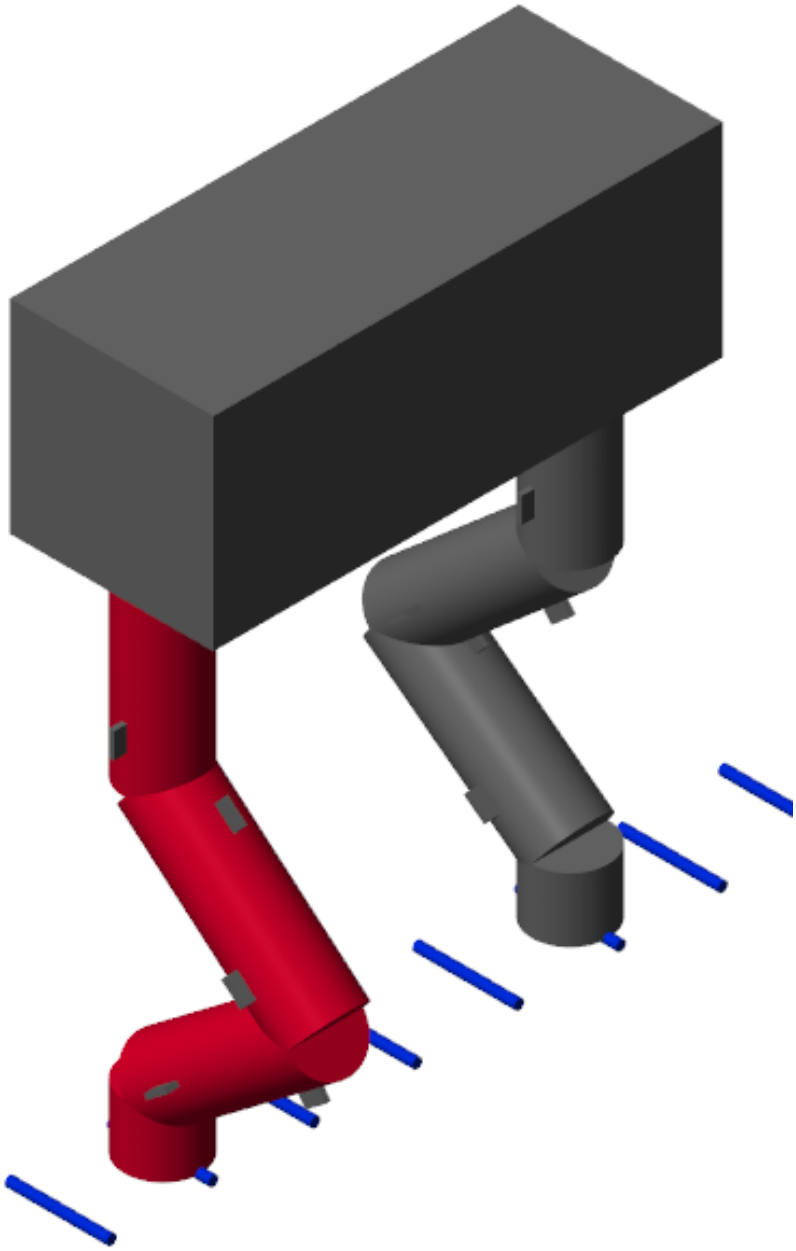
Obrázek 3.3: Rozměry robota KUKA iiwa [10].



Obrázek 3.4: Rozměry tělesa 8

3.2.3 Spojení těles

Jelikož robot má veškeré klouby rotační, využijeme Simscape blok Revolute Joint. Jak je v popisu bloku napsáno, osa rotace musí být pouze v ose Z. Toto nastavení zaručíme souřadnými systémy (frame) na povrchu těles. Ostatní parametry rotační vazby necháme defaultní, tj. nulové tření, tuhost a neomezené natočení.



Obrázek 3.5: Zjednodušený model kráčejícího robota

3.2.4 Konečný model

Obr. 3.5 zobrazuje zjednodušený dokončený model kráčejícího robota. Robot je zobrazen v nativním prostředí Simscape, Průzkumník modelu. Modré válce dole zobrazují úchopy, podle kterých bude robot kráčet. Malé značky na tělesech jsou pouze orientační, abychom poznali, zda se jednotlivá tělesa mechanismu spodních ramen otáčí v souladu s předpoklady simulace. Abychom robotická ramena lépe rozlišili, budeme je dále nazývat šedé a červené.

Kapitola 4

Inverzní kinematická úloha

Tato kapitola popisuje inverzní kinematické řešení robota. Pojmem inverzní kinematika rozumíme převod pracovních souřadnic do kloubových.

4.1 Pohyb robota

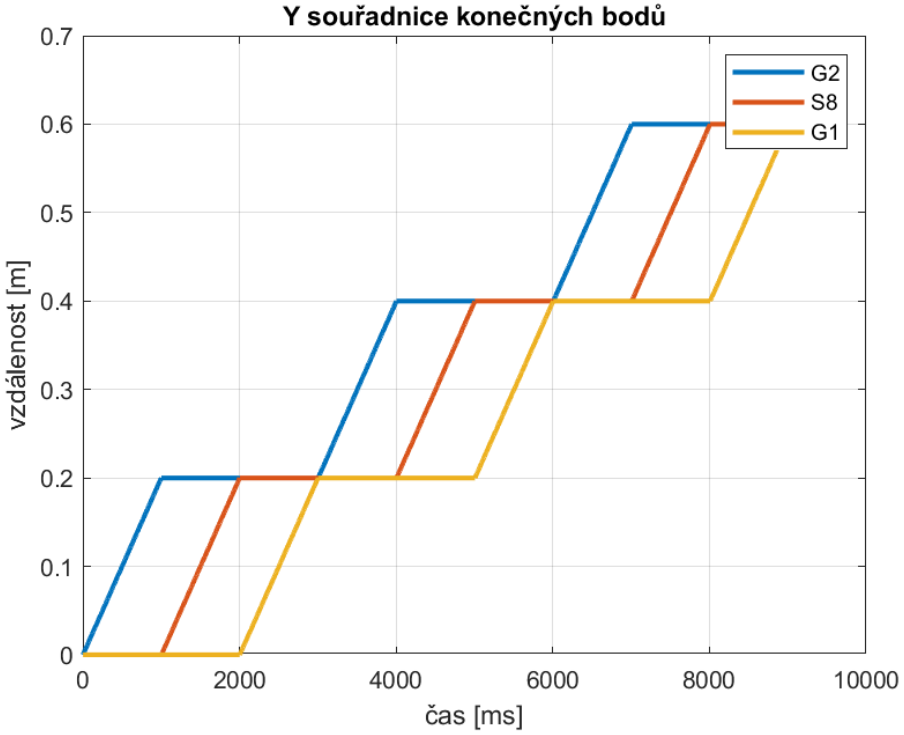
Po vytvoření modelu robota musíme navrhnout jeho pohyb. Jelikož se jedná o mobilního robota, musíme předepsat pohyb všech jeho částí, tj. pohyb červeného robotického ramena, těla a šedého ramena, viz body G1, G2 a střed tělesa 8 na obr. 3.2. Obr. 4.1 pak ukazuje polohy jednotlivých uvedených bodů v čase. Jak bylo výše naznačeno, chůze je rozdělena do 3 částí, kdy každá trvá jednu vteřinu a jednotlivé části robota na sebe čekají. Z obr. 4.1 plyne, že proběhlo 9 fází neboli robot ušel 3 kroky. Ramena jsou od sebe vzdálená 0.8 metru a tato hodnota je poté přičtena k výslednému průběhu polohy. Stejné platí pro střed tělesa 8, které je však posunuto pouze o 0.4 metru.

Další pohyb robota bude v ose Z. V této ose se budou pohybovat body G1 a G2, resp. konce robotických ramen, což reprezentuje zvedání ramen do určité výšky. Zvedání zajistí, že robotická ramena při pohybu do dalšího příchytného bodu do ničeho nenarazí a zároveň bere v potaz budoucí gripper, který se bude muset přichytit a pustit. Obr. 4.2 zobrazuje pohyb robotických ramen v ose Z. Výška kroku je 0.1 metru a trvá stejnou dobu jako přemístění obou ramen.

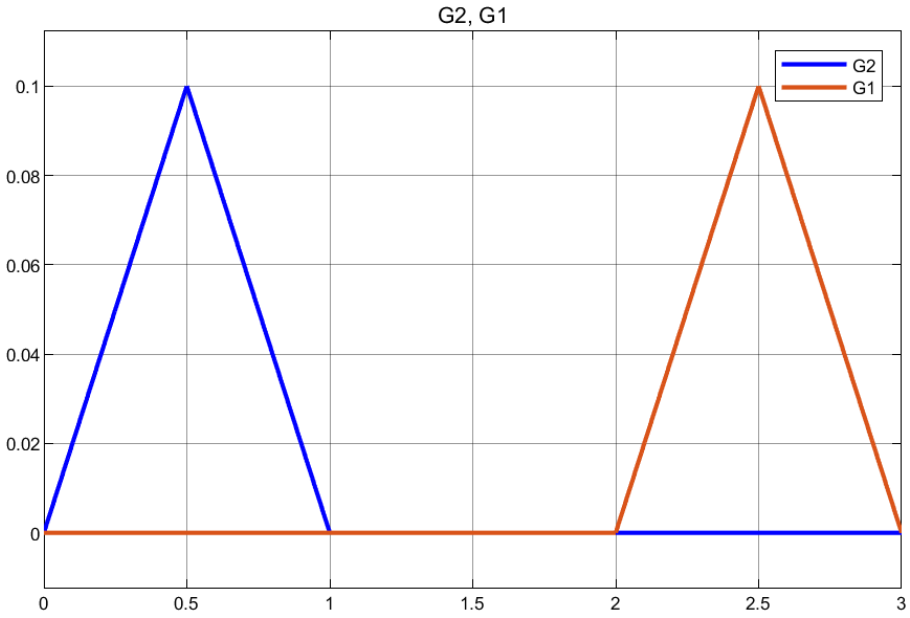
4.1.1 Simulink a Simscape

Jelikož je model robota vytvořený v prostředí Simulink/Simscape, využijeme v něm dostupných bloků. Pro vytvoření pohybu použijeme blok Repeating Sequence. Blok vytváří signál, kdy se konkrétnímu času přiřadí zvolená hodnota a hodnoty v nezadaných časech leží na přímce mezi známými body. Toto je velice jednoduché zadání naší trajektorie koncových bodů G2, S8 a G1.

Abychom vytvořenou trajektorii zadali robotu, potřebujeme v rozšíření Simscape



Obrázek 4.1: Y souřadnice konečných bodů robota



Obrázek 4.2: Z souřadnice bodů G1 a G2

blok s názvem "Bushing Joint". Jedná se o 3D posuvnou a rotační transformaci dvou navazujících těles. Jeden konec proto bude základna mechanismu (těleso 1) a druhým bude naše požadovaná poloha koncových bodů. Trajektorii zadáváme v čase, a tudíž i řešení inverzní kinematické úlohy bude časové.

4.2 Řešení inverzní kinematiky

4.2.1 Analytické řešení

Inverzní kinematické řešení mechanismů se šesti stupni volnosti je složitá úloha. Analytické řešení je možné v případě přítomnosti sférického zápěstí v mechanismu. Sférickým kloubem označujeme místo, kde se tři po sobě jdoucí osy robota protínají. Na tuto podmínku se v praxi velice dbá a většina robotů je vyvíjena s ohledem na sférické zápěstí. Pokud robot má sférické zápěstí, analytická metoda zaručuje přesné a rychlé řešení inverzní kinematické úlohy [13].

Teoretické řešení našeho robota předvedeme na příkladu, kdy bod G1 je statický. Střed tělesa 8 je koncový bod. Obecný pohyb středu tělesa 8 v prostoru popíšeme pomocí souřadnic x_{S8}, y_{S8}, z_{S8} a Cardanovými úhly φ_x, φ_y, φ_z jako:

$$\mathbf{T_{1S8}} = \mathbf{T_x}(x_{S8})\mathbf{T_y}(y_{S8})\mathbf{T_z}(z_{S8})\mathbf{T}_{\varphi_\mathbf{x}}(\varphi_x)\mathbf{T}_{\varphi_\mathbf{y}}(\varphi_y)\mathbf{T}_{\varphi_\mathbf{z}}(\varphi_z) \tag{4.1}$$

Náš robot KUKA lbr iiwa má sférické klouby dva. První je v ose mezi tělesy 6 a 7 a druhý je mezi tělesy 4 a 5. Pro nejjednodušší výpočet zvolíme sférický kloub mezi tělesy 6 a 7, který je bližší koncovému bodu robota.

Nyní popíšeme dopřednou kinematikou místo sférické vazby. Jelikož tělesa 3 a 4 jsou vždy na jedné přímce, můžeme jejich vzdálenosti sečíst. Stejné platí pro tělesa 4 a 5. Jelikož známe pozice bodů G1 a S8, můžeme popsat sférickou vazbu ze dvou směrů, směr z bodu G1:

$$\mathbf{T_{1L}} = \mathbf{T}_{\varphi_\mathbf{y}}(\varphi_{23})\mathbf{T_z}(l_{34})\mathbf{T}_{\varphi_\mathbf{z}}(\varphi_{34})\mathbf{T}_{\varphi_\mathbf{y}}(\varphi_{45})\mathbf{T_z}(l_{56})\mathbf{T}_{\varphi_\mathbf{z}}(\varphi_{56})\mathbf{T}_{\varphi_\mathbf{y}}(\varphi_{67}) \tag{4.2}$$

a směr z bodu S8:

$$\mathbf{T_{1R}} = \mathbf{T_z}(-r_{2z})\mathbf{T_{1S8}}\mathbf{T_y}(-S8y)\mathbf{T}_{\varphi_\mathbf{z}}(-\varphi_{78})\mathbf{T_z}(-S8z)\mathbf{T_z}(-l_7) \tag{4.3}$$

Jelikož se jedná o stejný bod, musí platit, že

$$\mathbf{T_{1L}} = \mathbf{T_{1R}} \tag{4.4}$$

Rozepsáním těchto matic poté dostaneme 12 rovnic o 6 neznámých φ_{23} až φ_{78}. Stejný postup bychom opakovali pro druhé, šedé rameno.

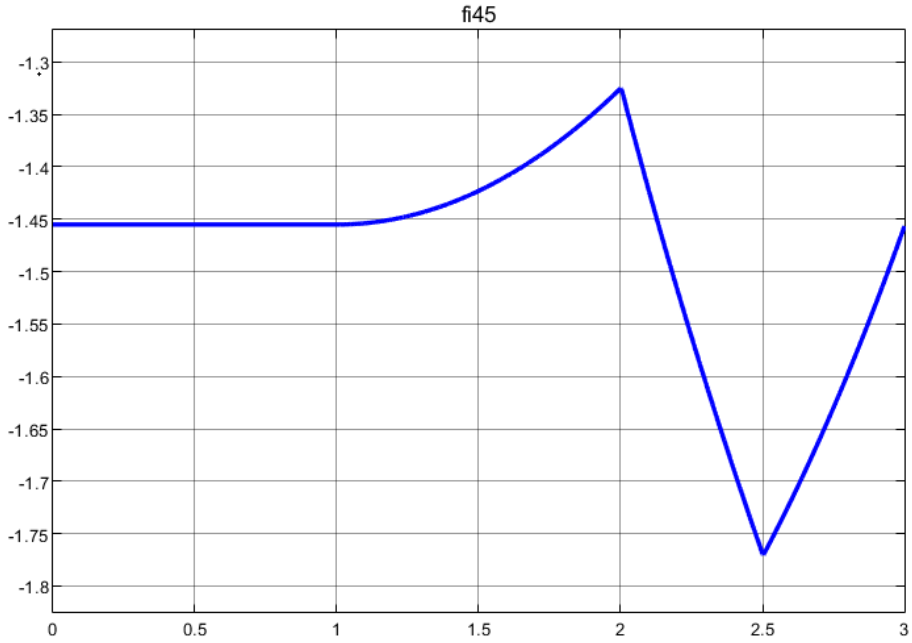
4.2.2 Numerické řešení

Kinematické rovnice popisující mechanismus ve tvaru:

$$f(z, q) = 0 \tag{4.5}$$

kde z jsou závislé proměnné a q jsou nezávislé. Rovnice (4.5) jsou nelineární a jejich analytické řešení je možné pro omezený počet případů, více [13]. Pokud tyto podmínky nejsou splněny, musíme použít numerické řešení. Pro tyto účely je nejlepší modifikovaná Newtonova iterační metoda. Metoda je založena na Taylorově rozvoji prvního řádu. Z diference se poté spočítá Jakobián a pokud není matice singulární, metoda vede k řešení do požadované přesnosti [7].

4.3 Řešení pomocí Simscape



Obrázek 4.3: Natočení φ_{45} [rad]

Pro urychlení celkového výpočtu a zjednodušení řešení bylo místo řešení uvedeného v kapitolách 4.2.1 a 4.2.2 použito řešení v prostředí Simscape-Multibody, ve kterém je již model robota vytvořen. Každý kloub v Simscape dokáže měřit své mechanické vlastnosti. Toto zde využijeme a zavedeme odměřování natočení. Jelikož se jedná o mechanismus se šesti stupni volnosti, jsou možné jakékoli hodnoty natočení a poloh koncových bodů v pracovním prostoru robota. Jedinou podmínkou, kterou musíme dodržet, je maximální vzdálenost koncových bodů, která je určena výškou (délkou) robota. V modelu nejsou nastaveny kolize mechanismu, a proto nám umožní udělat jakýkoli pohyb, který splňuje vazebné podmínky řešení.

Jelikož máme 12 kloubů, získáme z měření 12 natočení. Ukázka natočení je na obr. 4.3. Jedná se o úhel φ_{45}, což je osa červeného ramene. Do 1 sekundy simulace se nic neděje, protože se v první části pohybuje pouze šedé rameno.

Kapitola 5

Inverzní dynamická úloha

Kapitola popisuje řešení inverzní dynamické úlohy, která vypočítá momenty potřebné pro zadaný pohyb

5.1 Problém inverzní dynamiky

Inverzní dynamická úloha vypočítá potřebné síly, resp. momenty v každém kloubu robota. Musíme však znát jejich zrychlení, ze kterých pak vypočteme potřebné hodnoty momentů. Tyto momenty jsou pro nás důležité pro návrh konstrukce robota či návrhu materiálu, ze kterého bude robot vyroben. Také podle těchto hodnot momentů lze zhodnotit, zda mechanismus je v praxi použitelný.

5.1.1 Newton-Euler

Tato vektorová metoda využívá fyzikální souřadnice mechanismu. Spočívá v sestavení Newton-Eulerových rovnic mechanismu. Newtonovy pohybové rovnice jsou definovány jako:

$$m\mathbf{a_s} = \mathbf{F} \tag{5.1}$$

respektive

$$m\begin{bmatrix} a_{sx} & a_{sy} & a_{sz} \end{bmatrix} = m\begin{bmatrix} F_x & F_y & F_z \end{bmatrix} \tag{5.2}$$

a Eulerovy pohybové rovnice jsou definovány

$$\mathbf{I_s}\boldsymbol{\varepsilon} + \boldsymbol{\omega} \times \mathbf{I_s}\boldsymbol{\omega} = \mathbf{M_s} \tag{5.3}$$

kde index S referuje k těžišti tělesa a moment jeho setrvačnosti je daný bez potřeby Steinerovy věty. Tyto rovnice lze spojit v soustavu rovnic

$$\mathbf{Ma} = \mathbf{DR} + \mathbf{Q} \tag{5.4}$$

kde M je matice hmotností a momentů setrvačnosti, matice a obsahuje zrychlení a úhlová zrychlení. Matice D reprezentuje rozvržení sil a momentů, matice R jsou

vnitřní síly a momenty působící na těleso a v matici Q jsou uvedeny vnější síly a momenty. Rovnice (5.4) řeší jak dopředný, tak inverzní dynamický problém podle toho, které hodnoty známe a které chceme dopočítat.

5.1.2 Lagrangeovy rovnice smíšeného typu

Lagrangeovy rovnice jsou dalším typem řešení dynamické úlohy. Jsou založeny na kinetické energii mechanismu a virtuální práci vnějších sil. Mechanismus s n stupni volnosti je popsán pomocí m závislých souřadnic.

$$s_j, \quad j = 1, 2, ..m, \quad m \geq n \tag{5.5}$$

Tyto souřadnice jsou vázány vazbovými podmínkami

$$f_k(s_j, t) = 0, \quad k = 1, 2, .., r, \quad r = m - n \tag{5.6}$$

Následně definujeme kinetickou energii mechanismu jako

$$E_k = E_k(s_j, \dot{s_j}, t) \tag{5.7}$$

a pro jedno těleso platí Königova věta

$$E_k = \frac{1}{2} m v_s^2 + \frac{1}{2} \omega^T I_s \omega \tag{5.8}$$

Lagrangeovy rovnice smíšeného typu definujeme

$$\frac{d}{dt} \frac{\partial E_k}{\partial \dot{s_j}} - \frac{\partial E_k}{\partial s_j} = Q_j + \sum_{k=1}^{r} \lambda_k \frac{\partial f_k}{\partial s_j}, \quad j = 1, 2, ..m \tag{5.9}$$

kde kde Q_j jsou zobecněné síly a λ_k Lagrangeovy multiplikátory odpovídající vazbovým rovnicím f_k. Následuje několik způsobů výpočtuQ_j. Jedním z nich je princip virtuálních prací [7].

$$Q_j \partial s_j = \mathbf{F_i} \cdot \partial \mathbf{r_i} + \mathbf{M_i} \cdot \partial \boldsymbol{\varphi_i} \tag{5.10}$$

5.2 Simscape

Nicméně pro urychlení výpočtů bylo místo řešení v 5.1 opět využito prostředí Simscape-Multibody, kde natočení je již počítáno z inverzní kinematiky. Jak bylo v kapitole 4.3 naznačeno, každá rotační osa dokáže odměřovat veškeré veličiny spojené s rotačním pohybem. Zde nastavíme odměřování momentu.

Nyní musíme získat zrychlení jednotlivých os robota. Výhoda Simscape je, že zrychlení se dopočítává samotným programem. Proto stačí vzít neupravené natočení z inverzní kinematické úlohy a každé ose určit její natočení. Toto nastavení změníme v kolonce "aktuace vazby", při kterém zadáváme pohyb a necháme program dopočítat hodnoty momentů. Jak bylo již uvedeno, tyto momenty změříme stejným způsobem, jako jsme změřili natočení kloubů.

Pokud bychom pouze vzali úhly natočení a přímo je poskytli modelu robota, celá simulace by selhala. Abychom z modelu získali potřebná data, musíme jej pevně připojit k základně (těleso 1). Tato základna však není pevně daná, neboť se jedná o mobilního robota. Dále nám problém dělají dvě robotická ramena. V druhé fázi, kdy by měla být obě ramena přichycena a měla by hýbat s hlavním tělem, vznikají v mechanismu dvě smyčky, v nichž každé rameno reprezentuje jednu.

Takové nastavení modelu nám program zamítne. I přesto, že polohy jsou z inverzní kinematiky exaktní a konstrukce robota je totožná, ramena si vzájemně konkurují. Program tuto podmínku rovnou zkontroluje a zamezí simulaci hláškou, že jsme omezeni na 6 DOF.

5.3 3 modely

Řešení se nachází v rozdělení kroku robota do 3 fází, které jsou zachyceny příslušným modelem.

První model

První model bude zvedat pouze první rameno, které se bude volně pohybovat. Druhé rameno bude pevně připojené k základně. Takto vznikne mechanismus se 12 stupni volnosti, ale výsledný model bude obsahovat pouze jednu smyčku. Jelikož máme všech 12 natočení již určených, můžeme je volně aplikovat na model. Natočení na obr. 4.3 je po celou první sekundu konstantní a rameno se hýbat nebude. To však pro momenty v rameni neplatí, jelikož nesou hmotnost robota a současně se pohybující první paže. Vypočítané momenty z tohoto modelu platí pouze pro první třetinu simulace, zbylý čas bude řešen dalšími modely.

Třetí model

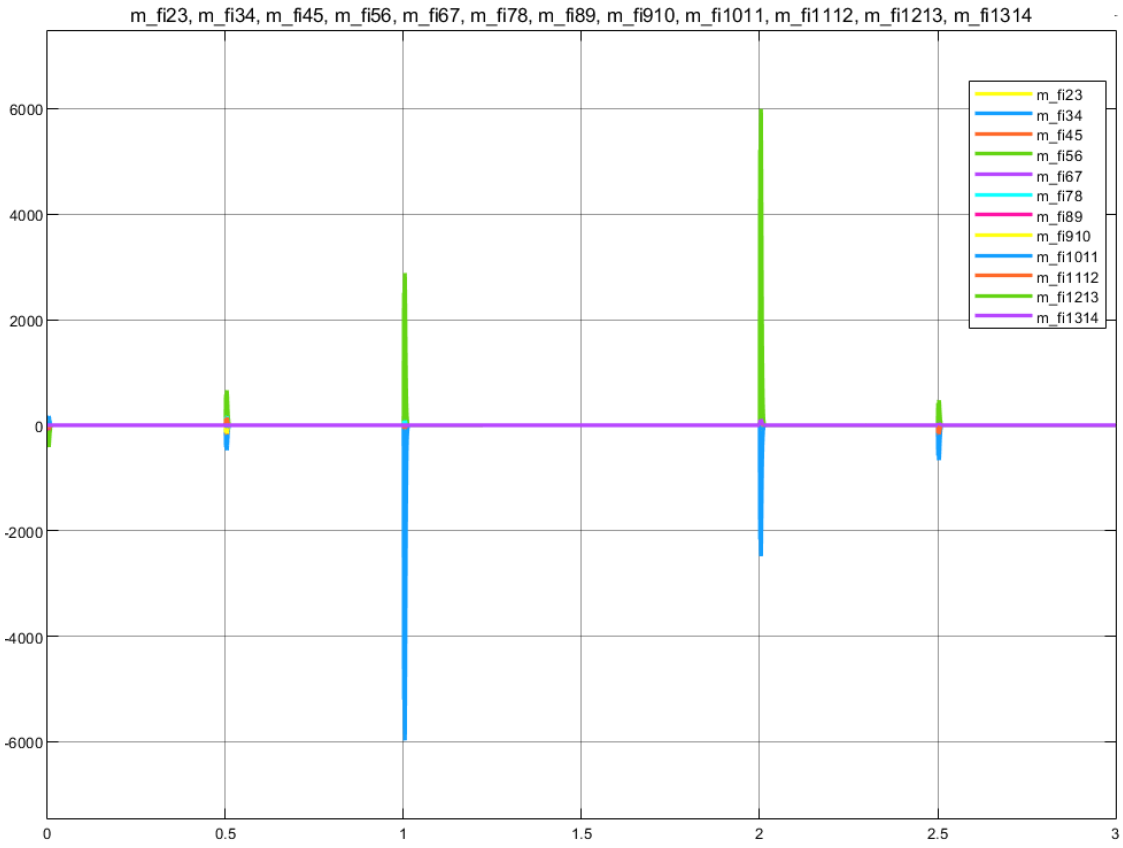
Třetí model bude platit v poslední třetině simulace. Jedná se o poslední fázi pohybu. Šedé rameno nebude vykonávat žádný pohyb a červené rameno se přitáhne na další úchyt. Třetí model je tak pouze obrácený první model první fáze.

Druhý model

Tento model se bude týkat situace, kdy jsou obě ramena přichycena a hýbe se společné tělo robota. Jak bylo již uvedeno, nucený pohyb obou ramen není možný. Nejjednodušší způsob je respektovat chybovou hlášku a vytvořit robota se šesti stupni volnosti. Proto jednomu ramenu nastavíme momenty nulové a trajektorii necháme dopočítat. A pokud dodržíme vazebné podmínky mechanismu, žádná kolize nevznikne.

Tento postup si můžeme dovolit za podmínek rovnováhy sil působících na celé těleso robota v podmínkách ISS .Vypočítané momenty pouze vystihují nucený pohyb

robota. Zde na Zemi by byly momenty v pohonech příliš veliké a museli bychom se uchýlit k alternativním řešením.



Obrázek 5.1: Vypočítané momenty [Nm] pro všech 12 kloubů

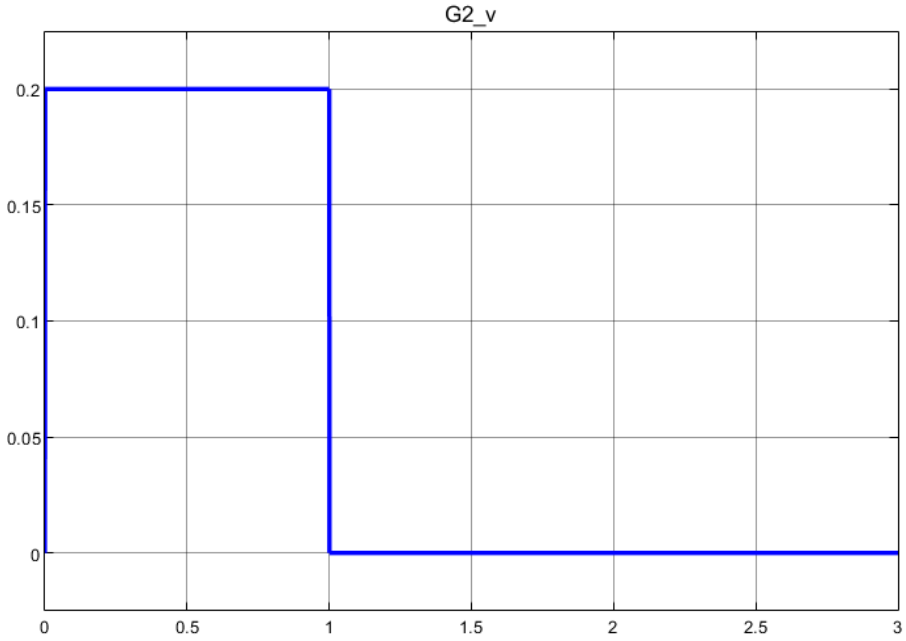
Nyní vypočítané momenty v každém separátním modelu spojíme v jeden výsledný průběh momentu. . Správně definujeme, v jakém časovém okamžiku každý model platí. Výsledné momenty jsou na obr. 5.1

5.4 Nově definovaná trajektorie

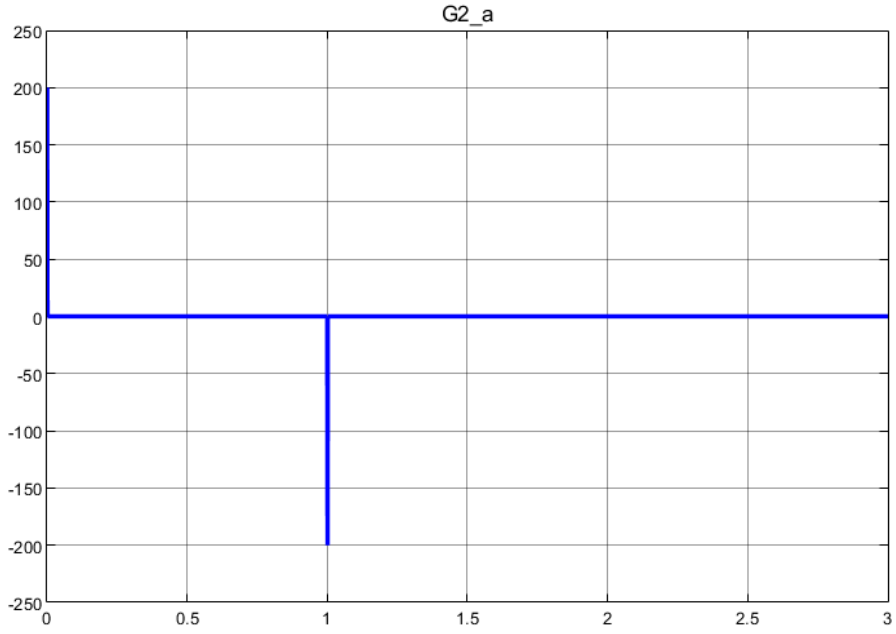
Z obr. 5.1 je patrné, že takový moment je nepřípustný. 6000 Nm je téměř nereálná hodnota a pohon, který dokáže tento moment vyvinout, je vysoce náročný na energii, které je v bateriích robota omezené množství. V kapitole 4.1 popisujeme dráhy koncových bodů robota jako přímky. Obr. 5.2 a 5.3 ukazují rychlost a zrychlení bodu G2. Zrychlení zde dosahuje velice vysokých hodnot, což odpovídá momentům v osách robota.

5.4.1 Návrh nové trajektorie

Pokud pořebujeme plynulé průběhy hodnot momentů robota, musíme najít funkci, která zajistí hladký průběh polohy, rychlosti a zrychlení. Polynom 5. řádu nám



Obrázek 5.2: Rychlost bodu G2



Obrázek 5.3: Zrychlení bodu G2

nejlépe vystihne tyto vlastnosti. Jeho 2. derivace (zrychlení) je poté polynom 3. stupně, který zaručuje pozvolný náběh motorů.

Výpočet trajektorie

Obecný polynom 5. stupně je popsán rovnicí:

$$y = a_0 t^5 + a_1 t^4 + a_2 t^3 + a_3 t^2 + a_4 t + a_5 \tag{5.11}$$

jeho první a druhá derivace:

$$\dot{y} = 5a_0t^4 + 4a_1t^3 + 3a_2t^2 + 2a_3t + a_4 \tag{5.12}$$

$$\ddot{y} = 20a_0t^3 + 12a_1t^2 + 6a_2t + 2a_3 \tag{5.13}$$

V těchto 3 rovnicích vystupuje 6 neznámých parametrů a_0 až a_5 Pro určení těchto parametrů zavedeme okrajové podmínky $y(0) = 0$, $\dot{y}(0) = 0$, $\ddot{y}(0) = 0$, $y(T) = y_F$, $\dot{y}(T) = 0$, $\ddot{y}(T) = 0$. Kde y_F je konečná pozice v ose Y.

Po dosazení okrajových podmínek do (5.11), (5.12) a (5.13) dostaneme soustavu lineárních rovnic

$$\mathbf{B} = \mathbf{Ta} \tag{5.14}$$

kde $\mathbf{B}$ je levá strana polynomu, $\mathbf{T}$ je matice polynomu a $\mathbf{a}$ je matice konstant polynomu. Koeficienty polynomu získáme převedením matice T na druhou stranu rovnice

$$\mathbf{a} = \mathbf{T}^{-1}\mathbf{B} \tag{5.15}$$

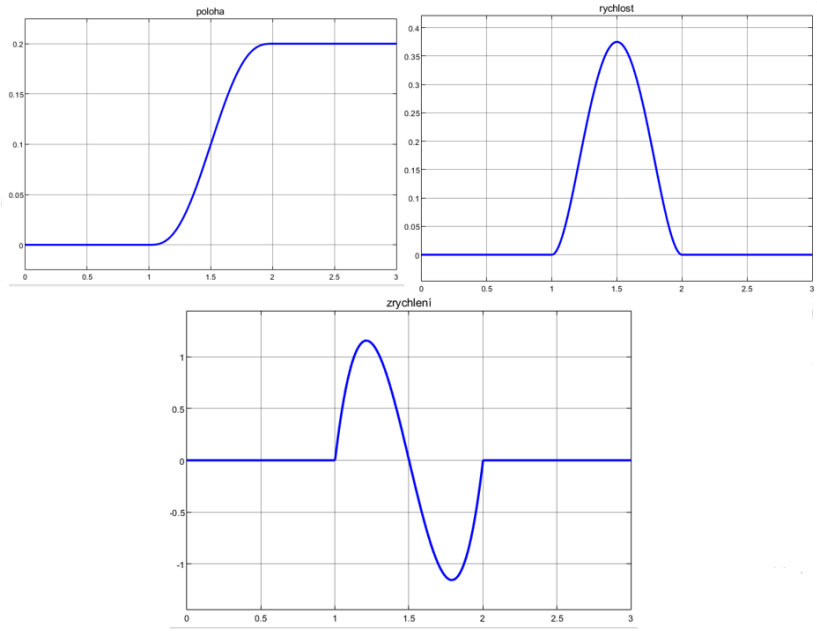
Podle našich požadavků lze poté jednoduše upravovat výpočet na jinou výslednou polohu i čas, kdy koncový bod dosáhne této polohy.

MATLAB Function

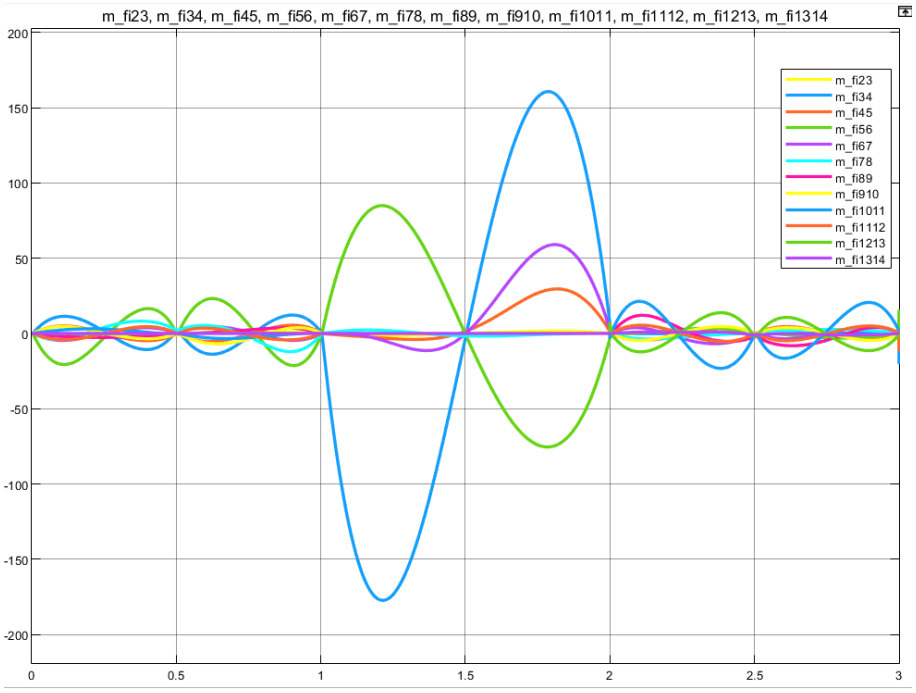
Pro takovýto výpočet však jednoduchý blok neexistuje a musíme aplikovat prostředí Matlab, které nám tuto soustavu vyřeší. Prostředí lze načíst blokem MATLAB Function. Toto prostředí má vlastní pracovní prostor a vlastní úložiště proměnných. Blok vypočítává výstup na základě vstupu v každém časovém kroku, a proto s variabilním časovým krokem se blok chová nepředvídatelně. Řešením je zafixování časového kroku. 1 kHz je dostatečně rychlá frekvence na zjištění vlastností modelu a zároveň výpočtový čas je přijatelný.

Výsledný průběh

Obr. 5.4 vlevo nahoře zobrazuje polohu Y, vpravo rychlost a dole zrychlení. Jak je vidět, celá trajektorie je mnohem plynulejší, což se i projeví na výsledném momentu v jednotlivých osách robota. Podobný polynom 5. stupně bude platit v ose Z, viz. obr. 4.2. Výsledné momenty po úpravě trajektorie jsou na obr. 5.5. Největší dosažená hodnota momentu je pouhých 175 Nm. Tento moment nastává v druhém modelu, který vystihuje posun společného těla robota. Společné tělo je nejhmotnější součást robota, z čehož lze odvodit, že výsledné modely jsou správné.



Obrázek 5.4: Poloha, rychlost a zrychlení polynomu 5. stupně



Obrázek 5.5: Konečné momenty robota

Kapitola 6

Řízení robota

Kapitola se zabývá simulací řízení os robota pomocí kaskádní regulace.

6.1 Kaskádní regulace

6.1.1 PID regulátor

PID regulátor je zpětnovazební regulátor, který funguje na principu minimalizace regulační odchylky

$$e(t) = w(t) - y(t) \tag{6.1}$$

kde w je požadovaná veličina a y je její aktulání hodnota. Poté obecný zásah regulátoru definujeme jako

$$u(t) = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(t)\, dt + T_d \frac{de(t)}{dt} \right) \tag{6.2}$$

K_p reprezentuje konstantu proporcionální složky regulátoru a vystupuje jako hlavní činitel v regulaci. T_i je integrační konstanta a dorovnává trvalou regulační odchylku. T_d je derivační konstanta a pomáhá systému překonat veliké rozdíly v regulační odchylce. Derivační složka je však často nestabilní a spíše systém rozkmitá, než vyrovná. Jednotlivé složky regulátoru se mohou kombinovat a v praxi se nejčastěji setkáme s P, I, PI, PD a PID regulátory.

6.1.2 Seřízení regulátoru

Existuje několik metod nastavení PID regulátoru. První metoda se nazývá Ziegler-Nicholsova metoda, jejíž použití se dělí podle systému, který ovládáme.

Ziegler-Nicholsova metoda kritického zesílení a frekvence

První způsob spočívá ve vypnutí integrační a derivační složky ($T_i \to \infty \quad T_d = 0$) a nastavení K_p na nejmenší možnou hodnotu. Poté K_p stále zvětšujeme až na mez

stability, kdy systém kmitá s konstantní amplitudou. Následně odečteme kritickou periodu kmitajícího systému T_k a nastavení P regulátoru, které nazveme K_0. Konstanty PID regulátoru určíme podle tabulky 6.1.

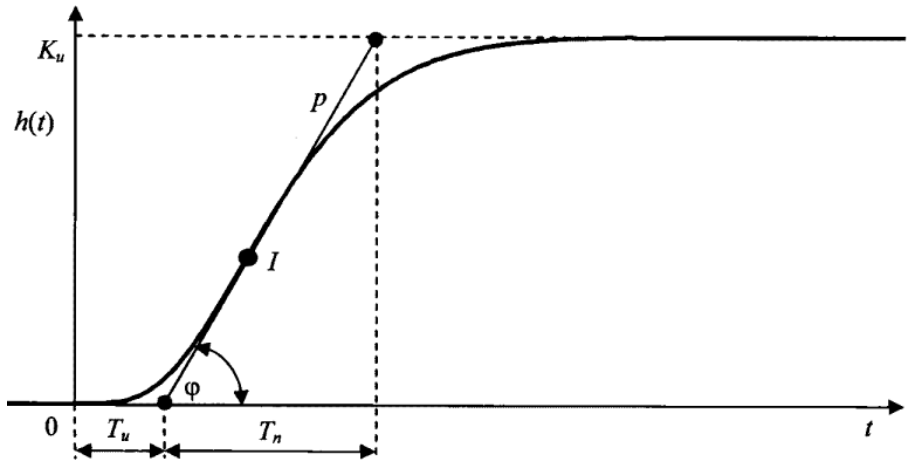
Regulátor	K_p	T_i	T_d
P	$0.5 \cdot K_0$	-	-
PI	$0.45 \cdot K_0$	$0.85 \cdot T_k$	-
PID	$0.6 \cdot K_0$	$0.5 \cdot K_k$	$0.125 \cdot K_0$
I	-	$2 \cdot K_k$	-

Tabulka 6.1: Nastavení konstant regulátoru Z-N metodou

Při seřizování regulátoru touto metodou není nutné znát model soustavy a kritické zesílení eventuálně najdeme vždy. V praxi si však nemůžeme dovolit překročit mez stability, protože by to vedlo ke zničení regulovaného systému. Ať už to je spálený motor, elektrické vedení nebo rozbitá součást, je to nepřípustné.

Ziegler-Nicholsova metoda přechodové charakteristiky

Jak bylo vysvětleno v minulé kapitole, rozkmitání systému na mez stability je často nemožné a pro tyto případy využijeme druhý způsob Ziegler-Nicholsovy metody. Ten je založen na přechodové charakteristice systému ve tvaru S. Charakteristika je zobrazena na obr. 6.1. Z této charakteristiky poté odečteme hodnoty K_u, T_u a T_n. Podle tabulky 6.2 vypočítáme konstanty regulátoru.



Obrázek 6.1: "S"charakteristika Z-N metody

Wadeho metoda seřízení PID regulátoru

Tato metoda se také nazývá pokus-omyl. Je to jednoduchá metoda, která seřizuje regulátor podle odezvy na skok, buď na požadované veličině nebo na poruše. Nejdříve

Regulátor	K_p	T_i	T_d
P	$\frac{T_n}{K_u \cdot T_u}$	-	-
PI	$0.9 \cdot \frac{T_n}{K_u \cdot T_u}$	$3.33 \cdot T_u$	-
PID	$1.2 \cdot \frac{T_n}{K_u \cdot T_u}$	$2 \cdot T_u$	$0.5 \cdot T_u$

Tabulka 6.2: Nastavení konstant regulátoru Z-N metodou

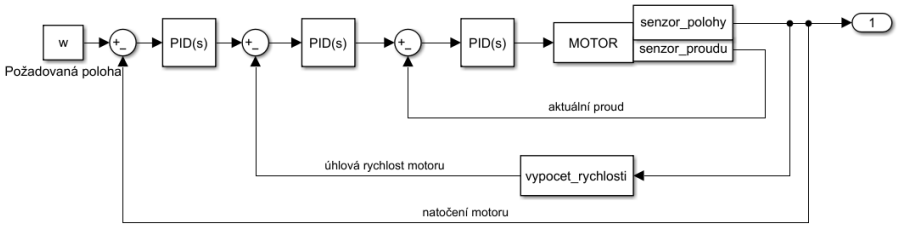
opět vypneme integrační a derivační složku a K_p nastavíme na nízkou hodnotu. Poté pomalu hodnotu K_p zvyšujeme, dokud systém nemá potřebnou odezvu na skokovou změnu žádané veličiny, zároveň trvalá regulační odchylka je zanedbána.

V dalším kroku se K_p sníží na 75% původní hodnoty a snižuje se T_i, které odstraní trvalou regulační odchylku. Pak se znovu doladí K_p a vznikne PI regulátor a seřizování je dokončené.

Pro D složku se zpočátku nastaví $T_d = 0.1 \cdot T_i$. Pokud se však průběh nezlepší, derivační složka se opět vypne. Jestliže se zlepší, T_d se může zvednout až na $T_i/4$. Pokud lze derivaci použít, v posledním kroku opět zvýšíme K_p a snížíme T_i [9].

6.1.3 Kaskádní regulace

Kaskádní regulace řadí více PID regulátorů za sebou. Každá smyčka PID má vlastní regulační odchylku a vlastní nastavení konstant regulátoru. Vzniká tak robustnější regulace, která je méně náchylná na chyby a rušivé signály. Obr. 6.2 ukazuje schéma kaskádní regulace elektrických motorů. Regulace obsahuje 3 smyčky. První nejnižší



Obrázek 6.2: Kaskádní regulace elektrických motorů

úroveň je regulace proudu. Tato smyčka pracuje na nejvyšší frekvenci. Aby smyčka byla realizovatelná, musíme mít zpětnou vazbu proudu, kterou jde realizovat senzorem proudu nebo stavovým pozorovatelem. Každá metoda má své výhody a nevýhody, ale senzor proudu je nejjednodušší aplikace.

Druhá smyčka je rychlostní. Aby smyčka opět fungovala, musíme znát aktuální úhlovou rychlost motoru. Rychlost nelze přímo měřit a ve většině aplikací se výpočet realizuje derivací polohy nebo integrací zrychlení.

Poslední smyčka je polohová, do té vstupuje naše žádaná poloha a je odečtena od aktuálního natočení motoru.

V kaskádní regulaci se jednotlivé regulátory seřizují stejnými metodami jako v kap. 6.1.2. Nejdříve nastavíme vnitřní smyčku a poté postupně ladíme vnější smyčky.

Dopředná složka regulace

Celý regulátor lze výrazně vylepšit přidáním dopředné složky. Tato část je přičtena k žádané veličině a zredukuje požadavek na regulátor, který pouze vyrovnává odklonění.

Jelikož jsme již dříve spočítali hodnoty momentů nutné k pohybu robota, můžeme tyto hodnoty využít v řízení robota jako dopřednou složku v regulaci. V mechatronických systémech mají proud a síla podobný význam, proto lze naše momenty přímo vložit do kaskádního řízení.

Dále lze regulaci vylepšit dopředným řízením rychlosti, kdy derivujeme požadovanou polohu a tuto hodnotu přičteme do rychlostní smyčky.

6.2 Implementace Simulink/Simscape

Jelikož výpočet momentů již proběhl v rozdělených modelech, řízení robota také rozdělíme na 3 fáze, které odpovídají pohybu robota. Robot má dohromady 12 motorů, které musíme ovládat, a proto potřebujeme 12 různých kaskádních regulací.

Na obr. 6.2 je zobrazena proudová smyčka. Jelikož Simscape nabízí ovládání kloubů pomocí síly/momentů, tak v takovém případě simulovat elektrický motor nemusíme a rovnou ovládáme mechanismus přes momenty. Z tohoto důvodu vnitřní proudovou smyčku odstraníme a zbude nám kaskádní regulace o dvou smyčkách.

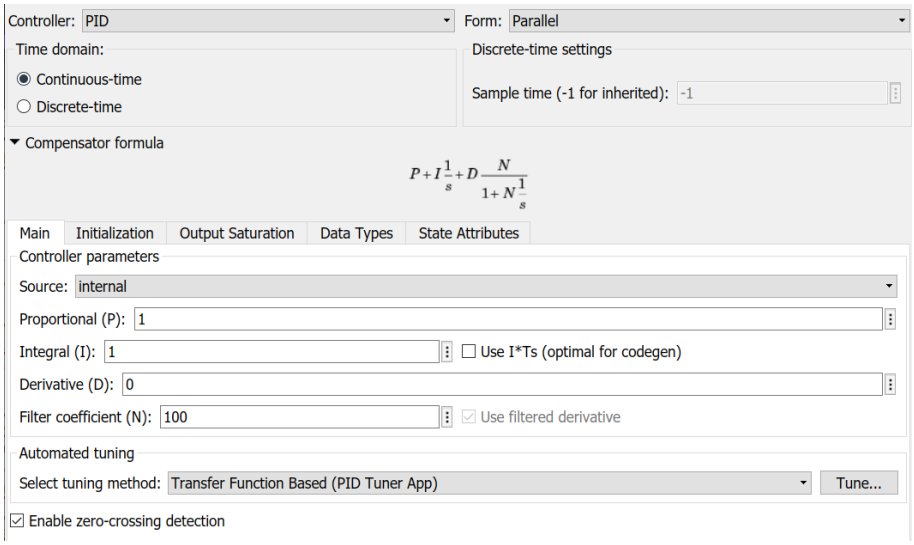
6.2.1 Kaskádní regulace v Simulinku

PID regulátor v Simulinku

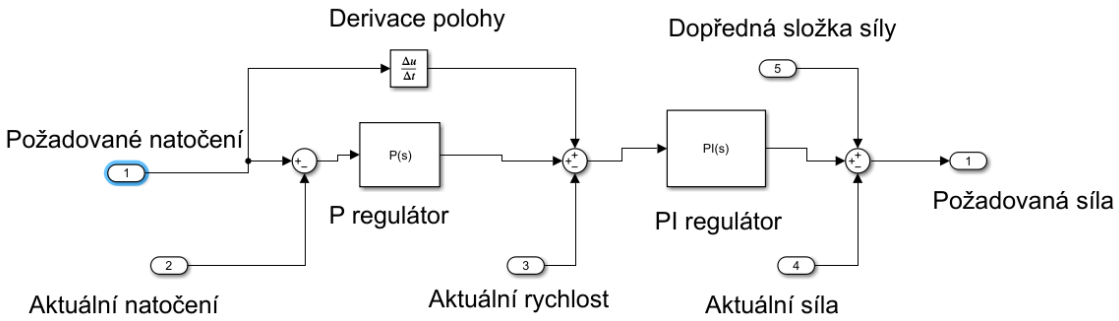
PID regulátor je již v Simulinku naprogramovaný, a proto bude stačit pouze sestavit kaskádní regulaci. Obr. 6.3 ukazuje defaultní nastavení PID regulátoru v Simulinku. Je zde možnost volby typu regulátoru (P, PI, PD, PID a I), parametry PID regulace s filtrací, saturace signálu, automatické seřízení regulátoru a mnoho dalšího.

Sestavení kaskádní regulace v Simulinku

Obr. 6.4 ukazuje kaskádní regulaci jednoho kloubu robota. Jak jsme dříve popsali, máme zde pouze regulátory polohy a rychlosti. Požadované natočení je již vypočítáno z inverzní kinematiky a dopředná složka síly je vypočítána z inverzní dynamiky.



Obrázek 6.3: PID regulátor v Simulinku



Obrázek 6.4: Kaskádní regulace robota v Simulinku

Referenční hodnoty natočení, rychlosti a momentu vezmeme z ovládaného modelu. Zde nastává problém kauzality. Nemůžeme spočítat rozdíl mezi požadovanou a aktuální polohou, jelikož aktuální poloha je nepřímo počítána z rozdílu požadované a aktuální polohy. Stejné tvrzení platí pro rychlost i moment. Pro tento problém existují dva bloky, Memory a Delay.

Oba bloky posunují signál o určitý časový krok dozadu. V Delay specifikujeme o kolik časových kroků se signál zpozdí a Memory má vždy pouze jeden. V obou případech jde nastavit počáteční hodnota. Pokusy bylo zjištěno, že Memory blok se používá v modelech s variabilním časovým krokem a Delay je určen pro fixovaný čas. Jelikož všechny naše modely jsou nastavené na fixním 1 kHz, použijeme Delay blok. Počáteční hodnoty jsou v případě rychlostí a momentů nulové, jelikož pohyb ještě nenastal. Natočení všech kloubů nastavíme podle inverzní kinematiky.

Seřízení regulátorů

V kaskádní regulaci byla zohledněna vazba mezi PID regulátory. Pro seřízení regulátorů využijeme metodu pokus-omyl, která je popsána v kapitole 6.1.2. Nejprve nastavíme vnitřní PI regulátor a poté vnější P regulátor. Derivační složka je zde

nežádoucí, jelikož model rozkmitá do hodnot, které program nedokáže zpracovat. Takto postupně nastavíme všech 12 regulátorů.

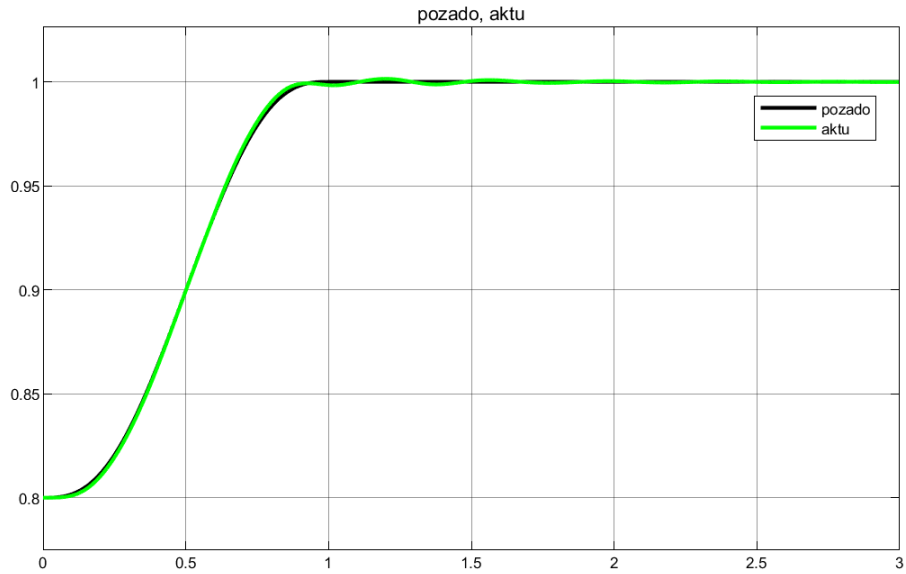
Z kapitoly 5.3 máme 3 modely, kde dva pohybují každý s jedním ramenem a třetí pohybuje s celým společným tělem robota. Ideálně bychom chtěli konstanty regulátoru pro všechny modely stejné, ať není potřeba více nastavení. Z obr. 5.5 vyplývá, že v modelu, který hýbe celým tělem, jsou momenty přibližně 7x větší než ve zbylých dvou modelech. Jelikož je pohyb mnohem náročnější, nastavení regulátoru musí být přiměřeně velké. Pokud ale regulátory s velkým zásahem aplikujeme na modely, kde zásahy nejsou potřeba, celý systém se stane nestabilním.

Regulátory však fungují i opačně. Když vezmeme regulaci, která dobře ovládá modely s malým zatížením a aplikujeme ji na 2. model, kde pohyb je mnohem náročnější, celý náběh a minimalizace odchylky trvá příliš dlouhou dobu.

Bohužel musíme mít tedy více sad regulátorů, pro každý model jednu sadu.

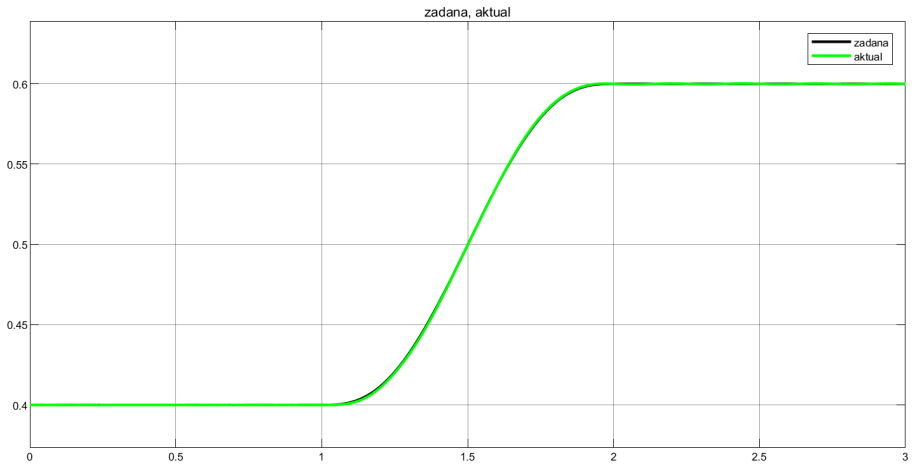
6.2.2 Vyhodnocení

Jelikož je pohyb rozdělen do 3 fází, musíme uvést všechny 3 modely. Obr. 6.5, 6.6 a 6.7 ukazují výslednou polohu bodů G1, S8 a G2. Je to kombinace všech 12 kaskádních regulací každého pohonu. První model simuluje zvednutí a posun šedého ramene. Obr. 6.5 ukazuje Y souřadnici šedého ramene. Krok má délku 0.2 metrů. Obr. 6.6 ukazuje pohyb ve směru Y společného těla robotů a obr. 6.7 ukazuje červené rameno. Čas je v posledním modelu posunut o 2 sekundy. Černé křivky jsou polynomy 5.

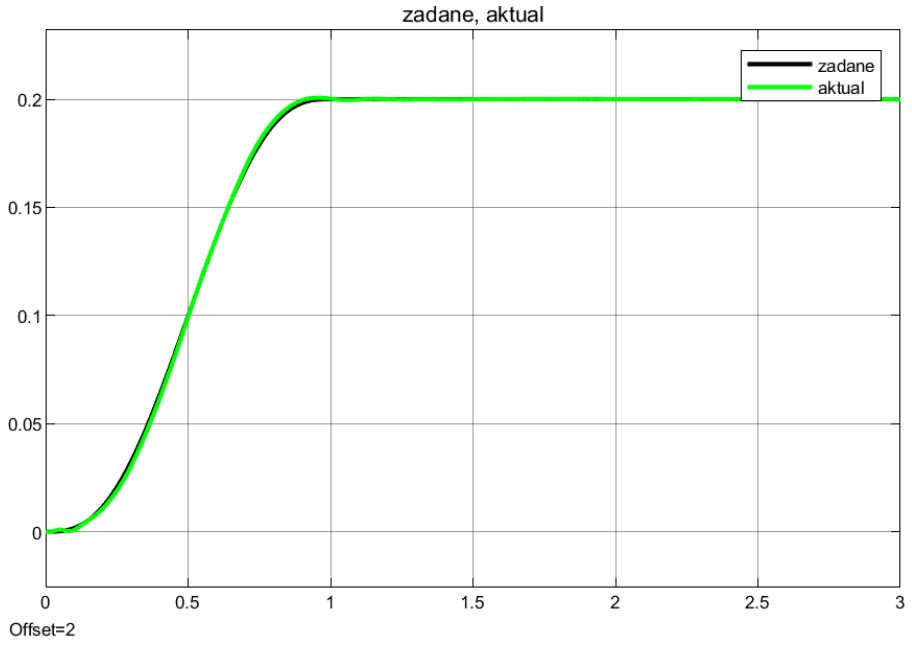


Obrázek 6.5: Výsledná poloha Y šedého ramene

stupně vysvětlené v kap. 5.4.



Obrázek 6.6: Výsledná poloha Y společného tělesa



Obrázek 6.7: Výsledná poloha Y červeného ramene

Kapitola 7

Optimalizace momentů

Kapitola popisuje optimalizační algoritmy a postupy pro minimalizaci potřebných sil pro pohyb mechanismu.

7.1 Optimalizace

Optimalizace znamená hledání extrému funkce, v případě optimalizace se hovoří o cílové funkci.

$$CF_i = CF_i(\mathbf{p}), \quad i = 1, 2, ..., n \tag{7.1}$$

kde n je počet cílových funkcí. a p jsou optimalizační parametry

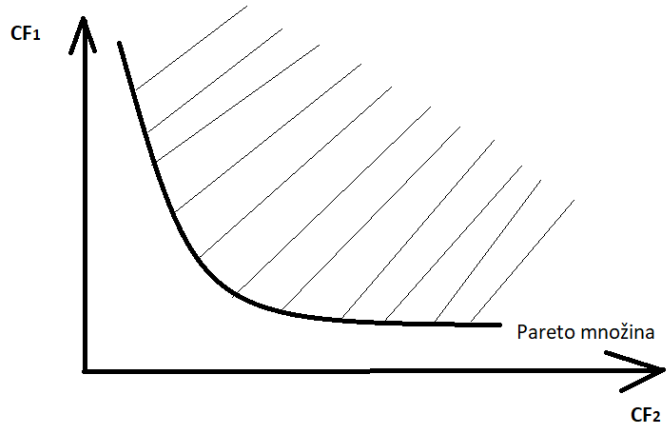
$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & ... & p_m \end{bmatrix}' \quad m = 1, 2, .., m \tag{7.2}$$

kde m je počet optimalizačních parametrů. Pokud forma cílové funkce je analytická, lze spočítat její gradient. Podle gradientu hledáme extrém funkce. Hovoří se o gradientové metodě. V opačném případě je gradient cílové funkce počítán numerickou metodou. Optimalizační metody, které využívají pouze funkční hodnotu cílové funkce, nazýváme negradientní.

Pokud v rovnici (7.1), $i \neq 1$, nastává vícekriteriální optimalizace a musíme definovat Pareto množinu. Obr. 7.1 zobrazuje Pareto množinu dvou cílových funkcí. Jedná se o kompromis mezi jednotlivými funkcemi, kde musíme obětovat jednu pro úspěch druhé. Je potřeba si uvědomit, na které záleží více. Kompromis lze definovat pomocí významových vah:

$$CF = q_1 CF_1 + q_2 CF_2 \tag{7.3}$$

Základní rozdělení optimalizačních metod je na lokální a globální metody. Lokální metody hledají nejbližší lokální extrém a výpočet končí, jakmile je jeden nalezen. Proto velice záleží na výběru počátečních parametrů, které definují, kde začne optimalizační algoritmus. Lokální metody požadují spojitost cílové funkce.



Obrázek 7.1: Pareto množina 2 cílových funkcí

7.1.1 Globální optimalizace

Globální optimalizační metody nacházejí extrém funkce na celém definičním oboru. Zde si musíme dávat pozor, jelikož extrémy jsou často v $\pm\infty$. Výhoda globálních algoritmů je v požadavcích na cílovou funkci. Funkce nemusí být spojitá, není potřeba znát její průběh a ani zda vůbec existuje její gradient. Nevýhoda spočívá v času výpočtu. Metody prohledávají celý definiční obor a hledání správné kombinace parametrů může trvat velice dlouho. Samozřejmě s vyšším počtem optimalizačních parametrů výpočtový čas stoupá a u globálních metod je to markantní. Globální optimalizace zaručují rychlý nález menší oblasti, kde se řešení nachází, ale najít extrém v té oblasti trvá dlouhou dobu. Tato metoda tak nezaručuje nalezení globálního extrému, a pokud se uchytí lokálního extrému, který nemá jiný extrém v blízké vzdálenosti, je zde šance, že tento bod určí jako globální extrém.

Genetický algoritmus

Jedna z globálních metod je genetický algoritmus. Genetické algoritmy jsou velice versatilní metody hledání globálních extrémů, které fungují na principu přirozeného výběru.

Základem je populace, která obsahuje jedince. Každý jedinec je hodnocen fitness funkcí neboli jeho zdatností. V prvním kroku algoritmu jsou jedinci náhodně vygenerováni. Je vypočítána hodnota zdatnosti a jedince s nejlepším hodnocením zvolíme jako rodiče pro další generaci. Noví jedinci jsou generováni několika způsoby. Křížení, mutace a reprodukce [12].

Křížení je výměna části genetické informace. Máme 2 hlavní druhy křížení: aritmetické a prosté. Rodiče nazveme $X^{'}$ a $Y^{'}$ a jejich potomky $\bar{X}$ a $\bar{Y}$. V aritmetickém dělení vznikají noví jedinci rozdělením genetické informace podle rovnic:

$$\bar{X} = rX^{'} + (1-r)Y^{'} \tag{7.4}$$

$$\bar{Y} = (1-r)X^{'} + rY^{'} \tag{7.5}$$

kde r je náhodné číslo $r \in \langle 0, 1 \rangle$. V prostém dělení se prohodí vlastnosti rodičů mezi sebou. p je náhodné celé číslo, které udává pozici prohozené informace. Rodiče definujeme jako:

$$X' = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ x_p' \\ \vdots \\ x_N' \end{bmatrix} \quad Y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_p' \\ \vdots \\ y_N' \end{bmatrix} \tag{7.6}$$

Po prostém křížení jsou potomci ve tvaru:

$$\bar{X} = \begin{bmatrix} x_1' \\ x_2' \\ \vdots \\ y_p' \\ \vdots \\ y_N' \end{bmatrix} \quad \bar{Y} = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ x_p' \\ \vdots \\ x_N' \end{bmatrix} \tag{7.7}$$

Mutace náhodně změní jednu genetickou informaci v jednom potomku. Mutace napomáhá udržovat variaci v populaci a zabraňuje příliš rychlé konvergenci algoritmu. Do zmutované souřadnice se vloží náhodná hodnota z hledaného intervalu. Tento zásah je však veliký, a proto pravděpodobnost mutace je velice omezena (0.001 až 0.1) [8].

Posledním způsobem vytváření nové generace je reprodukce, která pouze zanechá jedince beze změny. Napomáháme tímto rychlosti výpočtu, ale musíme si dát pozor, aby řešení nesklouzlo do lokálního minima.

Po vygenerování nové generace opět spočítáme její zdatnost a dokud není splněna podmínka ukončení, tak určíme další generaci.

7.2 Implementace v Matlab/Simulink

V programu Matlab je optimalizační "addon". Toto rozšíření poskytuje již naprogramované optimalizační algoritmy, které lze využívat pouze s definicí podmínek. Jelikož chceme najít absolutně nejmenší momenty v kloubech, budeme hledat globální extrém, a tudíž využijeme globální metody optimalizace.

Pro optimalizaci využijeme genetický algoritmus. Funkce je již v Matlabu předem naprogramovaná a můžeme si upravovat veškeré parametry uvedené v kap. 7.1.

7.2.1 Struktura programu

Program začíná definováním oblasti, ve které hledáme optimalizační parametry. Poté zavedeme cílovou funkci. Cílovou funkci budeme generovat ze Simscape modelu inverzní dynamiky.

Aby nevznikala nerealizovatelná řešení, zavedeme funkci "try, catch". Tento příkaz provede úkon, který je v sekci "try". Sem vložíme příkaz "sim", který odsimuluje zadaný Simulinkový model. Z něj získáme potřebné veličiny, ze kterých vznikne cílová funkce. Pokud nastane jakákoli chyba, program automaticky přeskočí na příkaz "catch". Chyby většinou nastávají, pokud se mechanismus nedokáže sestavit nebo pokud nějaká hodnota přesáhne paměť programu (odletí do ∞).

V posledním kroku nastavíme parametry genetického algoritmu. Nemusíme upravovat všechny, většina parametrů je v základním nastavení dostatečná. Změníme však podmínky ukončení, aby program neběžel příliš dlouho. První parametr bude "MaxStallGenerations", který zastaví algoritmus, pokud průměrná relativní změna v cílové funkci bude po určitém počtu generací menší než tolerance. Základní počet generací, přes který se tento průměr počítá je 50, my jej však zredukujeme na 30. Dlaší parametr, který zvolíme je tolerance cílové funkce. Základní nastavení je 10^{-6}. Hodnota je příliš nízká a zvedneme ji na 10^{-4}.

7.2.2 Optimalizační parametry

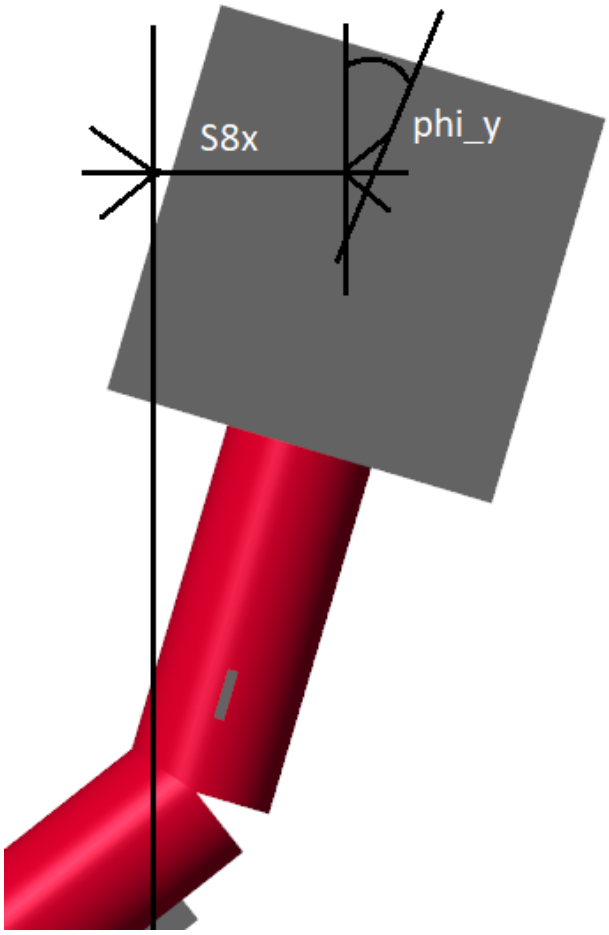
Optimalizační parametry jsou konstanty systému, ze kterých se počítá cílová funkce. Parametry si volíme podle potřeby nebo podle našich možností. Parametrem, který by mohl být zajímavý, je poloha společného tělesa 8. Víme, že pohyb v Y je zobrazen na obr. 5.4. Pohyby v ose X a Z jsou však konstantní a těleso je udržováno v předem dané výšce a poloze. Pokud změníme polohu X měli bychom také změnit natočení tělesa kolem osy Y: φ_y.

Jelikož se jedná o genetický algoritmus, globální metodu, výpočtový čas bude velice dlouhý, proto si nemůžeme dovolit optimalizovat všechny parametry robota. Abychom zachovali přibližnou konstrukci robotického ramene z předlohy, změníme polohu ramen vůči společnému tělesu. Obr. 7.2 ukazuje parametry $S8_x$, poloha středu společného tělesa v ose X a Cardanův úhel φ_y. Obr. 7.3 popisuje zbylé tři parametry, výšku společného tělesa a polohu robotických ramen.

Následně zvolíme oblast parametrů, ve které algoritmus hledá řešení. Parametry jsou limitovány reálným světem. Tab. 7.1 ukazuje oblast hledaných parametrů.

7.2.3 Cílová funkce

Hodnocení zdatnosti jedinců reprezentují hodnoty momentů v jednotlivých kloubech. Pro výpočet hodnot momentů využijeme model inverzní dynamiky. Z tohoto modelu získáme hodnoty momentů do Matlab Workspace. Nahrání proběhne pomocí bloku "To Workspace", který slouží k uložení proměnných do Matlabu.

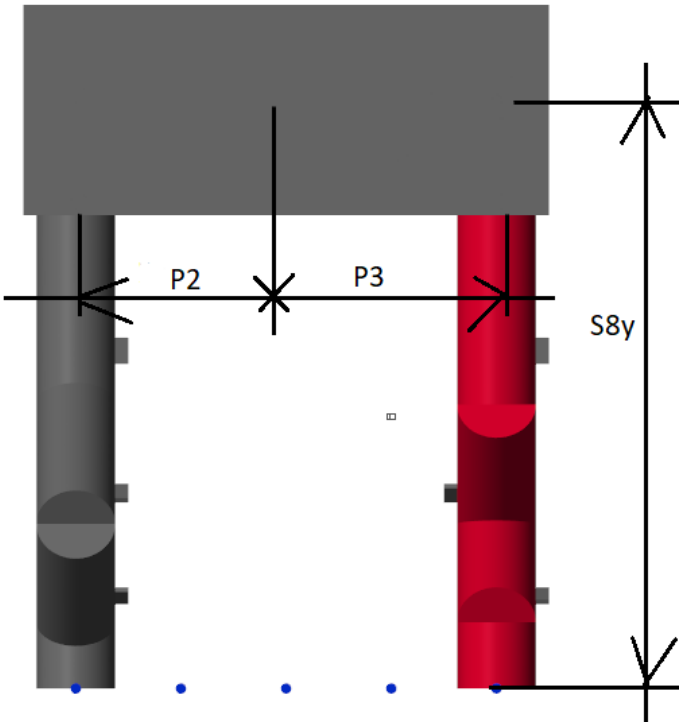


Obrázek 7.2: Optimalizační parametry

Parametr	Spodní mez	Horní mez
$S8_x$ [m]	-0.3	0.3
P_2 [m]	-0.1	0.4
P_3 [m]	-0.4	0.1
$S8_y$ [m]	0.9	1.3
φ_y [rad]	$-\pi/6$	$\pi/6$

Tabulka 7.1: Oblast hledaných parametrů

Aby cílová funkce nespadala do $-\infty$ obvykle se počítá její absolutní hodnota nebo její druhá mocnina. V našem případě stačí absolutní hodnota momentu z každého kloubu. To nám vytvoří matici o 12 cílových funkcích. Celý program běží na 1kHz a celá simulace kráčení trvá 3 vteřiny, ze kterých máme 3 000 hodnot momentů. Abychom dostali jediné charakterizující číslo, které vystihuje zdatnost této generace, vezmeme maximální hodnotu z 12 průběhů momentů a poté ten největší moment, který budeme genetickým algoritmem minimalizovat.



Obrázek 7.3: Optimalizační parametry

7.2.4 Výsledky optimalizace

Jakmile program splní požadavky pro ukončení, zobrazí hlášku a vypíše optimalizační parametry.

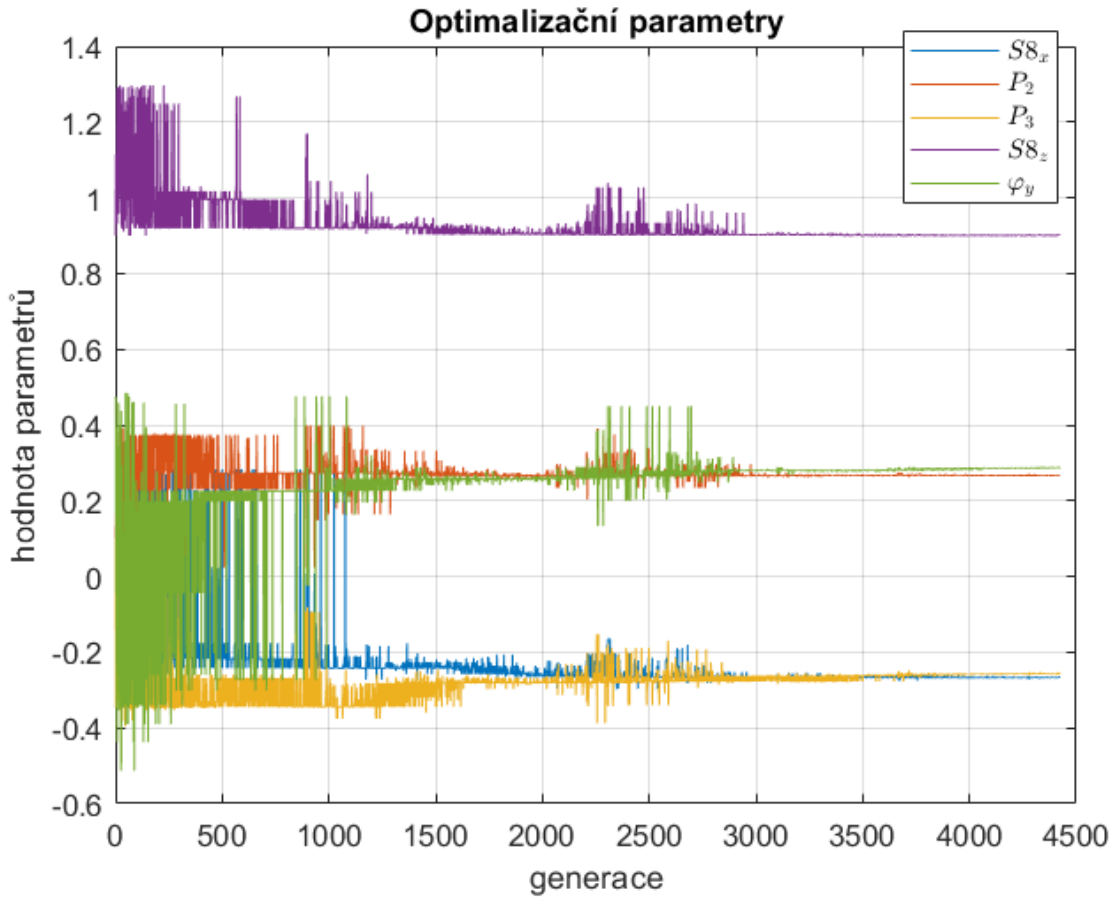
```
Optimization terminated: average change in the fitness value less than options.FunctionTolerance.

x =

   -0.2669    0.2667   -0.2579    0.9005    0.2877
```

Obrázek 7.4: Zakončení genetického algoritmu

Obr. 7.5 ukazuje vývoj optimalizačních parametrů v každé generaci a obr. 7.6 vývoj cílové funkce. Pokud byla hodnota cílové funkce 4000, znamená to, že příkaz "try"nebyl úspěšný a program spadl to sekce "catch", který nastavil cílovou funkci 4000 Nm a přeskočil tuto generaci.

Výsledky optimalizace jsou zobrazeny na obr. 7.4 a jsou uvedeny v tab. 7.2.

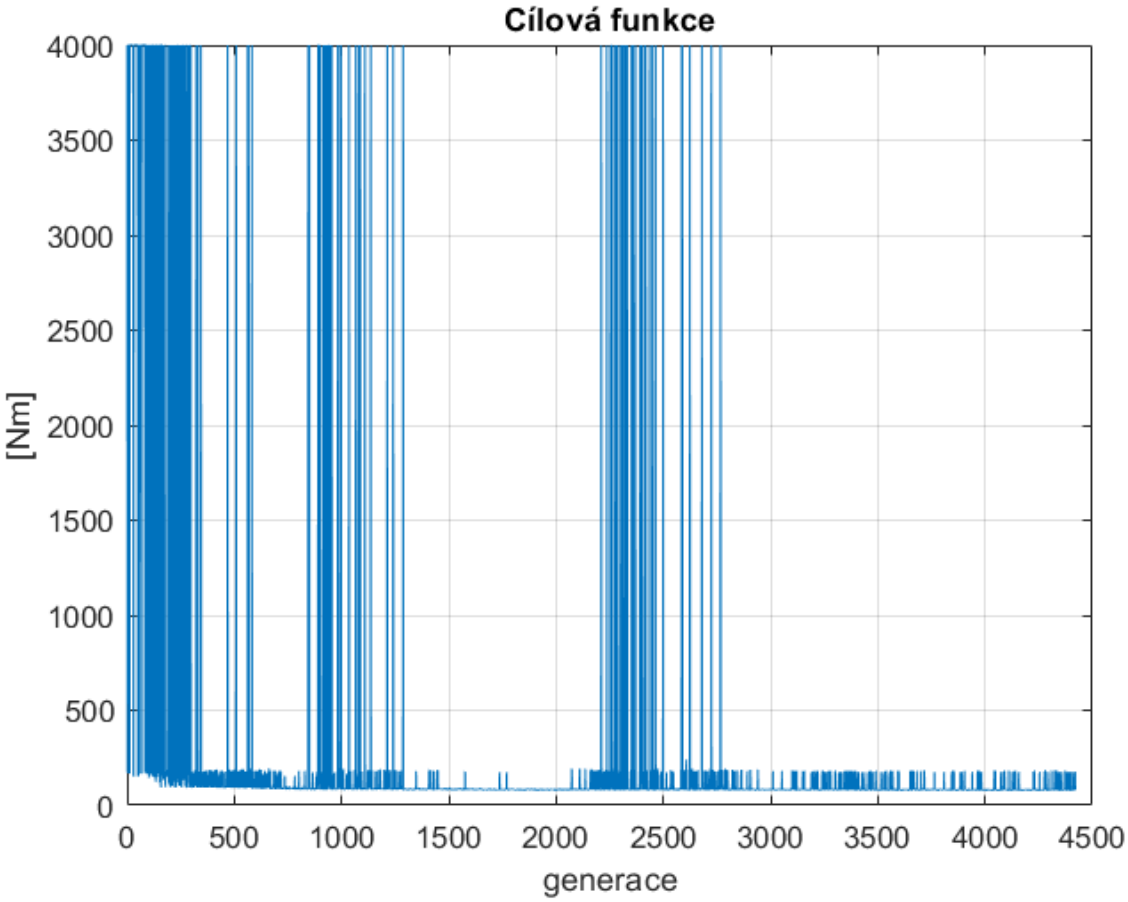
Po aplikování výsledných parametrů na model inverzní dynamiky dostaneme zlepšené výsledky. Obr. 7.7 ukazuje vývoj momentů po procesu optimalizace. V Obr. 5.5 je největší hodnota momentu v kloubu mezi tělesy 3 a 4 (modrý). Tato hodnota momentu je po optimalizaci naopak nízká. Můžeme si všimnout, že se hodnoty ostatních momentů během simulace zvýšily. Optimalizační program našel řešení, které více vyrovná jednotlivé momenty, které už nejsou dominantní pouze v jednom kloubu, ale ve všech kloubech. Maximální moment při původních parametrech byl $175Nm$. Po optimalizaci byl snížen na 93 Nm, což je rozdíl 48%. Obr. 7.8 poté ukazuje skutečnou pozici kráčejícího robota.



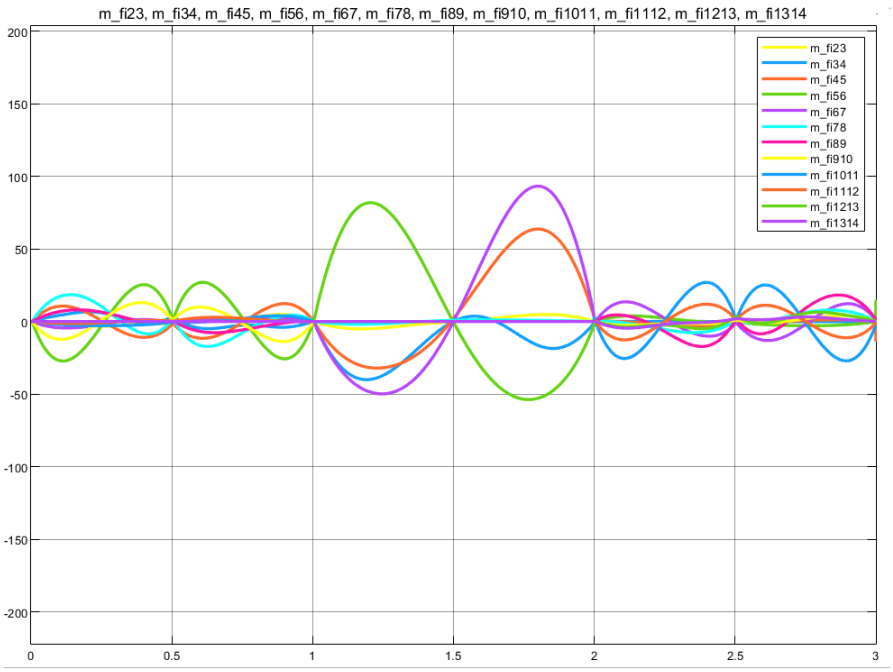
Obrázek 7.5: Vývoj optimalizačních parametrů

Parametr	Hodnota
$S8_x$ [m]	-0.2669
P_2 [m]	0.2667
P_3 [m]	-0.2579
$S8_x$ [m]	0.9005
φ_y [m]	0.2877

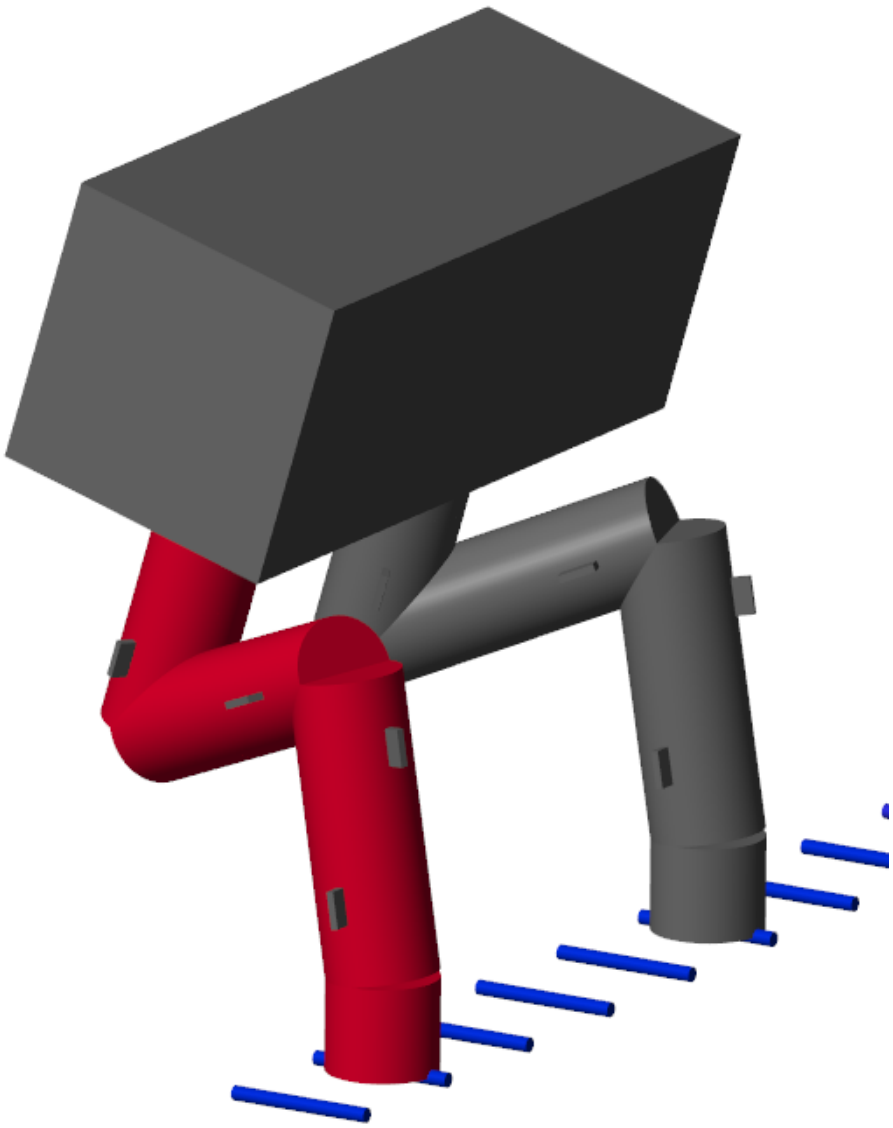
Tabulka 7.2: Výsledky optimalizačních parametrů



Obrázek 7.6: Vývoj cílové funkce



Obrázek 7.7: Průběh momentu po optimalizaci



Obrázek 7.8: Konečná podoba robota

Závěr

Tato práce se zabývá návrhem kráčejícího robota na vesmírné stanici ISS. Na začátku práce byly popsány roboty, které již na stanici pracují, následně byl předestřen základní koncept našeho kráčejícího robota. V programu Matlab/Simulink, s využitím knihovny Simscape-Multibody, byl sestaven jeho pokusný model. Navrhli jsme trajektorii jeho chůze a dále využili možností Simscape pro výpočet inverzní kinematické úlohy.

V dalším kroku jsme použili natočení kloubů robota z inverzní kinematiky na výpočet momentů pro zadaný pohyb. Protože z původně navrženého pohybu byly vypočítané hodnoty momentů příliš velké, přešli jsme na novou definici trajektorie pomocí polynomu 5. stupně, který pak zaručuje hladký náběh momentu. Abychom se vyhnuli nechtěným kinematickým smyčkám, robota jsme museli rozdělit na 3 submodely.

Po stanovení finálních hodnot momentů jsme navrhli řízení robota pomocí kaskádní regulace aplikované na každý jeho pohon. Využili jsme dopřednou složku již vypočítaných momentů ke zlepšení jeho řízení.

V závěru práce pro snížení nároků na konstrukci a řízení robota byly optimalizovány jeho parametry pomocí genetického algoritmu, který umožní minimalizovat hodnoty momentů při jeho pohybu. Výsledek optimalizace snížil hodnoty těchto momentu o 48 %.

Seznam použitých zdrojů

[1] https://www.issnationallab.org/the-engineering-feat-iss-robotics/

[2] https://www.asc-csa.gc.ca/eng/canadarm3/about.asp

[3] https://www.space.com/5058-space-station-ready-robot-room.html

[4] https://cs.wikipedia.org/wiki/MATLAB

[5] https://www.mathworks.com/products/simscape.html

[6] https://www.kuka.com/en-us/products/robotics-systems/industrial-robots/lbr-iiwa

[7] V. Stejskal, M. Valasek, Kinematics and Dynamics of Machinery, Marcel Dekker, Inc. New York, 1996

[8] https://cgg.mff.cuni.cz/~pepca/prg022/luner.html

[9] HOFREITER M. Základy Automatického Řízení. Praha: ČVUT, 2012. ISBN 978-80-01-05007-1

[10] https://www.kuka.com/-/media/kuka-downloads/imported/8350ff3ca11642998dbdc81dcc2ed44c/0000246833_cs.pdf?rev=cf5d1681985c4d90acb59f020a89ccb1&hash=3B24A9BFC6967585043EED25159C7BD6

[11] https://www.kuka.com/cs-cz/produkty,-slu%C5%BEby/robotick%C3%A9-syst%C3%A9my/software/pl%C3%A1nov%C3%A1n%C3%AD-projektov%C3%A1n%C3%AD-servis-bezpe%C4%8Dnost/kuka,-d-,sim

[12] https://cs.wikipedia.org/wiki/Genetick%C3%BD_algoritmus

[13] Sciavicco, L., Siciliano, B.: Modeling and control of robot manipulators, McGraw Hill, New York 1996