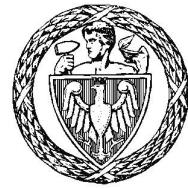


Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa magisterska

na kierunku Automatyka i Robotyka

Kompensacja siły grawitacji związanej z chwytnym
przedmiotem za pomocą regulatorów PID

Jakub Postępski
Numer albumu 257947

promotor
dr inż. Tomasz Winiarski

Warszawa 2019

Streszczenie

Tytuł: *Kompensacja siły grawitacji związanej z chwytanym przedmiotem za pomocą regulatorów PID*

Zastosowanie siłowych praw sterowania daje możliwość łatwej kooperacji robota z człowiekiem. Jednym z nich jest sterowania impedancyjne. W trakcie manipulacji z chwytaniem przedmiotów okazuje się, że siła grawitacji przedmiotu nie jest kompensowana przez silniki robota. Celem pracy jest implementacja algorytmu kompensującego wpływ siły grawitacji chwytanego przedmiotu o nieznanych parametrach masy i inercji w robocie sterowanym impedancyjnie.

Wykorzystywany do badań robot Velma zbudowany jest z dwóch ramion LWR-4 osadzonych na ruchomym korpusie. Oprogramowanie robota zostało zaprojektowane przy pomocy teorii agenta upustociowionego i zaimplementowane przy użyciu struktury ramowej FABRIC. Robot korzysta z impedancyjnego prawa sterowania i samodzielnie kompensuje siłę grawitacji ramion.

Słowa kluczowe: *Velma, FABRIC, ROS, LWR*

Abstract

Title: *Grasping objects in impedance control robot*

The use of impedance and force control gives the possibility of easy cooperation between a robot and a human being. During manipulation with gripping objects, it appears that the gravitational force of the object is not compensated by the robot motors. The aim of this thesis is to implement an gravity compensation algorith for impdiance control with unknown tool parameters.

The Velma robot used for research. It consists of two LWR-4 arms mounted on a movable body. The robot software has been designed with embodied agent theory and implemented using the FABRIC framework. The robot uses impedance control and independently compensates gravity of the arms.

Keywords: *Velma, FABRIC, ROS, LWR*



„załącznik nr 3 do zarządzenia nr 24/2016 Rektora PW

.....
miejscowość i data

.....
imię i nazwisko studenta

.....
numer albumu

.....
kierunek studiów

OŚWIADCZENIE

Świadomy/-a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że niniejsza praca dyplomowa została napisana przeze mnie samodzielnie, pod opieką kierującego pracą dyplomową.

Jednocześnie oświadczam, że:

- niniejsza praca dyplomowa nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym,
- niniejsza praca dyplomowa nie zawiera danych i informacji, które uzyskałem/-am w sposób niedozwolony,
- niniejsza praca dyplomowa nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów lub tytułów zawodowych,
- wszystkie informacje umieszczone w niniejszej pracy, uzyskane ze źródeł pisanych i elektronicznych, zostały udokumentowane w wykazie literatury odpowiednimi odnośnikami,
- znam regulacje prawne Politechniki Warszawskiej w sprawie zarządzania prawami autorskimi i prawami pokrewnymi, prawami własności przemysłowej oraz zasadami komercjalizacji.

Oświadczam, że treść pracy dyplomowej w wersji drukowanej, treść pracy dyplomowej zawartej na nośniku elektronicznym (płycie kompaktowej) oraz treść pracy dyplomowej w module APD systemu USOS są identyczne.

.....
czytelny podpis studenta”

Spis treści

1 Wstęp	13
2 Podstawy teoretyczne	15
2.1 Sterowanie impedancyjne	15
2.1.1 Przypadek prosty	15
2.1.2 Prawo sterowania	16
2.1.3 Sterowanie w przestrzeni konfiguracyjnej	16
2.2 Sterowanie PID	16
2.2.1 Przypadek prosty	17
2.2.2 Przypadek wielowymiarowy	17
2.3 Estymacja siły uogólnionej w końcówce	17
2.4 Jakość sterowania	18
2.5 Teoria agenta upostaciowanego	19
3 Środowisko badawcze	21
3.1 Budowa ogólna	21
3.2 Agenty	21
3.3 Pakiety oprogramowania	24
3.3.1 ROS	24
3.3.2 Orocó	25
3.3.3 FABRIC	25
3.4 Symulator robota	26
4 Specyfikacja systemu kompensacji grawitacji	27
4.1 Analiza wymagań	28
4.1.1 Przypadki użycia	28
4.1.2 Założenia	29
4.1.3 Wymagania	30
4.2 Rozwiązanie oparte na regulatorze PID	30
4.3 Rozwiązanie oparte na estymacji modelu robota	31

SPIS TREŚCI

4.4	Wybór rozwiązania	33
5	Implementacja systemu	35
5.1	Prawo sterowania	35
5.2	Parametry członu całkującego	36
5.3	Rejestracja trajektorii	36
5.4	Generowanie wykresów	37
5.5	Opis świata testowego	37
5.6	Algorytm eksperymentu	37
6	Działanie systemu	39
6.1	Podnoszenie przedmiotu	41
6.2	Ruch ósemkowy	44
6.3	Ruch w bok	47
6.4	Ruch do góry	50
6.5	Ruch do dołu	52
6.6	Ruch do przodu	55
6.7	Obrót końcówki	57
6.8	Wnioski	60
7	Podsumowanie	63

Rozdział 1

Wstęp

Współczesne roboty z powodzeniem zastępują człowieka przy wielu żmudnych i niewdzięcznych czynnościach. W zakładach produkcyjnych sprawują się znakomicie ale do współpracy robotów z ludźmi podchodzi się niezwykle ostrożnie. Niechęć do tego typu działań podyktowana jest nie tylko wysokimi kosztami ale także obawami związanymi z ochroną otoczenia, w którym robot ma operować.

Zakres zastosowań robotypki można znacznie poszerzyć projektując roboty bezpieczne zarówno dla otaczającego je środowiska jak i samych robotów. Jednym z fundamentów takiego bezpieczeństwa może być algorytm sterowania impedancyjnego, który znacznie ogranicza ryzyko zniszczeń.

Prawo sterowania jest skonstruowane tak by silniki ramienia działały w sposób symulujący układ ze sprężyną i amortyzatorem. Robot sterowany w taki sposób nie dąży do zadanej pozycji za wszelką cenę. W momencie kolizji robot pozostaje elastyczny i ugina się. Opisana zaleta może przerodzić się w wadę. Algorytm korzysta ze zdefiniowanego modelu, który nie uwzględnia wszystkich cech ramienia oraz chwytyanych przez robota przedmiotów. Z punktu widzenia prawa sterowania chwycony przedmiot jest traktowany tak samo jak reszta środowiska. Zamiast skompensować siłę grawitacji tego przedmiotu ramię robota ugina się pod ciężarem. Dalsza manipulacja ramieniem nie ma najczęściej sensu.

Celem pracy jest rozwiązywanie problemu kompensacji siły grawitacji przedmiotu o nieznanej masie i inercji w opisany środowisku. Do badań zostanie wykorzystany robot Velma oraz jego symulator skonstruowane przez Zespół Programowania Robotów i Systemów Rozpoznających. W niniejszej pracy zostanie zaprezentowana praktyczna realizacja sposobu kompensacji grawitacji chwyconego narzędzia wraz z testami.

ROZDZIAŁ 1. WSTĘP

Rozdział 2

Podstawy teoretyczne

Dyskutowane w pracy algorytmy znane są w wielu różnych odmianach. Poniżej zaprezentowano wersje używane w trakcie dalszych badań.

Do dalszych rozważań możemy zdefiniować uchyb jako:

$$\mathbf{e}_x = \mathbf{x}_d - \mathbf{x} \quad (2.1)$$

gdzie:

- \mathbf{x}_d to wektor zadanych pozycji uogólnionych
- \mathbf{x} to wektor pozycji uogólnionych

2.1 Sterowanie impedancyjne

Prawo sterowania impedancyjnego [17], [23] w przestrzeni operacyjnej sprawia, że chwytak robota zachowuje się jak przytwierdzony do układu ze sprężyną i amortyzatorem, gdzie drugi koniec tego układu jest pozycją zadaną. Pojawienie się nieprzewidzianych sił zewnętrznych powoduje, że uchyb pozycji nie jest minimalizowany za wszelką cenę. Przy kontakcie z otoczeniem stawy robota są w pewnym stopniu sprężyste i miękkie. W konsekwencji robot ugina się przed otaczającym go środowiskiem a w szczególności w chwili zderzenia z nieprzewidzianą przeszkodą.

2.1.1 Przypadek prosty

W najprostszym przypadku możemy więc opisać takie prawo jako układ:

$$F = kx + d\dot{x} + F_{ext} \quad (2.2)$$

gdzie:

- F to siła wynikowa

- k to parametr sztywności
- d to parametr tłumienia
- F_{ext} to nieznana siła zewnętrzna działająca na układ

2.1.2 Prawo sterowania

Prawo sterowania można sformułować jako wektor siły uogólnionej:

$$\mathcal{F} = \mathbf{K}_x e_x + \mathbf{D}_x \dot{e}_x \quad (2.3)$$

gdzie:

- \mathbf{K}_x to diagonalna macierz sprężystości
- \mathbf{D}_x to diagonalna macierz tłumienia
- \mathcal{F}_{ext} to wektor nieznanych uogólnionych sił zewnętrznych

2.1.3 Sterowanie w przestrzeni konfiguracyjnej

W rzeczywistym ramieniu robotycznym zadajemy momenty na poszczególne stawy robota. Opisane w podrozdziale 2.1.2 prawo sterowania opisuje przestrzeń operacyjną \mathcal{F} . Wartości wektora momentów $\boldsymbol{\tau}$ można uzyskać z jakobianem \mathbf{J} :

$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q}) \mathcal{F} \quad (2.4)$$

2.2 Sterowanie PID

Regulator PID (ang. Proportional Integral Derivative) [15], [22] jest popularnym prawem sterowania automatycznego. Podstawowym celem algorytmu jest minimalizacja uchybu. Uchyb statyczny jest minimalizowany za wszelką cenę nawet jeśli miałoby to spowodować uszkodzenia. Uchyby dynamiczne nie są już tak dobrze kompensowane.

Człon proporcjonalny algorytmu pozwala na wzmacnienie uchybu i w ten sposób odjęcie go od sygnału sterującego. Człon całkujący algorytmu sumuje przeszłe błędy i odejmuje od sterowania ich sumę. Człon różniczkujący wzmacnia sygnał sterujący w gdy wartość błędu zmienia się w celu przyspieszenia regulacji.

2.2.1 Przypadek prosty

W jednowymiarowym przypadku prawo sterowania ma postać:

$$F = Pe + I \int_0^t edt + D \frac{de}{dt} \quad (2.5)$$

gdzie:

- e to uchyb
- P to parametr członu proporcjonalnego
- I to parametr członu całkującego
- D to parametr członu różniczkującego

2.2.2 Przypadek wielowymiarowy

Rozpatrując wektor siły uogólnionej możemy założyć w uproszczeniu, że prawo sterowania rozpatruje każdą z wartości wektora niezależnie. Prawo sterowania można zapisać w postaci:

$$\mathcal{F} = \mathbf{P}\mathbf{e}_x + \int_0^t \mathbf{I}\mathbf{e}_x dt + \mathbf{D}\dot{\mathbf{e}}_x \quad (2.6)$$

gdzie:

- \mathbf{P}_x to diagonalna macierz proporcjonalności
- \mathbf{I}_x to diagonalna macierz członu całkowania
- \mathbf{D}_x to diagonalna macierz członu różniczkującego

2.3 Estymacja siły uogólnionej w końcówce

Dla ramienia robotycznego możemy opisać siły występujące w samym ramieniu zgodnie ze wzorem:

$$\mathcal{F}_m = \mathcal{F}_m(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\Lambda}(\mathbf{q})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\gamma}(\mathbf{q}) + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) + \mathcal{F}_{ext} \quad (2.7)$$

gdzie:

- \mathcal{F} to wektor sił wynikowych
- $\boldsymbol{\Lambda}$ to dodatnio określona macierz inercji w przestrzeni zadań
- $\boldsymbol{\mu}$ to macierz sił Coriolisa i sił odśrodkowych

- γ to wektor sił grawitacji
- η to macierz sił tarcia oraz nieuwzględnionych sił
- q to wektor położen stawów w przestrzeni konfiguracyjnej
- x to wektor położen końcówki w przestrzeni zadań
- \mathcal{F}_{ext} to nieznany wektor sił zewnętrznych działających na układ

Przy wyliczaniu estymowaniu sił działających na końówkę należy pamiętać, że w końówce występują siły wygenerowane przez prawo sterowania oraz rzeczywiste siły występujące w układzie. Wzór estymujący rzeczywistą wartość siły uogólnionej w końówce można zapisać jako:

$$\hat{\mathcal{F}} = \mathcal{F} + \mathcal{F}_m(x, \dot{x}, \ddot{x}, q, \dot{q}) \quad (2.8)$$

gdzie \mathcal{F} to wektor sił uogólnionych wyliczony prawem sterowania.

2.4 Jakość sterowania

Prostym sposobem oceny jakości algorytmu sterowania jest konfrontacja rzeczywistych pozycji ramienia robota z zadanimi. W pracy przyjęto metrykę APE (ang. Absolute Trajectory Error) [19]. Metryka jest popularnym wskaźnikiem testowania algorytmów SLAM (ang. Simultaneous Localization and Mapping) ale może być też użyta do porównania trajektorii zadanej przez interpolator oraz osiągniętej w rzeczywistości (rys. 2.1).

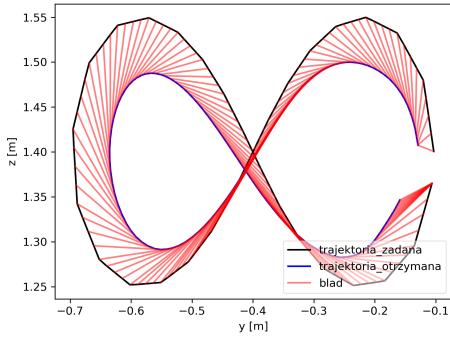
Dwie trajektorie są opisane w postaci list wektorów sił uogólnionych $P_{1..n}$ oraz $Q_{1..n}$ gdzie n to ilość próbek. Dla każdej chwili czasowej i jest wyliczany błąd postaci:

$$E_i = Q_i^{-1} S P_i \quad (2.9)$$

Macierz S jest optymalnym w sensie metody najmniejszych kwadratów rzutowaniem wektora Q_i na wektor P_i znalezionym za pomocą metody Horna [12].

Błąd całkowity jest wyliczany jako błąd średnio-kwadratowy:

$$RMSE(E_{1..n}) = \sqrt{\frac{1}{n} \sum_{i=1}^n \|E_i\|^2} \quad (2.10)$$



Rysunek 2.1: Rysunek przedstawia rzut APE w dwóch wymiarach. Czarną i niebieską linią oznaczono zadaną i osiąganą trajektorię. Linie czerwone łączą odpowiednie punkty na obydwu trajektoriach i obrazują błąd pomiędzy nimi.

Błąd całkowity jest wyliczany jako błąd średnio-kwadratowy:

$$RMSE(\mathbf{R}_{1..n}) = \sqrt{\frac{1}{n} \sum_{i=1}^n ||\mathbf{q}_d - \mathbf{q}_i||^2} \quad (2.11)$$

2.5 Teoria agenta upostaciowanego

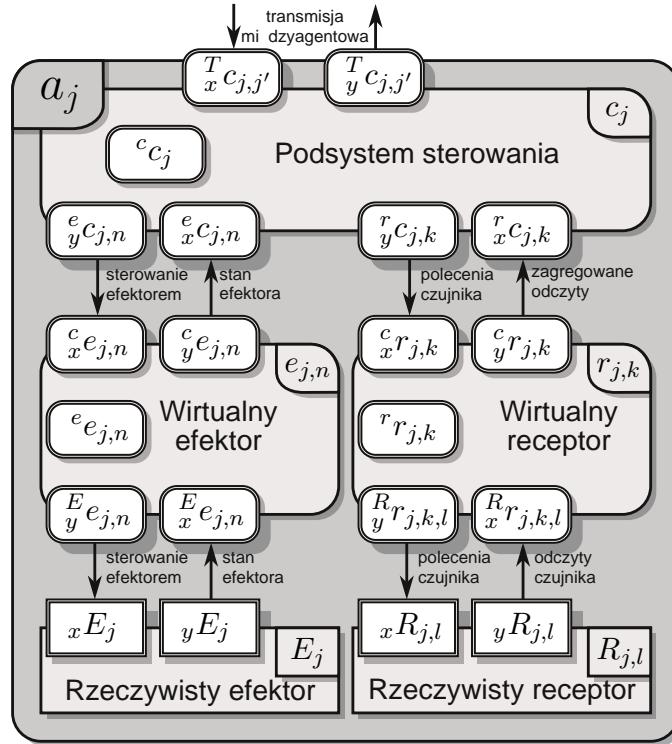
Agentem możemy nazwać jednostkę, która jest: zdolna do komunikowania się ze środowiskiem, zdolna do monitorowania swego otoczenia i podejmowania autonomicznych decyzji. Taka definicja gwarantuje spełnienie wielu cech, których oczekujemy od nowoczesnych systemów robotycznych. Z definicji agenty są w stanie samodzielnie reagować na zmiany zachodzące w środowisku w sposób inteligentny. Rozszerzeniem teorii agentowej jest teoria agenta upostaciowanego [29] [28]. Agent upostaciowany cechuje się tym czym zwykły agent oraz posiada fizyczne ciało. Agenty oraz podsystemy mogą komunikować się poprzez bufory.

W teorii agenta upostaciowanego występują pojęcia rzeczywistych efektorów i receptorów. Służą one odpowiednio do oddziaływania na środowisko i do pozyskiwania wiedzy o tym środowisku. W praktyce oznacza to, że rzeczywistym efektorem może być silnik a rzeczywistym receptorem enkoder.

Agent upostaciowany a (rys. 2.2) może się składać z trzech podsystemów:

- Podsystemu sterowania c który zajmuje się podejmowaniem decyzji na podstawie danych z innych podsystemów. Agent może mieć tylko jeden podsystem sterowania.
- Wirtualnego efektora e który pośredniczy w komunikacji pomiędzy podsystemem sterowania i rzeczywistym efektorem.

ROZDZIAŁ 2. PODSTAWY TEORETYCZNE



Rysunek 2.2: Rysunek przedstawia schemat budowy agenta upustacjowionego [?]. Typ podsystemu oznaczany jest dużą literą. Podsystemy i bufore posiadają prawy dolny indeks. Typ podsystemu jest oznaczany lewym dolnym indeksem. Typ buforu jest oznaczany lewym dolnym indeksem (x - wejściowy, y - wyjściowy).

- Wirtualnego receptora r który pośredniczy w komunikacji pomiędzy podsystemem sterowania i rzeczywistym receptorem.

W podejściu komponentowym wspomnianej teorii podsystemy składają się z komponentów czyli algorytmów. Każdy z podsystemów może mieć wiele komponentów. Podsystem posiada zdefiniowaną maszynę stanów. Stany maszyny mogą się zmieniać przy pomocy funkcji przejścia (predykatów). Stan maszyny definiuje tak zwane zachowanie podsystemu czyli sposób reakcji poprzez uruchomienie konkretnych komponentów.

Rozdział 3

Środowisko badawcze

3.1 Budowa ogólna

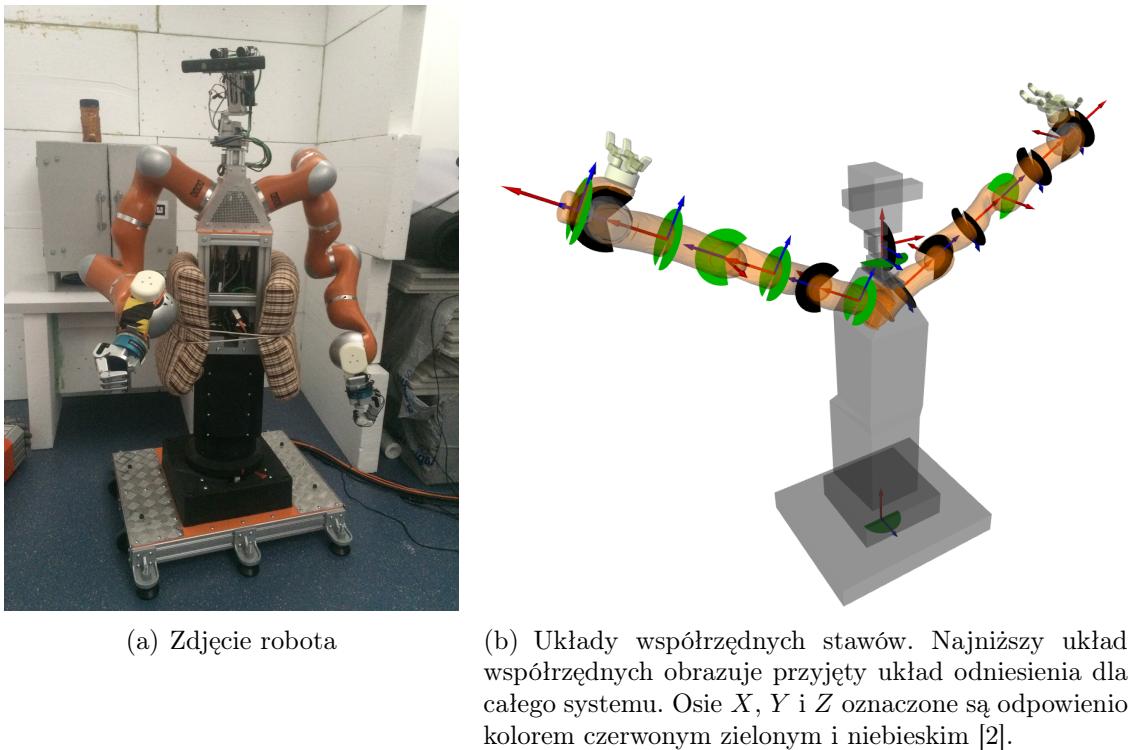
Środowiskiem badawczym jest robot usługowy Velma (rys. 3.1) [2], [26]. Został zaprojektowany i wykonany przez Zespół Programowania Robotów i Systemów Rozpoznających [8]. Do obrotowego korpusu przytwierdzono dwa ramiona robotyczne Kuka LWR-4+ [3] [11]. Mają siedem stopni swobody i udźwig 7 kg. Na ich końcach znajdują się chwytaki Barretta [1] oraz nadgarstkowe czujniki FTS (ang. Force Torque Sensor). Głowa robota umieszczona jest na dedykowanej konstrukcji która za pomocą dwóch silników elektrycznych pozwala na zginanie i obrót głowy [18] [25]. Robot wyposażony jest w czujnik wizyjny Microsoft Kinect oraz dwie kamery połączone w stereoparę. Do głowy zamocowano mikrofon. Podzespoły robota są połączone z komputerem sterującym przy pomocy magistral czasu rzeczywistego.

System sterowania o twardych ograniczeniach czasowych pracuje z częstotliwością 500 Hz. Struktura oprogramowania została stworzona w oparciu o teorię agentową. Oprogramowanie robota pisane jest przy wykorzystaniu struktury ramowej FABRIC. Programy nadzoruje system Linux z nakładką Linux-RT. Dostępny jest symulator robota stworzony w przy wykorzystaniu Gazebo i silnika fizyki DART.

3.2 Agenty

System sterowania [2] zbudowany jest z dwóch agentów (rys. 3.2). Agent *velma_core* jest odpowiedzialny za kontrolę zadań związanych z manipulacją w przestrzeni operacyjnej i konfiguracyjnej robota. Drugi z agentów *velma_task_cs_ros_interface* jest interfejsem pomiędzy programami użytkownika pisanymi w ROS oraz agentem *velma_core*. Zawiera tylko jeden podsystem o tej samej nazwie. Użytkownikom został udostępniony dodatkowy interfejs *VelmaInterface* napisany w Pythonie. Nie jest on częścią agenta *velma_task_cs_ros_interface*

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE

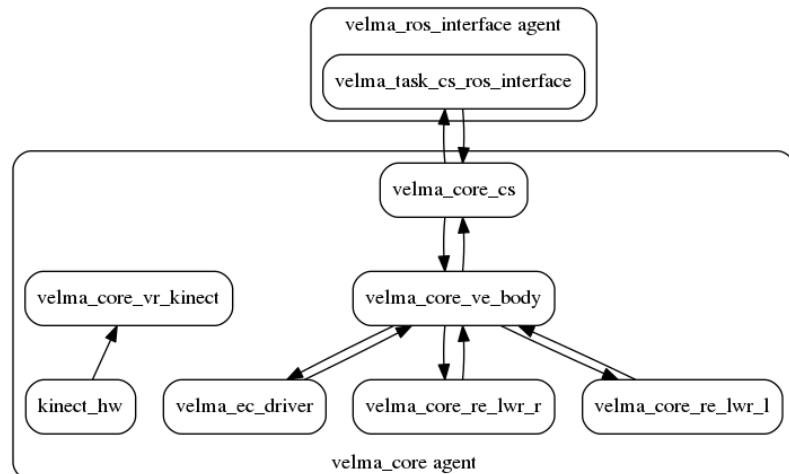


Rysunek 3.1: Robot usługowy Velma

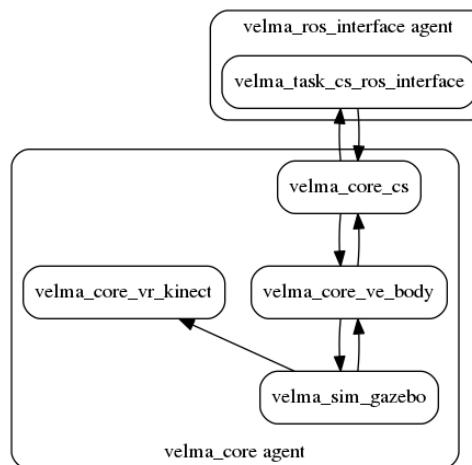
i służy jedynie jako pomoc przy komunikacji z agentem z poziomu programów użytkownika.

Podsystem *velma_core_cs* agenta *velma_core* wylicza prawa sterowanie oraz zajmuje się interpolacją trajektorii. Podsystem *velma_core_ve_body* kontroluje bazowe zachowania bezpieczeństwa. Są to ograniczenia prądowe, wykrywanie krańcowych położen stawów oraz wykrywanie kolizji. Podsystem przekazuje też sterowanie do efektorów. Podsystemy w najniższej warstwie abstrakcji mogą być stosowane wymiennie. Do pracy w rzeczywistym świecie uruchamiane są podsystemy *velma_core_re_lwr_r* i *velma_core_re_lwr_l* oraz podsystem *velma_ec_driver*. Służą odpowiednio do kontroli prawego i lewego ramienia LWR-4 oraz pozostałego sprzętu połączonego magistralą EtherCAT. Do pracy w trybie symulacji podsystemy w najniższej warstwie abstrakcji są wymieniane na podsystem *velma_sim_gazebo*, w którym uruchamiany jest symulator świata Gazebo. Podsystem symuluje pracę wszystkich trzech podsystemów odpowiedzialnych za komunikację ze sterownikami sprzętu.

- **Interfejs akcji ROS** - Komponent *CartImpActionRight* znajduje się w podsystemie *velma_task_cs_ros_interface* i odbiera rozkazy wysłane przez użytkownika przez mechanizm akcji ROS oraz przesyła je do innych podsystemów. Zwraca też status wykonania operacji. Komponent służy do obsługi prawego ramienia i istnieje analogiczny komponent *CartImpActionLeft* służący do ob-



(a) Tryb pracy ze sprzętem



(b) Tryb symulacji

Rysunek 3.2: Poglądowy i uproszczony schemat agentów robota Velma zawierający także nieopisywane części systemu odpowiedzialne za nadzór Kinecta [2].

sługi lewego ramienia.

- **Publikacja wektorów pozycji** - Komponent *TfPublisher* wysyła do użytkownika za pomocą ROSa położenie i obrót istotnych dla działania systemu miejsc robota takich jak chwytki czy środek ciężkości korpusu.
- **Pozycje stawów** - Odczyt pozycji stawów jest zależny od trybu pracy oprogramowania. Jeśli używamy trybu obsługi rzeczywistego sprzętu to za odczyt pozycji są odpowiedzialne podsystemy *velma_core_re_lwr_r* i *velma_core_re_lwr_l* oraz podsystem *velma_ec_driver* które posiadają pojedyncze komponenty służące do komunikacji z fizycznymi sterownikami magistral. W trybie symulatora emulacją tych trzech podsystemów zajmuje się podsystem *velma_sim_gazebo*, który jako symulator nie jest podzielony na konkretne podsystemy.
- **Zadawanie momentów** - Zadawanie momentów obrotowych w stawach następuje w analogiczny do odczytywania pozycji sposób.
- **Kinematyka prosta** - Komponent *FK* pobiera dane o pozycjach stawów i wylicza pozycję chwyptaka oraz innych części robota.
- **Interpolator trajektorii** - Komponent *INT_tool_r* z podsystemu *velma_core_cs* odpowiada za interpolacje trajektorii zadanej prawemu ramieniu. Interpolator służy temu by jedno polecenie przesunięcia ramienia zamienić na wiele mniejszych i rozłożonych w czasie. W konsekwencji algorytm sterowania nie jest narażony na duże uchyby a ruch ramienia jest wykonywany płynnie. Do działania komponent potrzebuje aktualnych i zadanych pozycji. Komponent służy do obsługi prawego ramienia. Analogicznie do obsługi lewego ramienia służy komponent *INT_tool_l*.
- **Prawo sterowania impedancyjnego** - Komponent *cart_imp* z podsystemu *velma_core_cs* służy do wyliczania momentów zadawanych na stawy zgodnie z algorytmem prawa sterowania impedancyjnego w przestrzeni kartezjańskiej. Pobiera zadaną pozycję z interpolatora trajektorii.

3.3 Pakiety oprogramowania

3.3.1 ROS

Struktura ramowa ROS (Robot Operating System) [7] zapewnia biblioteki usprawniające pisanie programów dla robotyki w C++ i Pythonie. Pozwala na komunikację pomiędzy programami na zasadzie tematów oraz na zasadzie usług. Posiada

gotowe funkcje wspomagające obliczenia kinematykę i wizualizujące pracę robota. Pakiet daje możliwość akwizycji danych. Podstawowymi narzędziami w pakiecie są:

- **Węzły** - Programy pisane w C++ lub Pythonie spełniające zdefiniowaną w systemie funkcję.
- **Serwisy** - Usługi udostępniane przez węzły pozwalające na komunikację w architekturze zapytanie odpowiedź. Każdy program może udostępniać wiele serwisów. Służą do wywoływania zdalnych procedur.
- **Akcje** - Usługi podobne do serwisów lecz nieblokujące wykonywania węzła będącego serwerem.
- **Tematy** - Usługi udostępniane przez węzły pozwalające na asynchroniczną komunikację. Każdy program może udostępniać i pobierać wiele tematów. Służą do aktualizowania bieżącego stanu całego systemu.
- **Zarządca** - Odpowiada za komunikację i nadzoruje pracę innych narzędzi pakietu.

3.3.2 Orocoss

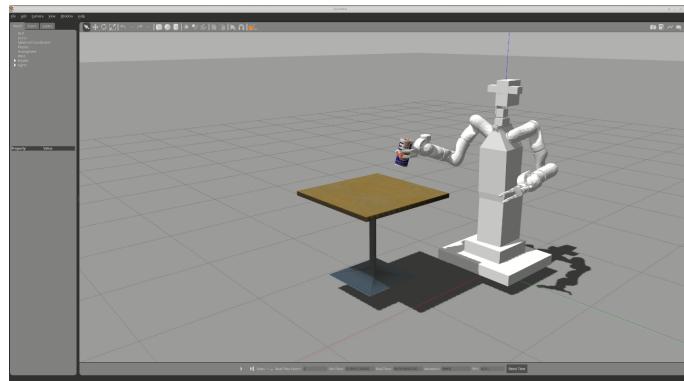
Orocoss [6] jest wolnym oprogramowaniem napisanym w C++ służącym do tworzenia aplikacji zgodnych z wymaganiami czasu rzeczywistego. Pozwala na implementację oprogramowania w podejściu komponentowym. Zestaw bibliotek pakietu Orocoss składa się między innymi z:

- **RTT (ang. Real-Time Toolkit)** - Biblioteki służące do tworzenia komponentów w aplikacjach czasu rzeczywistego.
- **OCL (ang. Orocoss Component Library)** - Biblioteki pomocne w trakcie uruchamiania komponentów.
- **OroGen** oraz **TypeGen** - Narzędzia do automatycznego generowania komponentów i typów danych.
- **Deployer** - Uruchamia komponenty zgodnie z opisem zawartym w pliku XML oraz pozwala na nadzór tych komponentów w trakcie pracy.

3.3.3 FABRIC

Framework for Agent-Based Robot Control Systems - FABRIC [16] wykorzystuje Orocosa oraz strukturę ramową ROS zapewniając interfejs programistyczny pozwalający na tworzenie komponentów zgodnych z założeniami teorii agentowej.

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE



Rysunek 3.3: Robot Velma w trakcie manipulacji. Symulacja w środowisku Gazebo.

Ma zaimplementowane algorytmy komunikacji pomiędzy poszczególnymi podsystemami poprzez zdefiniowane wiadomości. Posiada narzędzie wizualizujące stan predykatów, zachowań i podsystemów *rqt_agent* [27]. Pokazuje też przepływ danych między komponentami.

3.4 Symulator robota

Symulator robota Velma jest wytworzony w oprogramowaniu Gazebo [4] które symuluje obiekty i zachowania między nimi zgodnie z prawami fizyki. Użytkownik ma możliwość zdefiniowania całego środowiska wraz z robotem (rys. 3.3). Posiada gotowe modele wielu receptorów i efektorów oraz pozwala na tworzenia własnych. Daje możliwość emulowania sterowników sprzętu. Interfejs użytkownika udostępnia opcję wywarcie siły bądź momentu na dowolny przedmiot będący w symulacji. Emuluje czas jeśli symulacja nie jest przeprowadzana w czasie rzeczywistym.

Oryginalny kod Gazebo został przystosowany do pakietu oprogramowania DART (ang. Dynamic Animation and Robotics Toolkit) [5] symulującego fizykę zdefiniowanego świata. Pakiet DART został wybrany przez Zespół Programowania Robotów i Systemów Rozpoznających ponieważ po jego zastosowaniu został wyeliminowany problem narastających oscylacji członów robota w trakcie symulacji. W przeciwieństwie do wielu symulatorów fizyki DART daje dostęp do wielu przydatnych informacji takich jak jakobian bądź macierz inercji przedmiotów. Zastosowano w nim algorytm LCP (ang. Linear Complementarity Problem), [20] który sprowadza model fizyczny świata do zestawu równań kwadratowych. DART symuluje działanie wielu sił w tym siły Koriolisa i tarcia dynamicznego. Dzięki temu możliwe jest zaawansowane wykrywanie kolizji.

Rozdział 4

Specyfikacja systemu kompensacji grawitacji

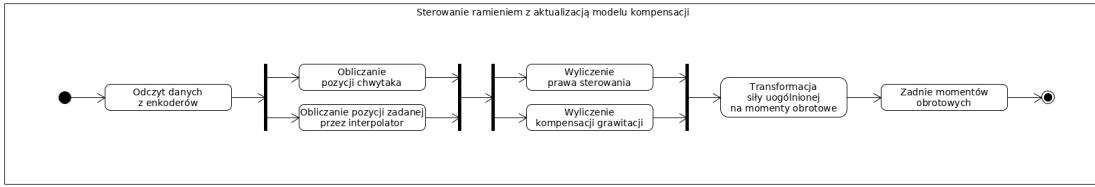
W tym rozdziale zostanie przeprowadzona dyskusja na temat wyboru odpowiedniego rozwiązania które zostanie potem zaimplementowane. Ramiona robota muszą uginać się w trakcie kolizji z otoczeniem. To założenie jest podstawową motywacją do zastosowania prawa sterowania impedancyjnego opisanego w sekcji 2.1.2. Każdy staw robota posiada jeden silnik oraz enkoder i czujnik momentu obrotowego. Z odczytów pozycji i momentu może korzystać prawo sterowania, czyli algorytm wyliczający momenty obrotowe zadawane silnikom. Wyliczanie pozycji końcówki następuje poprzez jakobian. W sterowaniu impedancyjnym momenty zadawane są w taki sposób, że ramię robota zachowuje się jak zawieszone na niewidocznych sprężynach. Algorytm sterowania w przestrzeni kartezjańskiej działa tak, że jeden z jej końców jest przytwierdzony w punkcie zadanym przez użytkownika a drugi koniec do chwytaka.

Punkty zadane są wyznaczane przez interpolator trajektorii na podstawie rozkazów użytkownika. Interpolator ma za zadanie rozłożyć ruch zadany przez użytkownika w czasie. Skokowa zmiana pozycji zadanej do pozycji wskazanej przez użytkownika spowodowałaby oscylacje w stawach i zadanie dużych momentów obrotowych stawom.

Do prawa sterowania impedancyjnego dodany jest algorytm kompensacji grawitacji członów robota. Na podstawie znanego modelu wyliczany jest moment obrotowy, dla każdego stawu, odpowiadający sile grawitacji członów. Momenty o przeciwnym znaku i tej samej wartości są dodawane do momentów sterujących ramieniem co kompensuje siłę grawitacji poszczególnych członów (rys. 4.1). Model może być też rozszerzony i wyliczać momenty dla innych sił np. Koriolisa bądź tarcia.

Zadany odgórnie model stosowany przy wyliczaniu kompensacji grawitacji ma pewne wady. Jeśli model nie jest dokładny siła grawitacji nie zostanie skompensowana właściwie. W trakcie pracy robot zmienia swoje własności np. przez zwiększenie tarcia związanego ze wzrostem temperatury. Dodatkową niedokładność wpro-

ROZDZIAŁ 4. SPECYFIKACJA SYSTMU KOMPENSACJI GRAWITACJI



Rysunek 4.1: Diagram aktywności pokazujący przebieg sterowania ramieniem robota.

wadza zmiana konfiguracji stawów ujętego w modelu jako jeden człon chwytaka. Ważniejsze dla rozważań jest to, że jeśli robot chwyci narzędzie czyli obiekt, którym ma manipulować, to jego parametry nie zostaną uwzględnione w tym modelu. Spowoduje to, że ramiona robota będą opadać.

Najprostszym rozwiązaniem jest poznanie masy narzędzia poprzez odczyt siły grawitacji z czujnika siły zamieszczonego w chwytku a następnie dodanie wartości tej siły o przeciwnym znaku do prawa sterowania. Takie rozwiązanie nie jest dość dokładne. Na pomiar siły mają wpływ inne siły niż siła grawitacji i znacznie zaburzają wynik. Istnieje potrzeba znalezienia bardziej wysublimowanego rozwiązania.

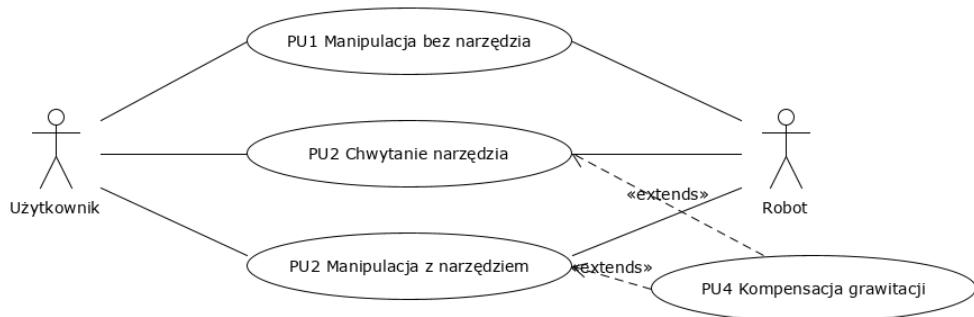
Poniżej zaprezentowano dwa możliwe rozwiązania opisanego problemu. Pierwsze z nich oparte na dodaniu do istniejącego prawa sterowania członu całkującego jest proste w implementacji lecz prawdopodobnie mniej dokładne. Nie pozwala też na poznanie żadnych parametrów narzędzia które jest chwytyane. Druga z, zakładająca obliczenie parametrów modelu narzędzia, jest skomplikowana w implementacji i wymaga większych zasobów obliczeniowych. Stanowi to realny problem w systemie czasu rzeczywistego.

4.1 Analiza wymagań

Rozważania dotyczące możliwych rozwiązań ogólnie zarysowanego problemu można usystematyzować. Środowiskiem badawczym jest robot Velma. Przypadki użycia (rys. 4.2) odzwierciedlają pracę systemu. Algorytm kompensujący grawitację ma działać przez cały czas pracy robota gdy pracuje z narzędziem.

4.1.1 Przypadki użycia

Kompensacja grawitacji jest odzielnym przypadkiem użycia robota, który rozszerza przypadki użyci systemu w trakcie pracy w trybie sterowania impedancyjnego.



Rysunek 4.2: Diagram przypadków użycia robota z uwzględnieniem algorytmu kompensacji grawitacji

Aktorami korzystającymi z systemu są:

- **Robot** - robot wykonujący zadanie chwycenia obiektu
- **Użytkownik** - system zewnętrzny wydający polecenia

Przypadki użycia to:

- **PU1 Manipulacja bez narzędzia** - Konfiguracja ramienia robota może się zmieniać lecz robot nie trzyma żadnego narzędzia. Algorytm kompensacji grawitacji nie powinien zmieniać prawa sterowania.
- **PU2 Chwytyanie narzędzia** - Robot zaciska chwytek na narzędziu o nieznanym parametrach a następnie je podnosi.
- **PU3 Manipulacja z narzędziem** - Konfiguracja ramienia może się zmieniać a algorytm kompensacji grawitacji powinien przeciwdziałać sile grawitacji chwyconego narzędzia.
- **PU4 Kompensacja grawitacji** - Robot kompensuje siłę grawitacji chwyconego narzędzia

4.1.2 Założenia

Konkretnie wymagania dotyczące systemu są podyktowane środowiskiem badawczym oraz potrzebami Zespołu Programowania Robotów i Systemów Rozpoznających.

- System dostarcza gotowe komponenty potrzebne do pracy ramienia, w szczególności: generatora trajektorii, interpolatora oraz wyznaczania kinematyki prostej
- System pozyskuje dane o otoczeniu na podstawie odczytów z czujników położenia stawów.

- System samodzielnie kompensuje siłę grawitacji związaną z masą wszystkich członów robota.
- Parametry chwytanego obiektu związane z masą i inercją nie są znane i mogą być zmienne w czasie.

4.1.3 Wymagania

Przedstawione wymagania pozwalają na wyspecyfikowanie założeń dotyczących modyfikacji oprogramowania robota. Algorytm kompensacji grawitacji musi wyliczać dodatkowy moment w stawach robota. Do momentów obrotowych wyliczanych na postawie prawa sterowania impedancyjnego w przestrzeni operacyjnej robota należy dodać te związane z kompensacją masy narzędzia.

- Algorytm kompensacji grawitacji zostanie dodany do komponentu odpowiedzialnego za wyliczanie prawa sterowania impedancyjnego.
- Trajektoria końcówki ramienia powinna być zbliżona do trajektorii zadanej.
- Algorytm kompensacji nie może zaburzać cech sterowania impedancyjnego pozwalających na uginanie się robota w momencie kolizji.

4.2 Rozwiązań oparte na regulatorze PID

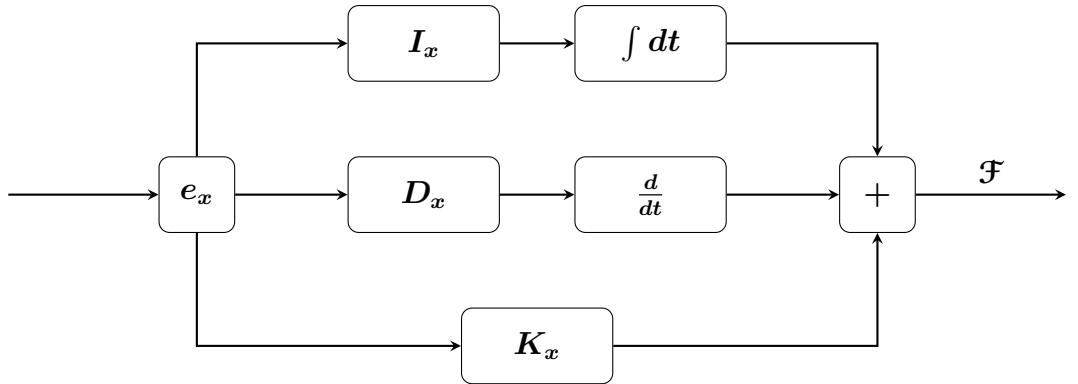
Opadanie ramion można interpretować jako błąd pomiędzy pozycją zadaną przez interpolator i pozycją zadaną czyli uchyb. Problem uchybu rozwiązuje regulator PID opisany w sekcji 2.2. Jego zastosowanie rodzi ryzyko, że ramię robota usztywnią się i będą powodować uszkodzenia w trakcie kolizji. Stąd wniosek aby połączyć prawo sterowania impedancyjnego i regulator PID w nowe prawo sterowania [10, 21]. Po połączeniu obydwu praw sterowania i zastosowaniu ograniczenia całkowania ramiona nie powinny być nadmiernie sztywne.

Niektóre człony są takie same w zaprezentowanych prawach sterowania. W impedancyjnym prawie sterowania nie ma członu całkującego i został on skopiowany z algorytmu PID. W rezultacie nowe prawo sterowania jest postaci (rys. 4.3):

$$\mathcal{F} = \mathbf{K}_x e_x + \mathbf{D}_x \dot{e}_x + \int_0^t \mathbf{I}_x e_x dt \quad (4.1)$$

gdzie:

- \mathcal{F} to wektor sił wynikowych
- \mathbf{K}_x to diagonalna macierz sprężystości



Rysunek 4.3: Schemat nowego prawa sterowania. W pierwszej fazie wyliczany jest algorytm z członem całkującym a następnie transformowany do przestrzeni stawów przez jacobian.

- D_x to diagonalna macierz sztywności
- I_x to diagonalna macierz członu całkującego
- e_x to wektor uchybu

Parametry członu całkującego powinny mieć na tyle małe wartości, żeby nie zdominowały początkowego prawa sterowania impedancyjnego a jedynie poprawiały jego działanie. Ważne jest też zastosowanie ograniczenia całkowania. Po wyliczeniu prawa sterowania należy zastosować przekształcenie obliczające momenty obrotowe, które mają być zadane na silniki robota analogicznie do procedury pokazanej w sekcji 2.1.3.

4.3 Rozwiązańe oparte na estymacji modelu robota

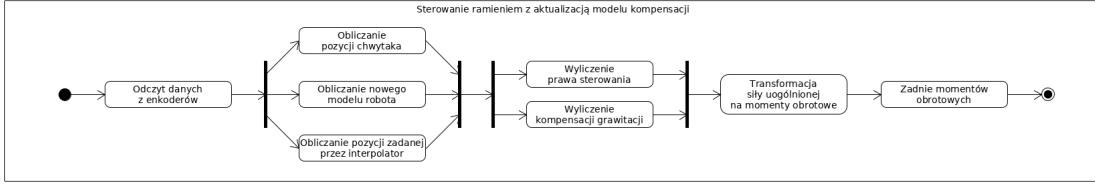
Problem można też rozwiązać przez estymację nowego modelu wykorzystywanego do kompensacji grawitacji postaci takiej jak w sekcji 2.3 ale rozszerzonego o zastosowane prawo sterowania [13,14] (rys. 4.4). Równanie siły uogólnionej dla chwili i jest wtedy postaci:

$$\mathcal{F}_{mi} = \mathbf{K}_x e_x + \mathbf{D}_x \dot{e}_x + \boldsymbol{\Lambda}(\mathbf{q}) \ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\gamma}(\mathbf{q}) + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) \quad (4.2)$$

gdzie:

- $\boldsymbol{\Lambda}$ to dodatnio określona macierz inercji w przestrzeni zadań
- $\boldsymbol{\mu}$ to macierz sił Coriolisa i sił odśrodkowych
- $\boldsymbol{\gamma}$ to wektor sił grawitacji
- $\boldsymbol{\eta}$ to macierz sił tarcia oraz nieuwzględnionych sił

ROZDZIAŁ 4. SPECYFIKACJA SYSTMU KOMPENSACJI GRAWITACJI



Rysunek 4.4: Diagram aktywności pokazujący przebieg sterowania ramieniem robota z aktualizacją modelu.

- \mathbf{q} to wektor położen stawów w przestrzeni konfiguracyjnej
- \mathbf{x} to wektor położen końcówki w przestrzeni zadań
- \mathbf{K}_x to diagonalna macierz sprężystości
- \mathbf{D}_x to diagonalna macierz sztywności
- \mathbf{e}_x to wektor uchybu

Czujnik FTS dla każdej chwili i pozwala na odczytanie rzeczywistego wektora siły uogólnionej w końcówce \mathbf{F}_i . Dzięki temu możemy uzyskać różnicę pomiędzy dwoma tymi wartościami:

$$\mathbf{F}_{ei} = \mathbf{F}_i - \mathbf{F}_{mi} \quad (4.3)$$

a następnie zastosować optymalizację, której parametrami są macierze stosowane do wyliczania modelu. Problem optymalizacji powinien być oparty o minimalizację różnicy dla n próbek:

$$\min_{\Lambda(\mathbf{q})\ddot{\mathbf{x}}, \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}), \boldsymbol{\gamma}(\mathbf{q}), \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}})} \sum_{i=1}^n \|\mathbf{F}_{ei}\| \quad (4.4)$$

W trakcie praktycznych obliczeń można stosować pewne uproszczenia modelu. Przy założeniu braku ruchu pomiędzy narzędziem a chwytkiem możemy potraktować narzędzie jako część chwytaka. W takiej sytuacji dodanie narzędzia do chwytaka powoduje zmianę parametrów środka ciężkości masy oraz ostatniego członu i wprowadza zmiany tylko z tym związane. Można też przyjąć pewne uproszczenia wynikające z faktu, że chwytny przedmiot ma stosunkowo małą masę w porównaniu z masą całego ramienia robotycznego. Estymowany model powinien być aktualizowany na bieżąco w trakcie pracy robota i stosowany w celu kompensacji grawitacji zamiast pierwotnego modelu.

Optymalizacja nieliniowa jest rozwiązywana na tyle wolno, że model nie może być aktualizowany tak często jak cały system sterowania robota. Rozwiązanie jest też dużo bardziej skomplikowane w implementacji niż opisane w sekcji 4.2.

4.4 Wybór rozwiązania

Implementacja rozwiązania z sekcji 4.3 rodzi sporo problemów. W praktyce trudne jest zbudowanie odpowiedniego zadania optymalizacji, które zagwarantuje na tyle dobry model, że grawitacja będzie odpowiednio kompensowana. Dodatkowo optymalizacja obliczana jest na tyle wolno, że nie ma możliwości umieszczenia estymatora tego modelu jako jeden z komponentów podsystemu sterowania *velma_core_cs*. Komponent musiałby zostać umieszczony w nowym agencie, który pracuje z mniejszą niż cały system częstotliwością i komunikuje się z systemem robota za pośrednictwem interfejsu ROSa. W początkowej fazie, gdy przedmiot zostanie chwycony najczęściej ramiona robota zostają nieruchome, bo grawitacja chwyconego narzędzia nie jest kompensowana. Wtedy estymacja modelu jest wyjątkowo niedokładna. W początkowej fazie należałoby stosować uproszczony model tylko po to żeby robot podniósł przedmiot. Dopiero po kilku ruchach, gdy zadanie optymalizacji zostanie poprawnie rozwiązane, można wprowadzić właściwy model z użyciem narzędzi logiki rozmytej.

Wspomniane wady były podstawą decyzji o implementacji rozwiązania z sekcji 4.2. Główną wadą jest niemożność wyekstrahowania danych dotyczących narzędzia które chwyta robot. Kolejną wadą może być lekkie usztywnienie stawów. Nie powinno mieć dużego znaczenia z dwóch powodów. W sterowaniu impedancyjnym w przestrzeni operacyjnej wirtualna sprężyna jest zawieszona pomiędzy chwytkiem a punktem zadanym, a nie w konkretnych stawach robota. Do tego zakładamy niskie współczynniki członu całkującego.

ROZDZIAŁ 4. SPECYFIKACJA SYSTMU KOMPENSACJI GRAWITACJI

Rozdział 5

Implementacja systemu

W celu rozwiązania problemu opisanego w pracy dokonano szeregu modyfikacji w systemie oprogramowania robota Velma. Zmodyfikowano prawo sterowania impedancyjnego, tak aby dodać człon całkujący. Dodano możliwość konfiguracji nowych parametrów prawa sterowania związanych z członem całkującym. Stworzono środowisko pozwalające na śledzenie oraz wizualizację wymaganych trajektorii chwytaka.

5.1 Prawo sterowania

Modyfikacja prawa sterowania została wykonana w komponentie *cart_imp* znajdującym się w podsystemie *velma_core.cs*. Ponieważ mamy do czynienia z systemem z dyskretnym czasem nie jest możliwe bezpośrednie zaimplementowanie wzoru 4.1. Człony związane z impedancyjnym prawem sterowania były już zaimplementowane. Do implementacji członu całkującego skorzystano z dyskretnej wersji regułatora PID [24] dla przypadku jednowymiarowego w chwili k jest postaci:

$$u(k) = u(k-1) + P[(1 + \frac{\Delta}{T_i} + \frac{T_d}{\Delta})e(k) + (-1 - \frac{2T_d}{\Delta})e(k-1) + \frac{T_d}{\Delta}e(k-2)] \quad (5.1)$$

przy założeniach:

$$T_i = P/I \quad (5.2)$$

$$T_d = D/P \quad (5.3)$$

gdzie:

- $u(k)$ to sterowanie w chwili k .
- $e(k)$ to uchyb w chwili k .
- Δ to czas jednego kroku symulacji, dla systemu robota Velma ma wartość $\frac{1}{500}$ s.

- P to parametr członu proporcjonalnego.
- I to parametr członu całkującego.
- D to parametr członu różniczkującego.

Dla naszych potrzeb istotny jest tylko człon całkujący więc równanie 5.1 upraszcza się do postaci:

$$u(k) = u(k-1) + e(k) + \frac{\Delta e(k)}{T_i} - e(k-1) - \frac{2T_d e(k-1)}{\Delta} \quad (5.4)$$

Następnie wzór 5.4 jest rozszerzany do postaci sześciowymiarowego wektora poprzez zastosowanie w każdym wierszu tego wektora. Taki wektor można łatwo dodać do istniejącego prawa sterowania w przestrzeni operacyjnej robota.

5.2 Parametry członu całkującego

W celu umożliwienia konfiguracji parametrów członu całkującego został zmodyfikowany interfejs *VelmaInterface*. Do ruchu ramieniem w przestrzeni operacyjnej służy funkcja *moveCartImpRight()*. Jej nowy nagłówek został rozszerzony o parametr *gcomp* który ma być listą opisującą przekątną macierzy członu całkowania. Opisane parametry są przekazywane przez system akcji ROSa do agenta *velma_task_cs_ros_interface*. Odpowiedzialny za to jest komponent *CartImpActionRight*, który został odpowiednio zmodyfikowany w ramach pracy. Komponent przyjmuje na zmodyfikowaną w ramach pracy wiadomość *RightImpAction*.

5.3 Rejestracja trajektorii

Ocena jakości algorytmu sterowania następuje poprzez porównanie odpowiednich trajektorii. W celu wydobycia trajektorii określonych przez wektory siły uogólnionej zmodyfikowano komponent *TfPublisher* z podsystemu *velma_task_cs_ros_interface* w taki sposób, żeby udostępnił pozycję zadaną, generowaną przez interpolator trajektorii, w odpowiednim temacie ROS o nazwie *right_arm_cmd*. W podobny sposób udostępniona jest pozycja osiągnięta w danym momencie przez robota w temacie *right_arm_tool*. Za pomocą pomocniczego programu *writer.py* pisанego w Pythonie pozycje są odbierane w trakcie działania systemu i zapisywane wraz ze znacznikiem czasu.

5.4 Generowanie wykresów

Zapisane pliki z danymi są przetwarzane przez napisany program `plotter.py` zaimplementowany w Pythonie. Jego zadaniem jest generowanie obrazów pozwalających na porównanie zapisanych w plikach trajektorii oraz wyliczeń związanych z oceną jakości trajektorii na podstawie metryki APE. Program tworzy przebiegi w zależności od czasu dla konkretnych osi oraz kątów wielu trajektorii. Ma też możliwość generowania dwuwymiarowych rzutów metryki APE przez wywołanie zewnętrznego skryptu [9]. Program `plotter.py` może porównywać dowolną liczbę trajektorii. Pliki które podlegają obliczeniom są podawane w argumentach programu.

5.5 Opis świata testowego

Stworzono nowe pliki `can_world.world` oraz `drill_world.world` które są opisem świata używanym przez symulator świata Gazebo. Pliki są strukturami typu XML które zawierają opis i umiejscowienie w świecie konkretnych przedmiotów. Obydwa pliki opisują świat z robotem Velma oraz znajdującym się obok niego stole. Na stole położone są przedmioty testowe czyli puszka oraz wiertarka.

5.6 Algorytm eksperymentu

Do testowania modyfikacji systemu stworzono program `test_script.py`, który wykorzystuje interfejs `VelmaInterface` i określa ruchy, które ma wykonać robot w trakcie testów. Program wykonuje poniższe czynności:

1. Inicjalizacja systemu poprzez inicjalizację obiektów potrzebnych do komunikacji z ROSem.
2. Ułożenie robota w odgórnie zdefiniowanej pozycji startowej.
3. Ruch ramienia do pozycji w której może chwycić przedmiot.
4. Chwycenie przedmiotu poprzez zaciśnięcie chwytyka.
5. Ruch ramienia do pozycji rozpoczęcia następnego testu.
6. Wykonanie chwytakiem ruchu ósemkowego.
7. Wykonanie chwytakiem ruchu do góry i spowrotem.
8. Wykonanie chwytakiem ruchu w bok i spowrotem.
9. Wykonanie chwytakiem ruchu do przodu i spowrotem.
10. Wykonanie chwytakiem obrotu.

ROZDZIAŁ 5. IMPLEMENTACJA SYSTEMU

Rozdział 6

Działanie systemu

W celu zaprezentowania działania nowego algorytmu kompensacji grawitacji uruchomiono program testowy w trybie symulacji. Wszystkie eksperymenty wykazywały, że przy odpowiednio długim czasie eksperymentu uchyb końcówki dąży do zera. Do obliczeń przyjęto krótsze obserwacje przebiegów, w celu pokazania zachowania systemu w realnych warunkach. Eksperymenty opisane są w bazowym układzie odniesienia (rys. 3.1(b)). Ocena jakości sterowania następuje poprzez porównanie przebiegów trajektorii realizowanych w opisanych wcześniej wersjach.

W pracy zamieszczono wyniki eksperymentów o tych samych trajektoriach zadanego w czterech wersjach:

- bez algorytmu kompensacji, bez narzędzia
- z algorymem kompensacji, bez narzędzia
- z algorymem kompensacji, z chwyconą puszką o kształcie walca, masie 1 kg i jednolitym rozkładzie mas
- z algorymem kompensacji, z chwyconą wiertarką o nierównomiernym kształcie i masie 1,5 kg

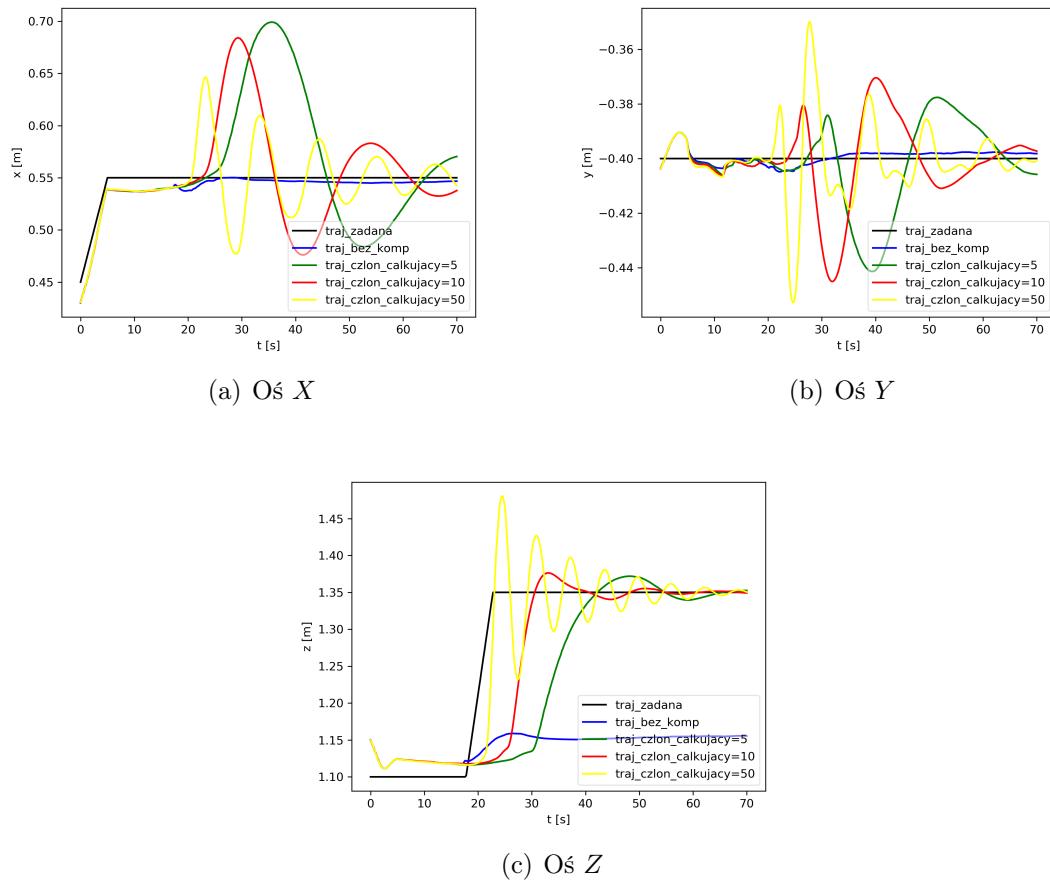
W pracy nie zamieszczono wyników działania algorytmu sterowania bez kompensacji i z chwyconymi przedmiotami. Takie trajektorie nie były w żadnym stopniu wykonywane.

W trakcie wszystkich eksperymentów wszystkie parametry algorytmu sterowania były takie same. Macierze sprężystości i sztywności miały takie same niskie wartości. Dobór parametrów członu całkującego nastąpił eksperymentalnie przy założeniu, że wszystkie parametry macierzy diagonalnej członu będą miały te same wartości. W trakcie doboru wartości przeprowadzono eksperyment podnoszenia przedmiotu (rys. 6.1, 6.2). Poszukiwana wartość parametru nie powinna spowodować by człon całkujący zdominował inne człony oraz aby nie wprowadził dużych oscylacji zarówno. Nie

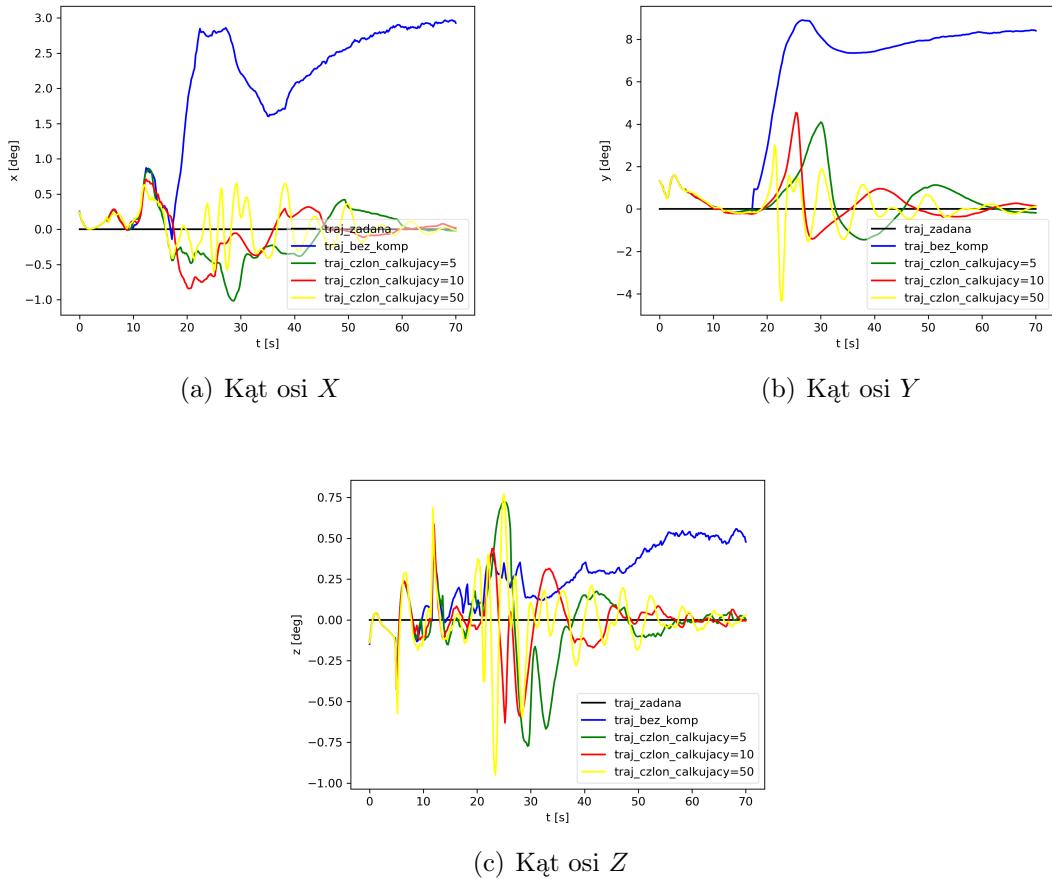
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

może mieć za dużej wartości, gdyż wtedy ramiona robota usztywnią się w trakcie manipulacji. Musi być na tyle dużej wartości by nastąpiła kompensacja grawitacji testowego przedmiotu o masie 1 kg w rozsądnym czasie.

Dla parametru o wartości 50 widać wyraźne oscylacje widoczne zarówno w pozycji jak i w obrocie końcówki. Dla parametru o wartości 2 regulacja następuje za wolno. Dla wyłączonej kompensacji grawitacji (parametr równy 0) przedmiot nie jest podnoszony. Finalnie wybrano wartość parametru równą 10 jako pierwszą która gwarantuje kompensację siły grawitacji w rozsądnym czasie.



Rysunek 6.1: Podnoszenie przedmiotu. Porównanie trajektorii z różnymi parametrami członu całkującego.



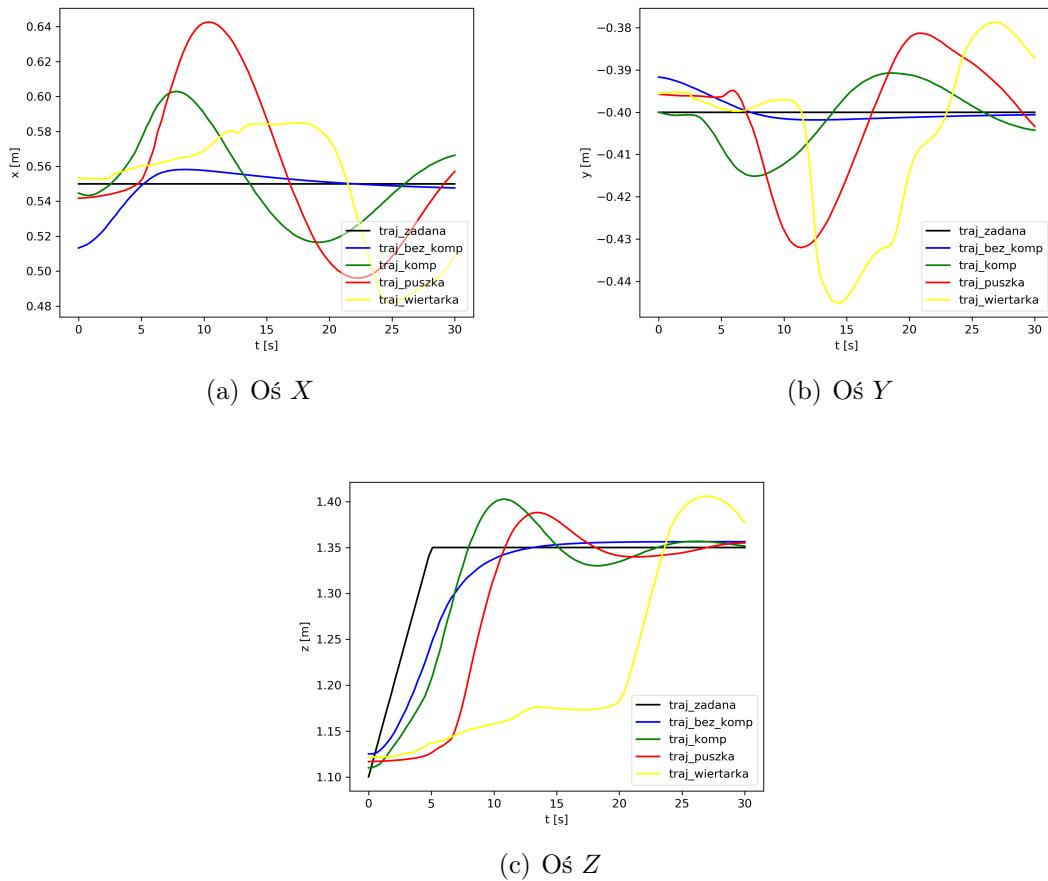
Rysunek 6.2: Podnoszenie przedmiotu. Porównanie parametrów członu całkującego jako kątów w notacji Eulera w zależności od czasu.

6.1 Podnoszenie przedmiotu

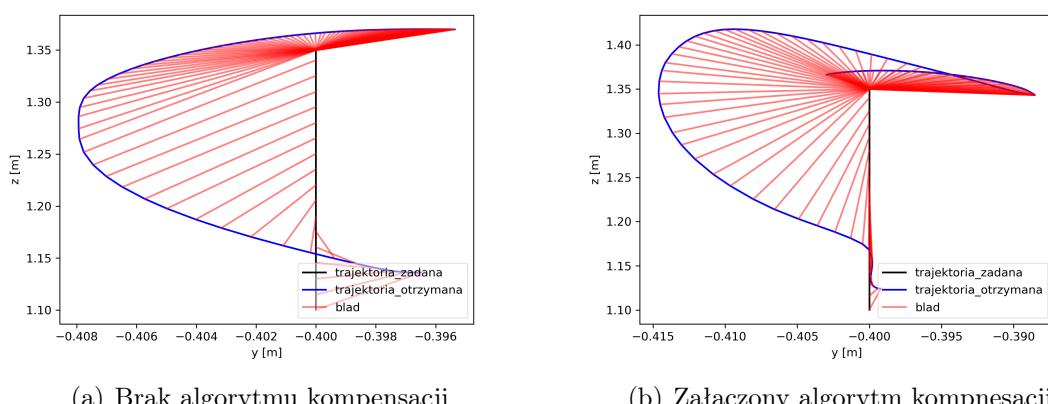
Eksperyment ma przetestować zachowanie algorytmu kompensacji przy podnoszeniu przedmiotu o nieznanych parametrach ruchach (rys. 6.3, 6.5). Ruch widoczny jest głównie w osi Z .

Trajektoria ruchu w rzucie na wprost ruchu została zaprezentowana na rys. 6.4 i 6.6. Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.7,i 6.8.

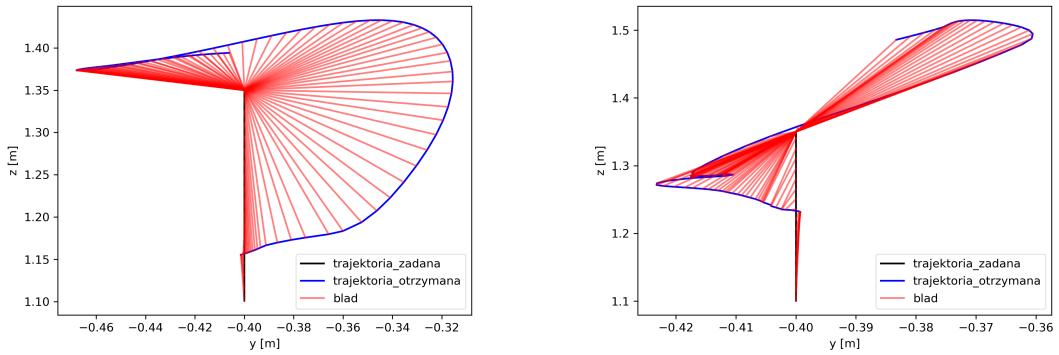
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



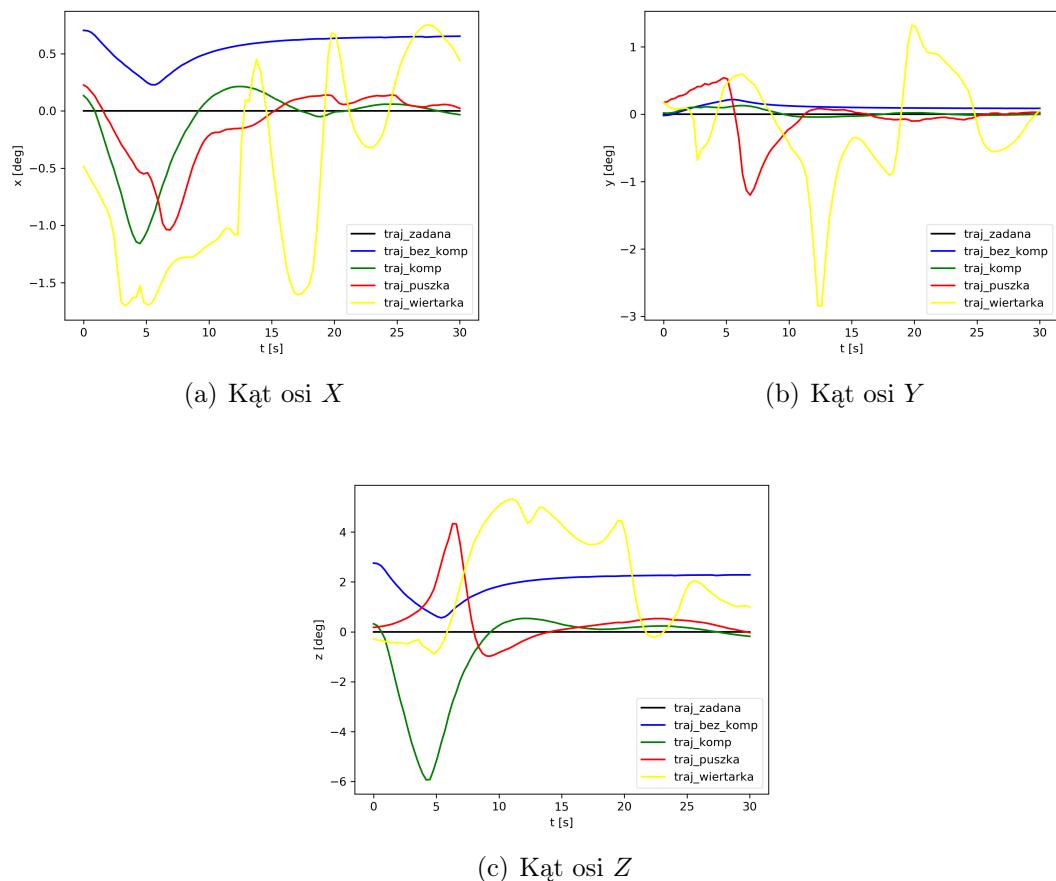
Rysunek 6.3: Podnoszenie przedmiotu. Porównanie trajektorii pozycji w zależności od czasu.



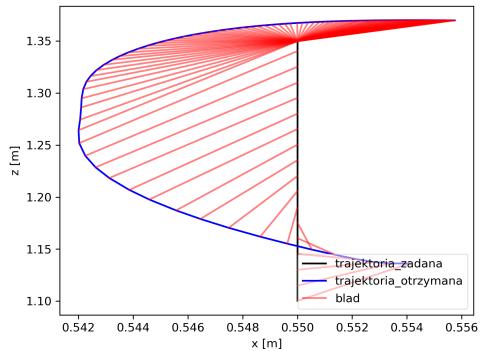
Rysunek 6.4: Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach Y i Z



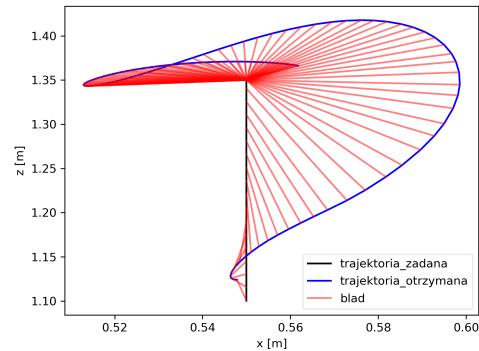
Rysunek 6.6: Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach Y i Z



Rysunek 6.5: Podnoszenie przedmiotu. Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.

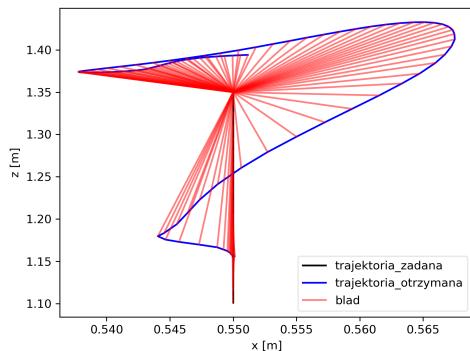


(a) Brak algorytmu kompensacji

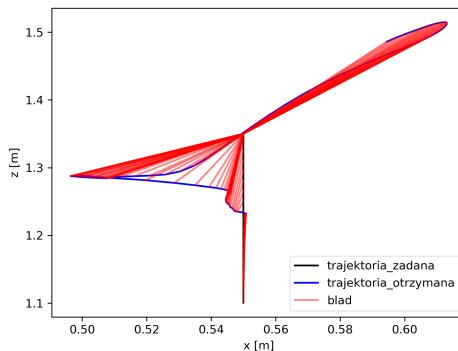


(b) Załączony algorytm kompensacji

Rysunek 6.7: Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach X i Z



(a) Trajektoria z chwyconą puszką



(b) Trajektoria z chwyconą wiertarką

Rysunek 6.8: Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach X i Z

6.2 Ruch ósemkowy

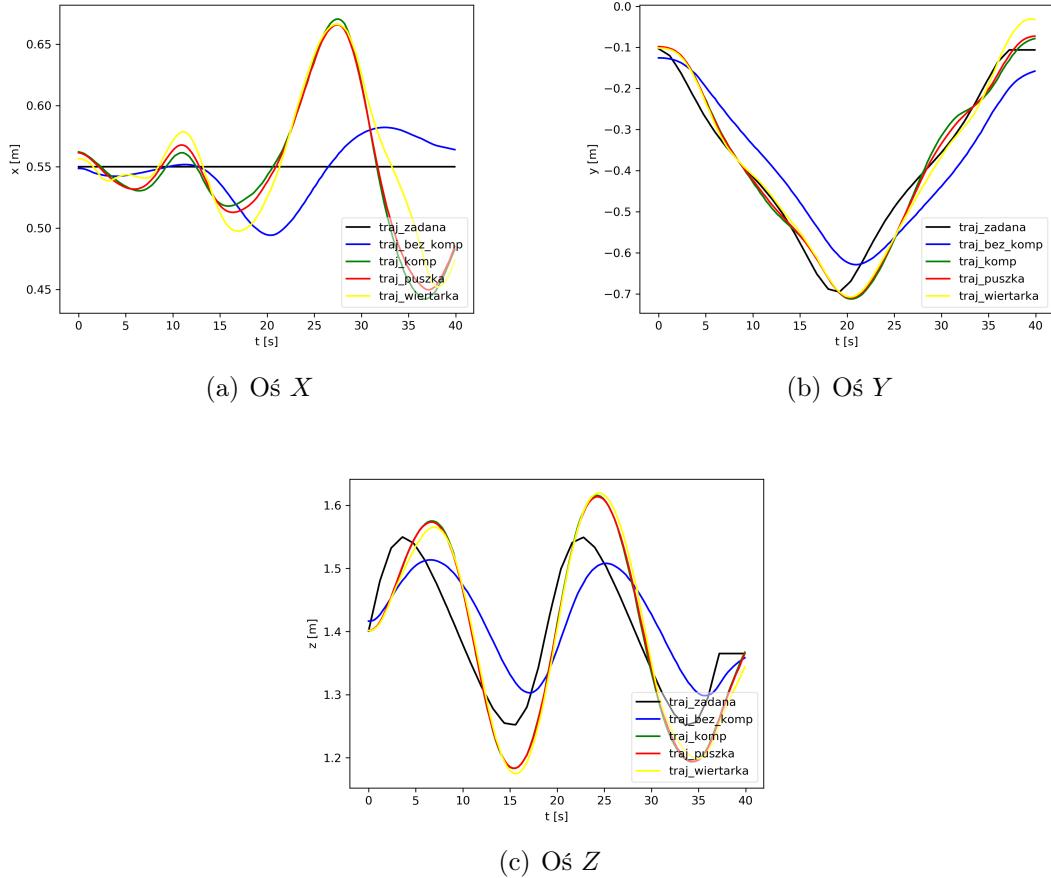
Eksperyment ma przetestować zachowanie algorytmu kompensacji przy skomplikowanych ruchach (rys. 6.9, 6.11). Ruch ósemkowy zadany jest w osi Y oraz Z . Trajektoria jest zadana zgodnie ze wzorem lemniskaty Bernoullego opisanej wzorem:

$$(y^2 + z^2)^2 = 2a^2(y^2 - z^2) \quad (6.1)$$

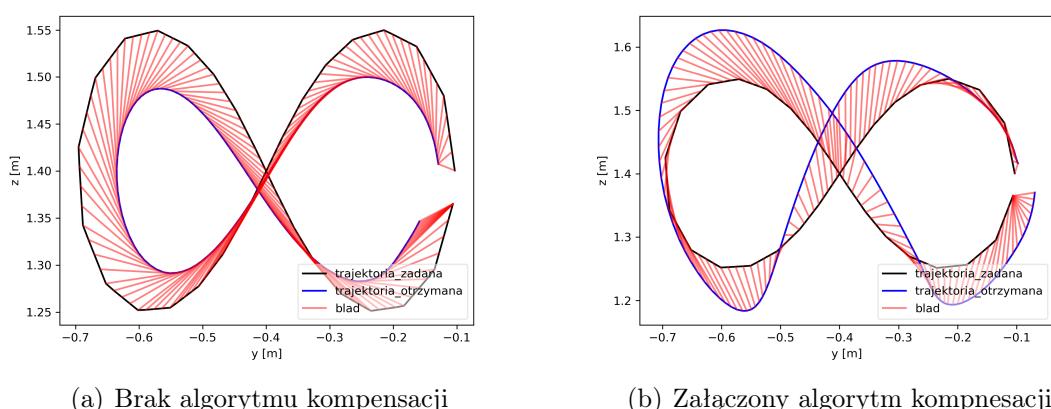
gdzie:

- y oraz z to współrzędne trajektorii
- a to parametr równania

Trajektoria ruchu w rzucie ATE ruchu została zaprezentowana na rys. 6.10 i 6.12. Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.13, 6.14.

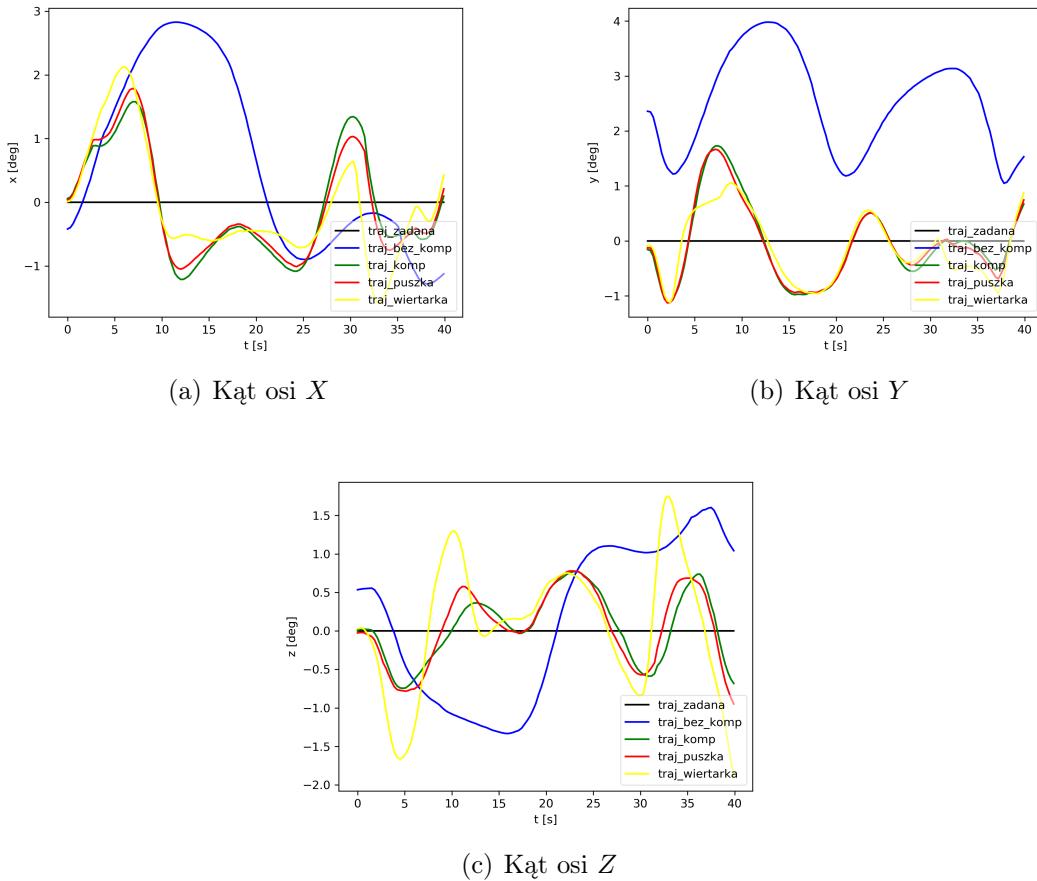


Rysunek 6.9: Ruch ósemkowy. Porównanie trajektorii pozycji w zależności od czasu.

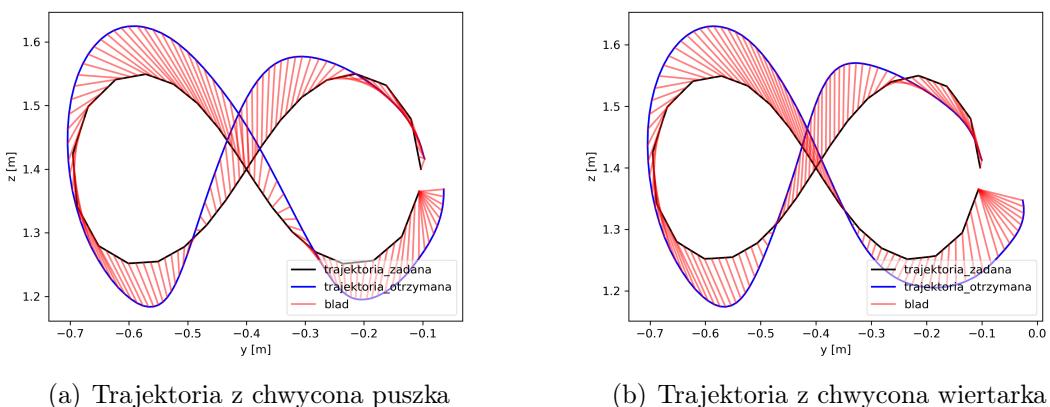


Rysunek 6.10: Porównanie trajektorii chwytaka w osiach Y i Z

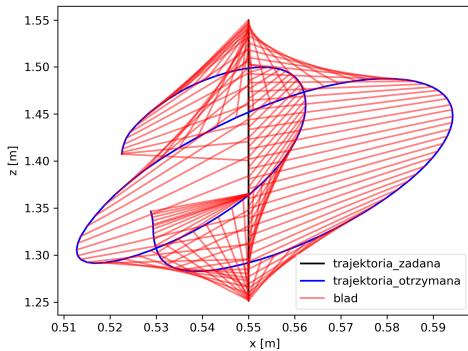
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



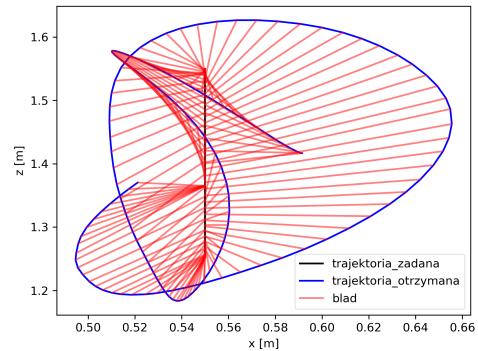
Rysunek 6.11: Ruch ósemkowy. Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.



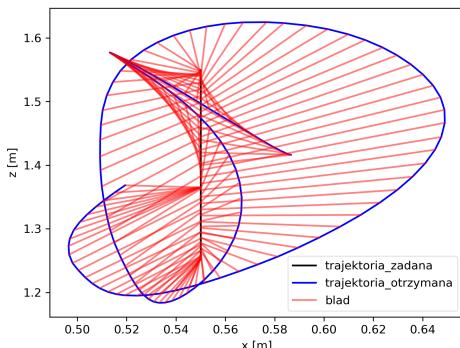
Rysunek 6.12: Ruch ósemkowy. Porównanie trajektorii chwytnika w osiach Y i Z



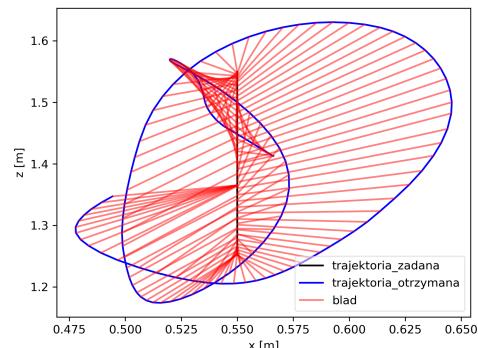
(a) Brak algorytmu kompensacji



(b) Załączony algorytm kompensacji

Rysunek 6.13: Ruch ósemkowy. Porównanie trajektorii chwytaka w osiach X i Z 

(a) Trajektoria z chwyconą puszka



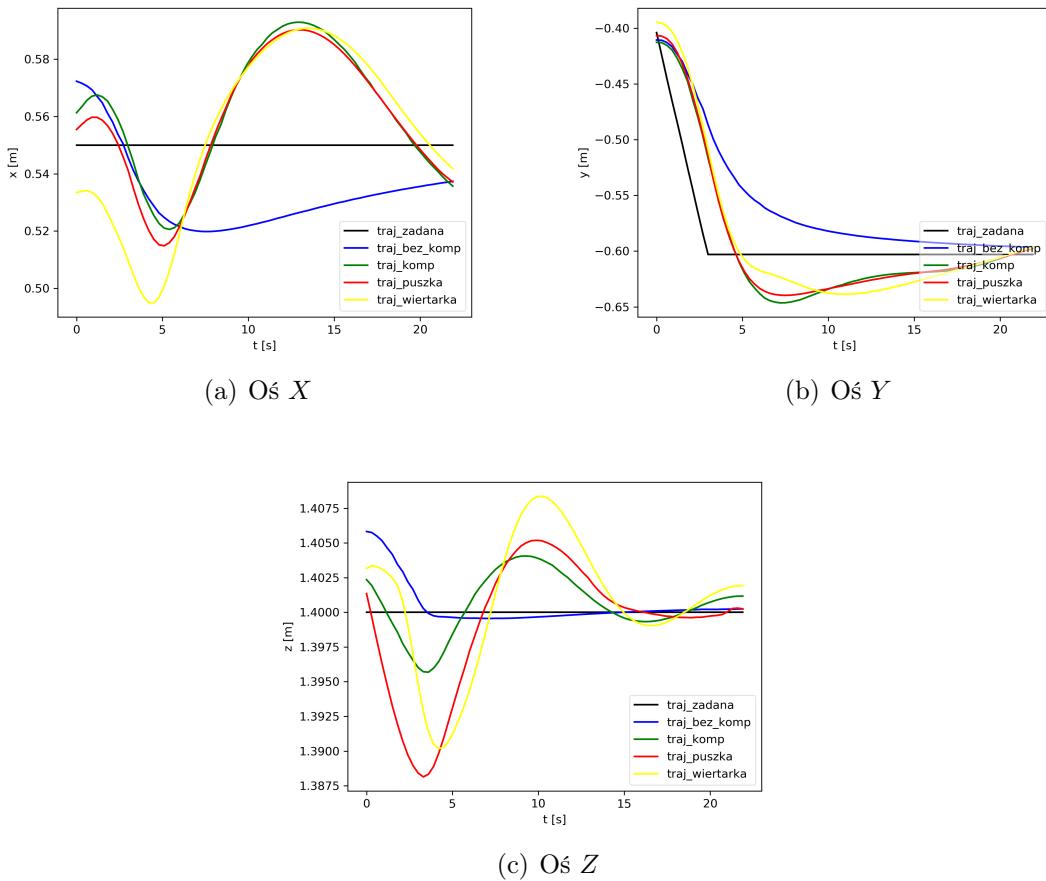
(b) Trajektoria z chwyconą wiertarka

Rysunek 6.14: Ruch ósemkowy. Porównanie trajektorii chwytaka w osiach X i Z

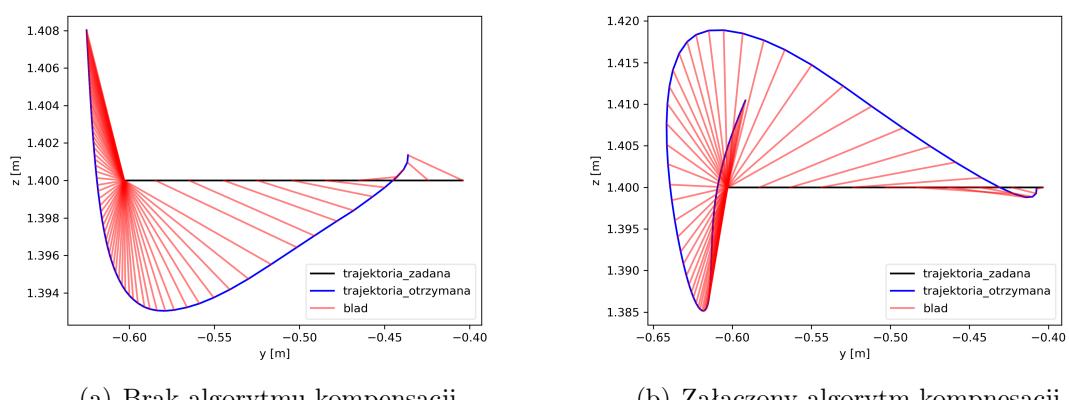
6.3 Ruch w bok

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu końcówki w bok. Trajektoria ruchu w rzucie ATE została zaprezentowana na rys. 6.16 i 6.18.

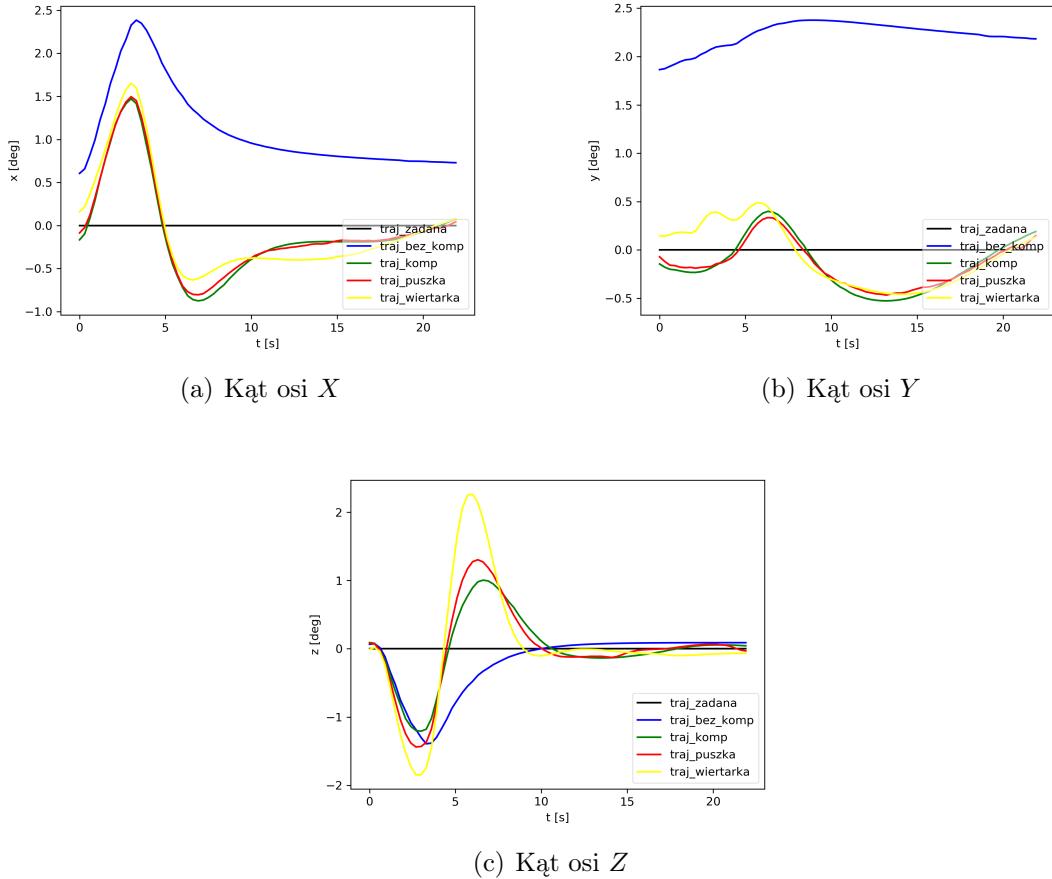
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



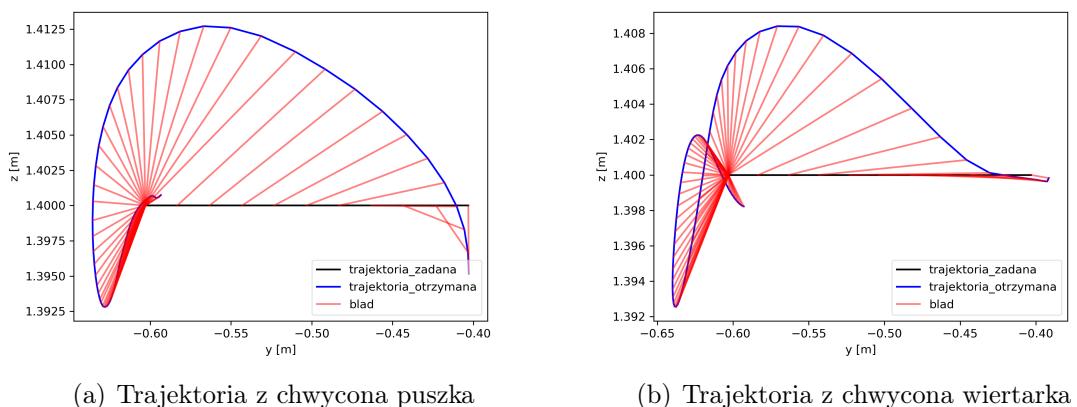
Rysunek 6.15: Ruch w bok. Porównanie trajektorii pozycji w zależności od czasu.



Rysunek 6.16: Ruch w bok. Porównanie trajektorii chwyptaka w osiach Y i Z



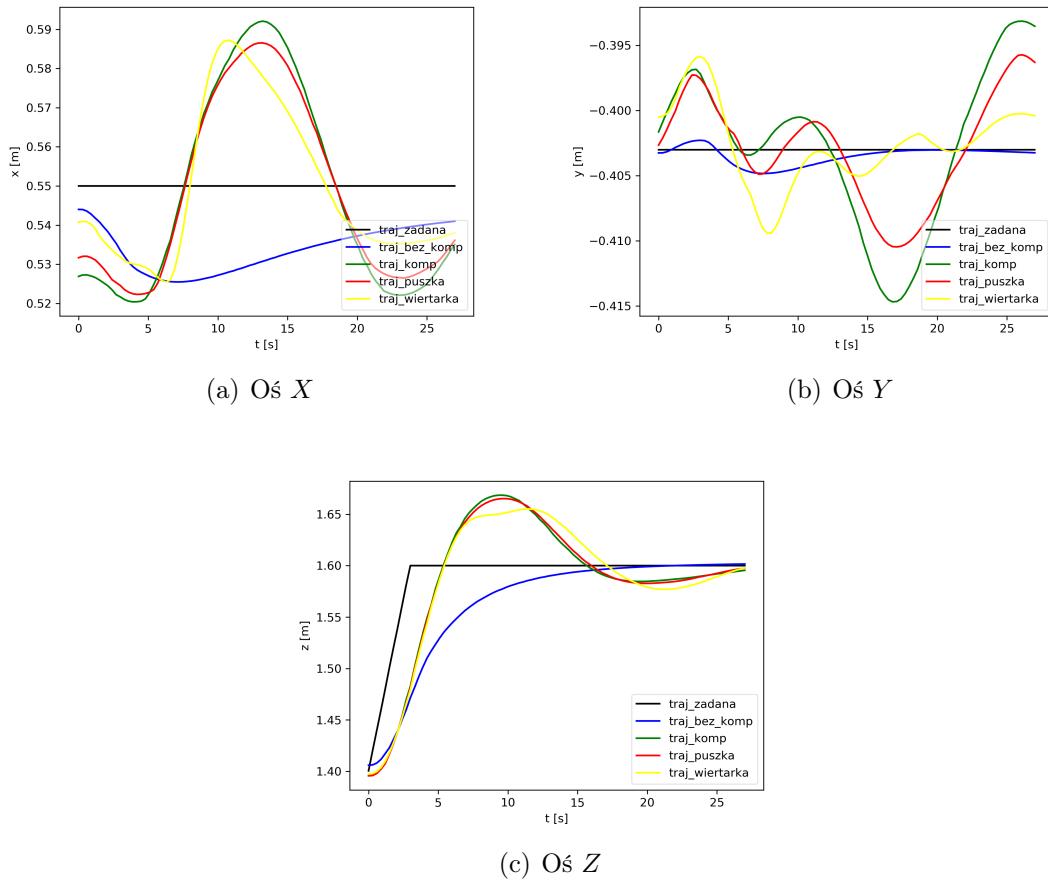
Rysunek 6.17: Ruch w bok. Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.



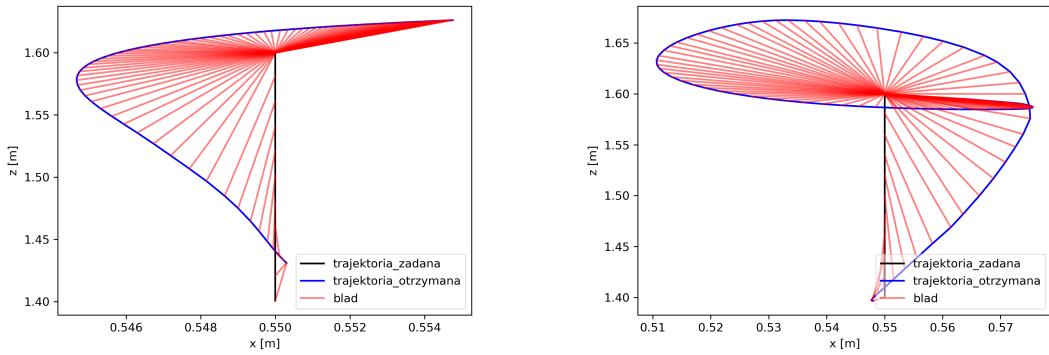
Rysunek 6.18: Ruch w bok. Porównanie trajektorii chwytnika w osiach Y i Z

6.4 Ruch do góry

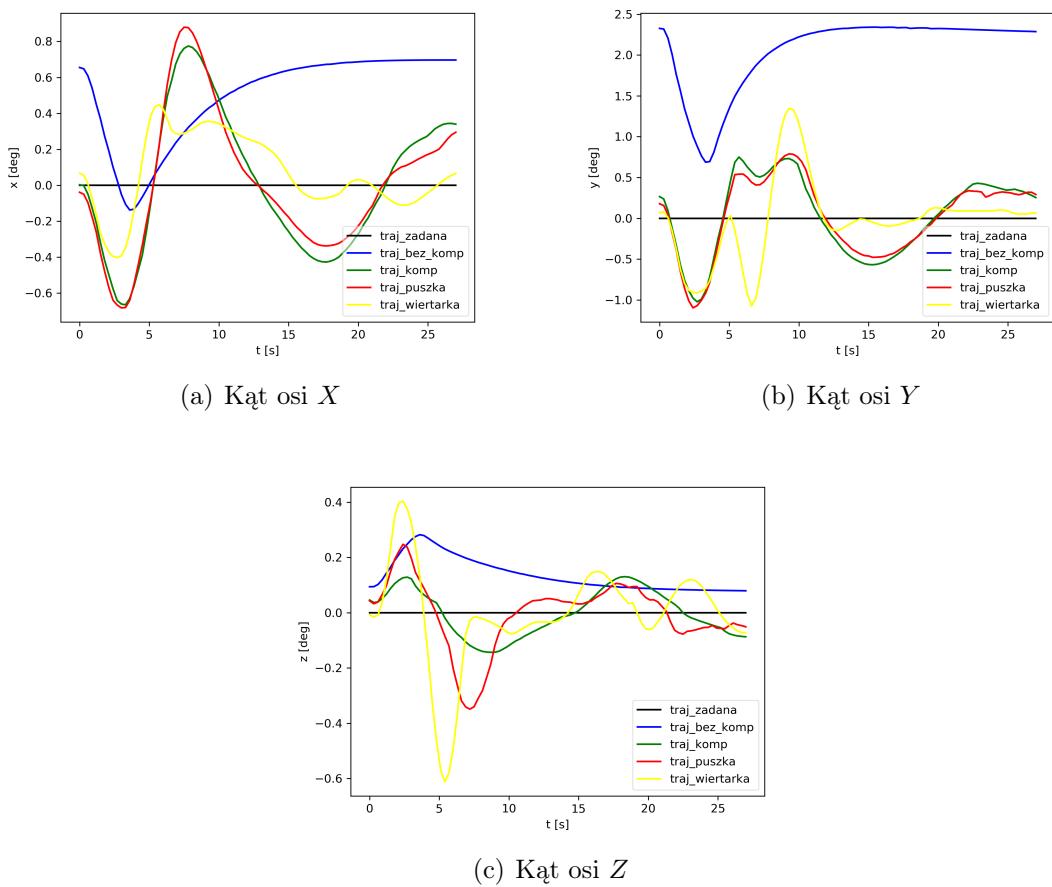
Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu końcówki do góry (rys. 6.19, 6.21). Ruch jest interesujący przez działanie siły grawitacji właśnie w tej osi. Ruch różni się od ruchu podnoszenia przedmiotu, ponieważ grawitacja przedmiotu jest już wstępnie skompensowana. Trajektoria ruchu w rzucie ATE została zaprezentowana na rys. 6.20, 6.22.



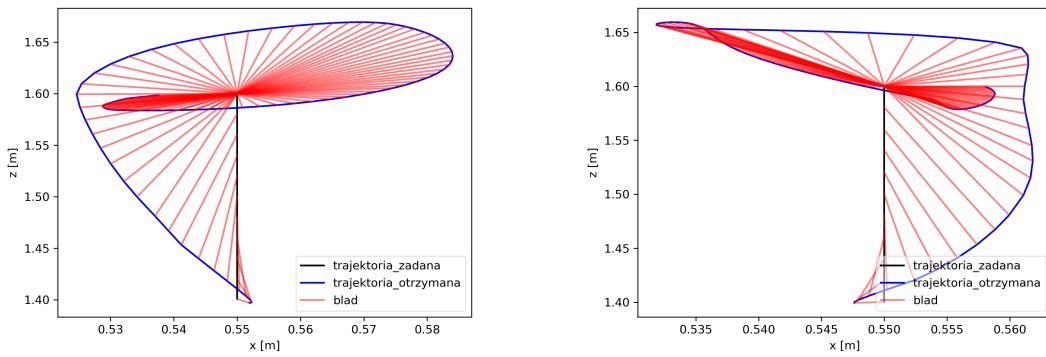
Rysunek 6.19: Ruch do góry. Porównanie trajektorii pozycji w zależności od czasu.



Rysunek 6.20: Ruch do góry. Porównanie trajektorii chwytaka w osiach X i Z



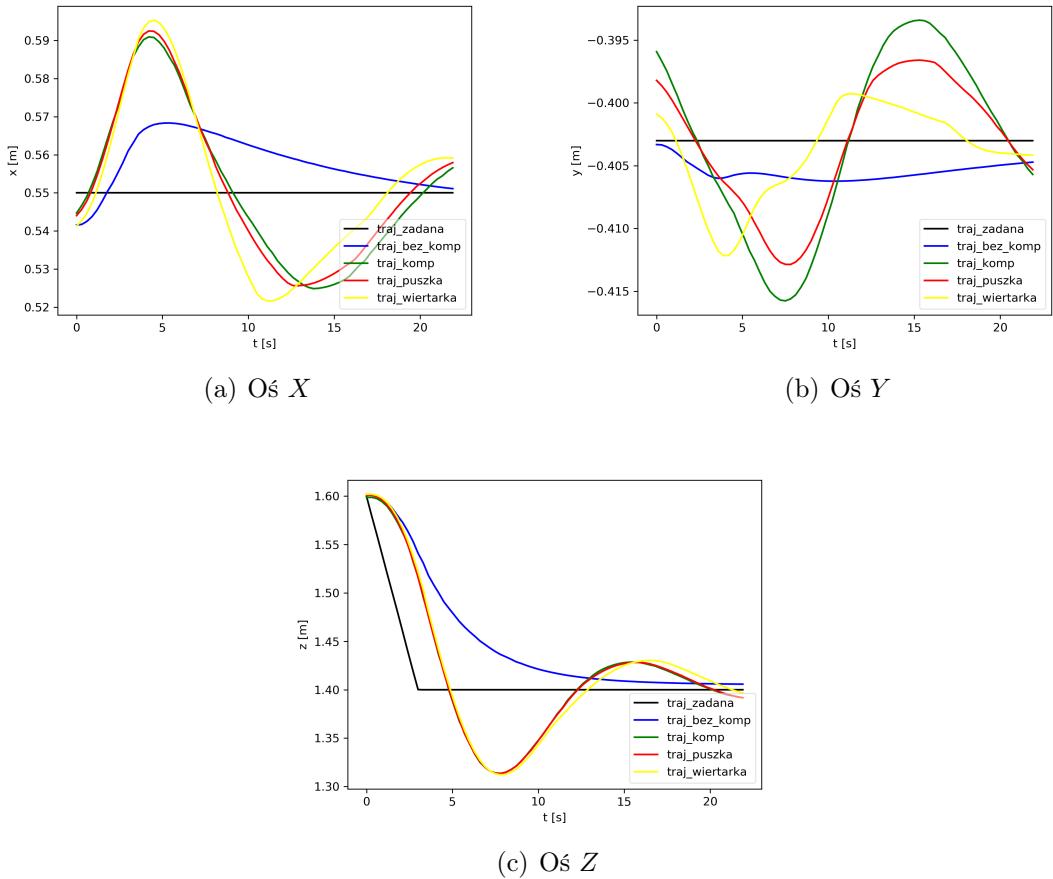
Rysunek 6.21: Ruch do góry. Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.



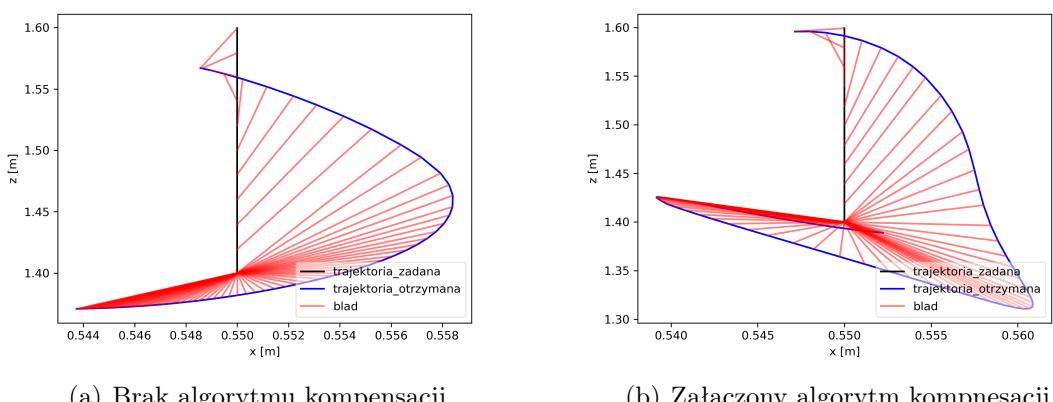
Rysunek 6.22: Ruch do góry. Porównanie trajektorii chwytaka w osiach X i Z

6.5 Ruch do dołu

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu końcówki do dołu (rys. 6.23, 6.25). Trajektoria ruchu pokrywa się z osią w której działa siła grawitacji. Trajektoria ruchu w rzucie ATE została zaprezentowana na rys. 6.24, 6.26.

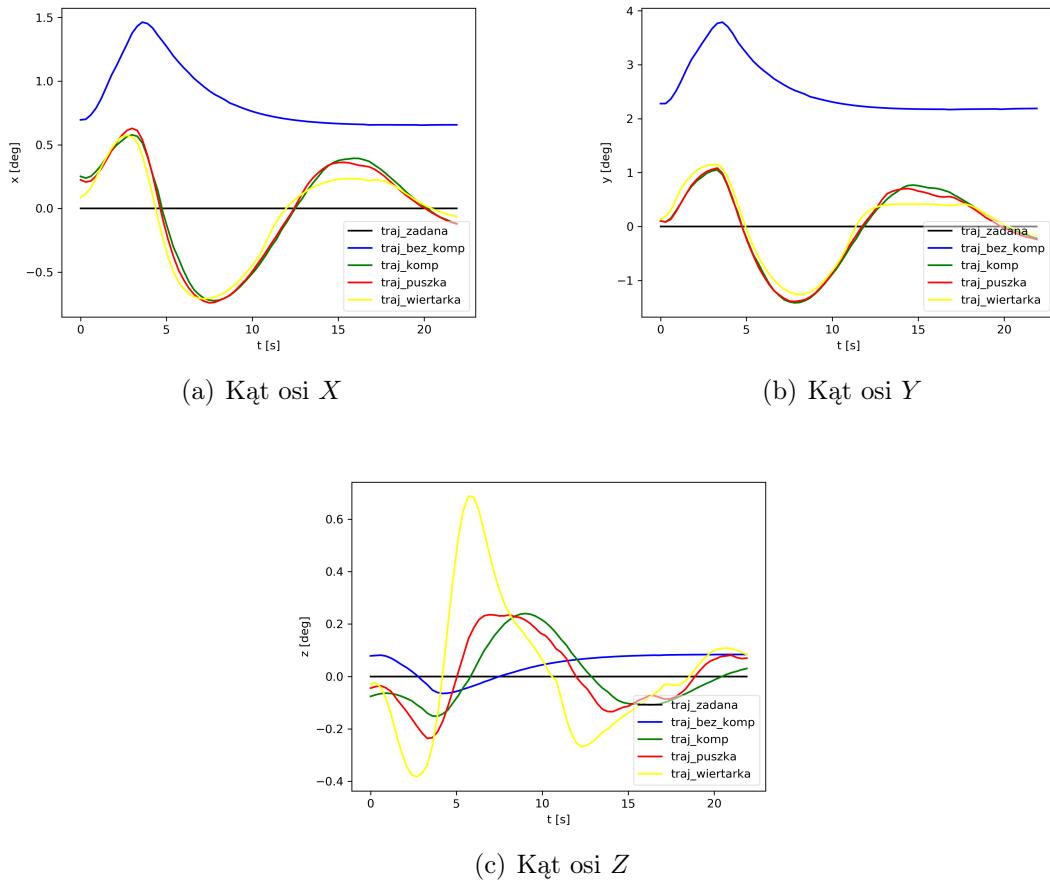


Rysunek 6.23: Ruch do dołu. Porównanie trajektorii pozycji w zależności od czasu.

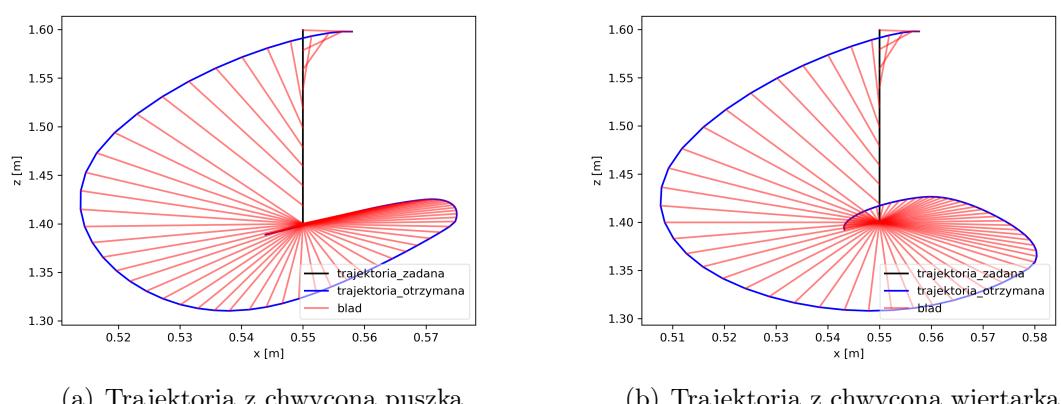


Rysunek 6.24: Ruch do dołu. Porównanie trajektorii chwytaka w osiach X i Z

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



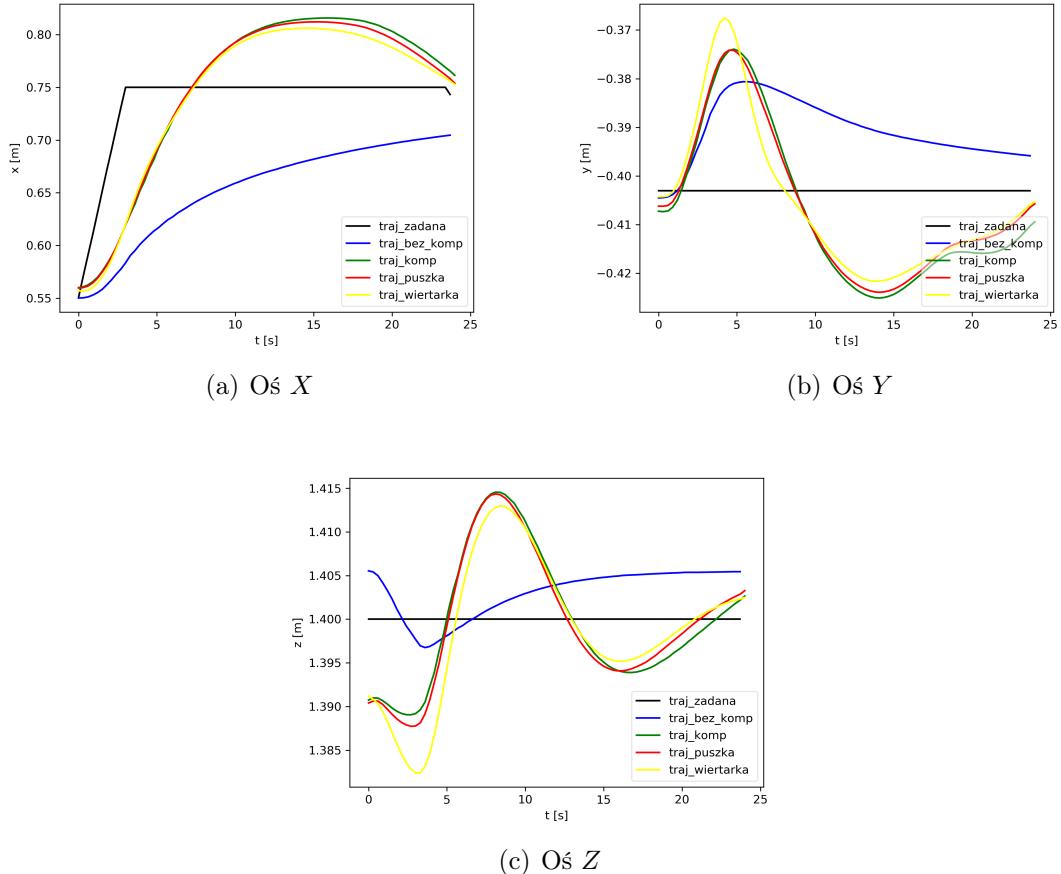
Rysunek 6.25: Ruch do dołu. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.



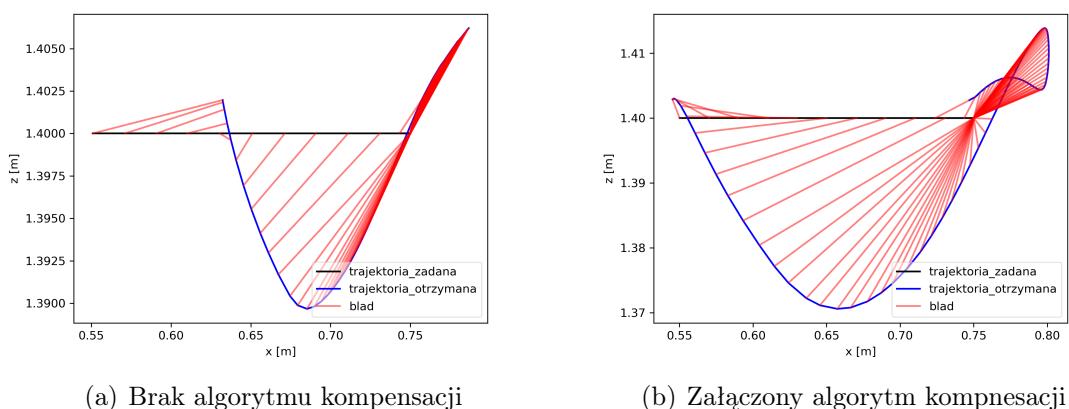
Rysunek 6.26: Porównanie trajektorii chwytyka w osiach X i Z

6.6 Ruch do przodu

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu końcówki w kierunku od robota (rys. 6.27, 6.29). Trajektoria ruchu w rzucie ATE została zaprezentowana na rys. 6.28, 6.30.

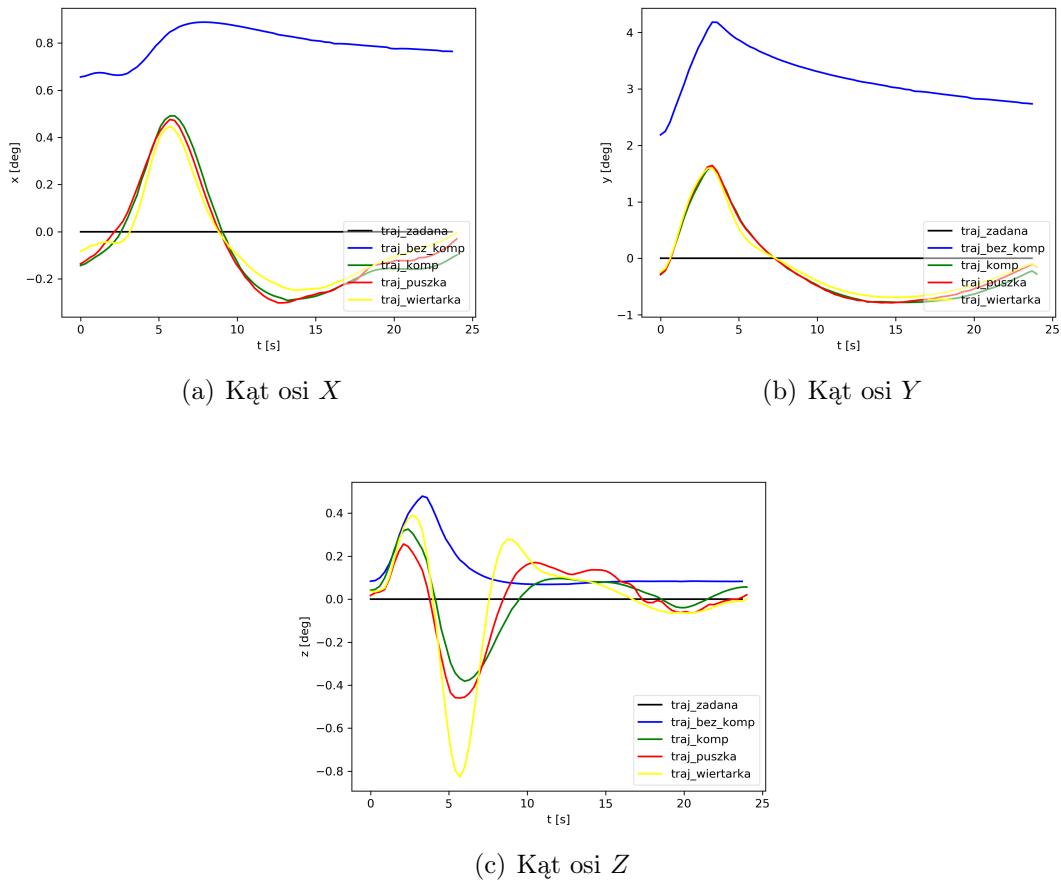


Rysunek 6.27: Ruch do przodu. Porównanie trajektorii pozycji w zależności od czasu.

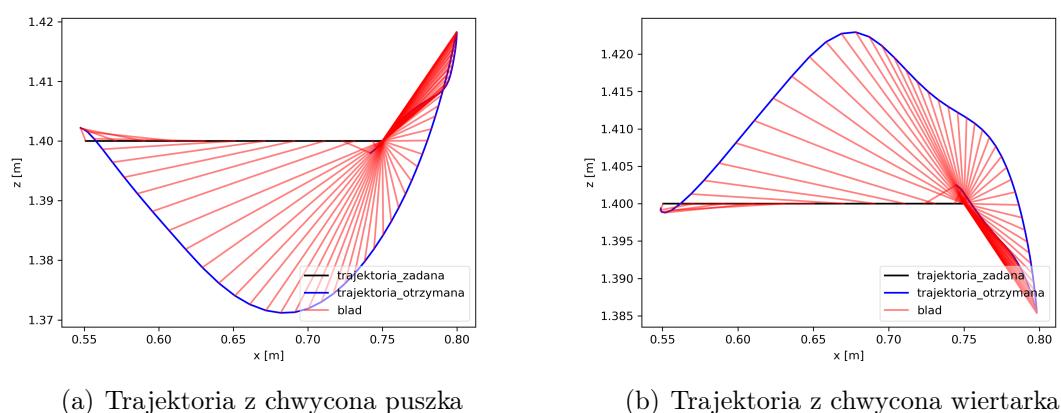


Rysunek 6.28: Ruch do przodu. Porównanie trajektorii chwytki w osiach X i Z

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



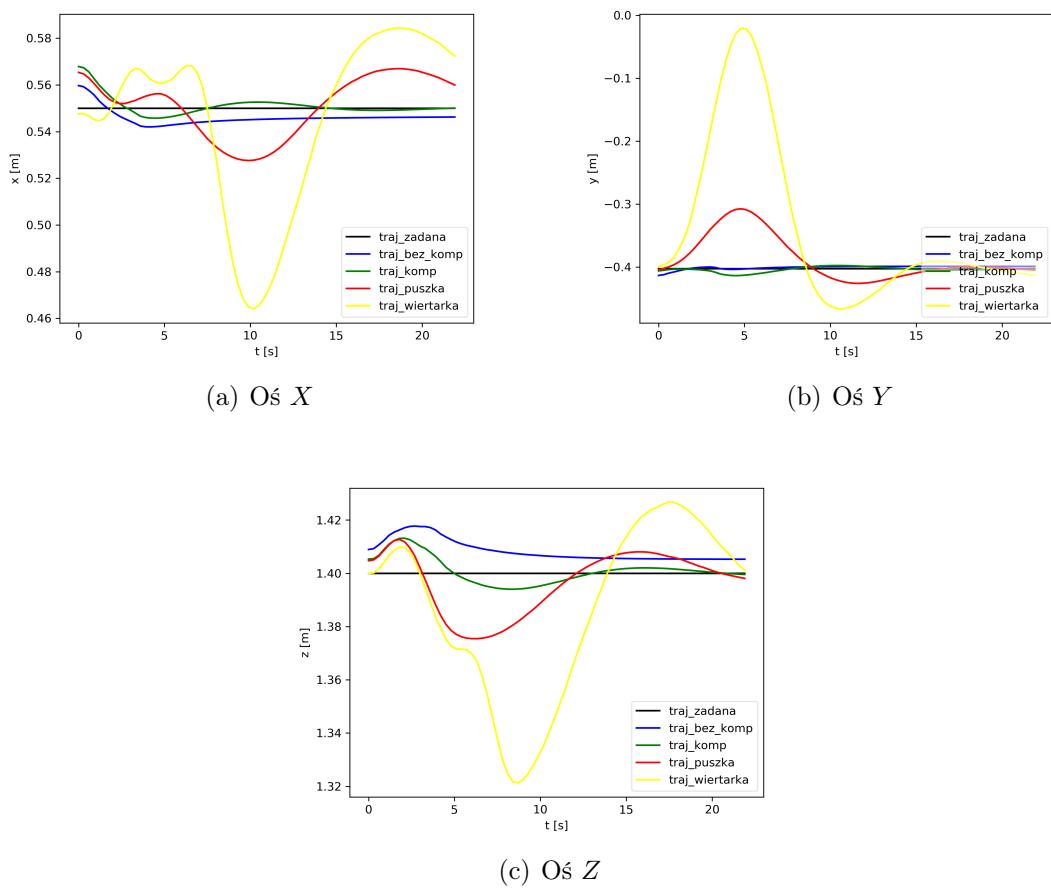
Rysunek 6.29: Ruch do przodu. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.



Rysunek 6.30: Ruch do przodu. Porównanie trajektorii chwytaka w osiach X i Z

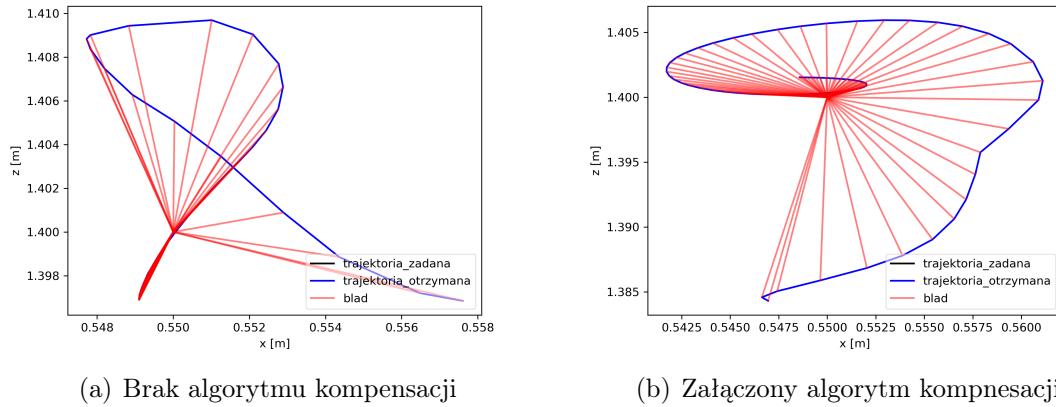
6.7 Obrót końcówki

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy obrocie końcówki. Obserwacje polegają na zmianie położen kątowych końcówki bez zmiany pozycji. Końcówka ma obrócić się o zadany kat we wszystkich osiach (rys. 6.31, 6.33) Trajektoria ruchu (w osiach X oraz Y) została zaprezentowana na rys. 6.32 i 6.34. Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.35, 6.37.

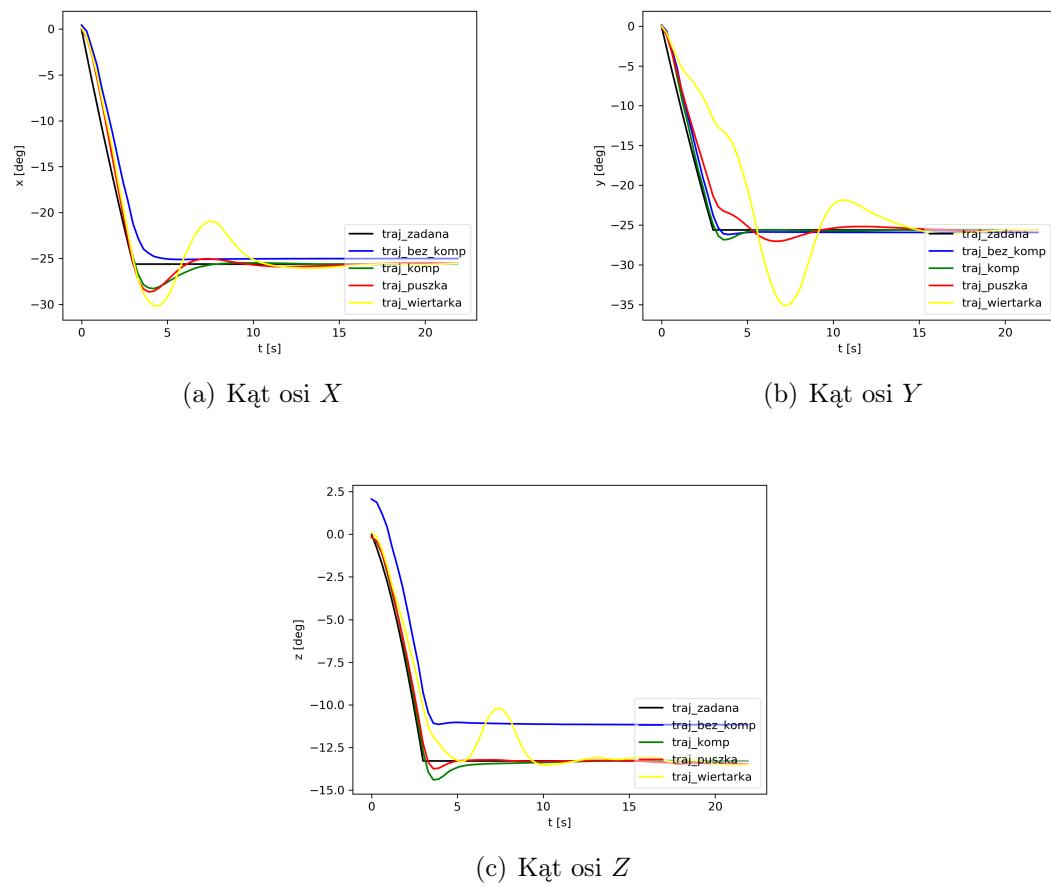


Rysunek 6.31: Porównanie trajektorii pozycji w zależności od czasu.

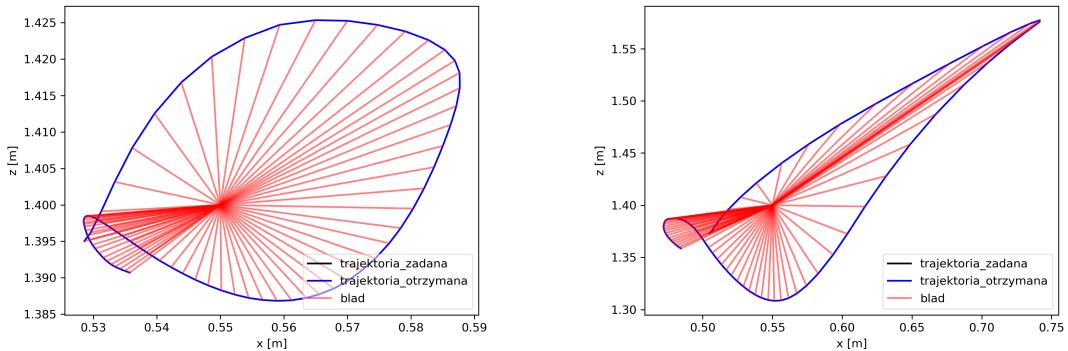
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



Rysunek 6.32: Porównanie trajektorii chwytaka w osiach X i Z



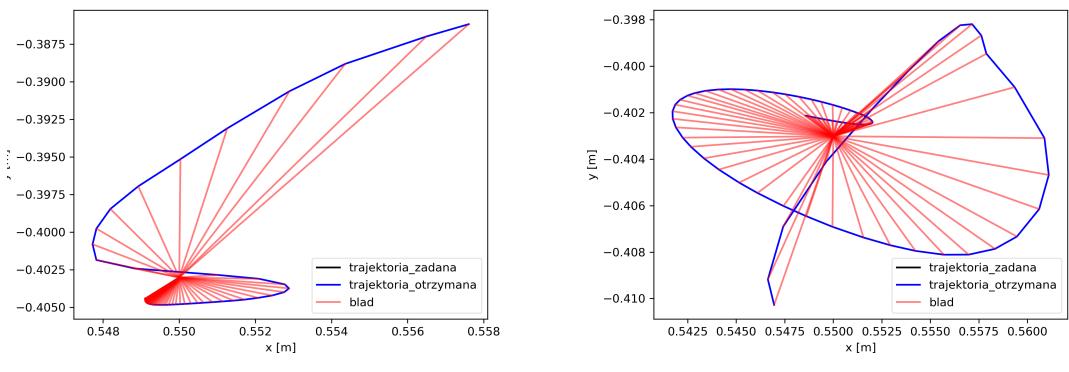
Rysunek 6.33: Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.



(a) Trajektoria z chwycona puszka

(b) Trajektoria z chwycona wiertarka

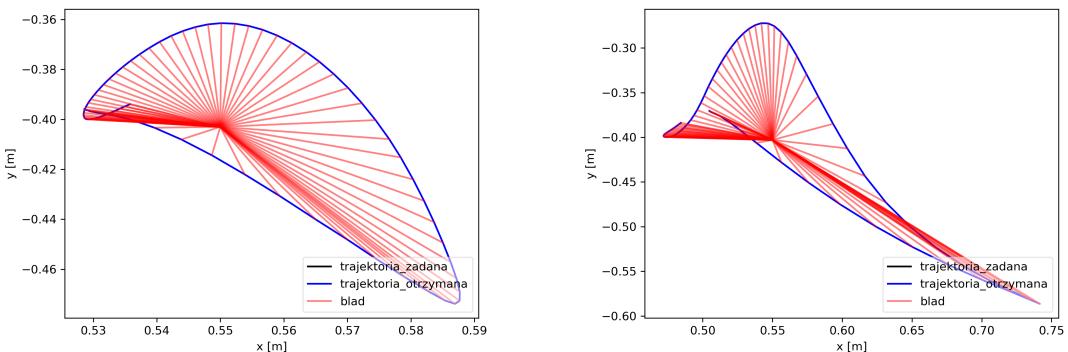
Rysunek 6.34: Porównanie trajektorii chwytaka w osiach X i Z



(a) Brak algorytmu kompensacji

(b) Załączony algorytm kompnesacji

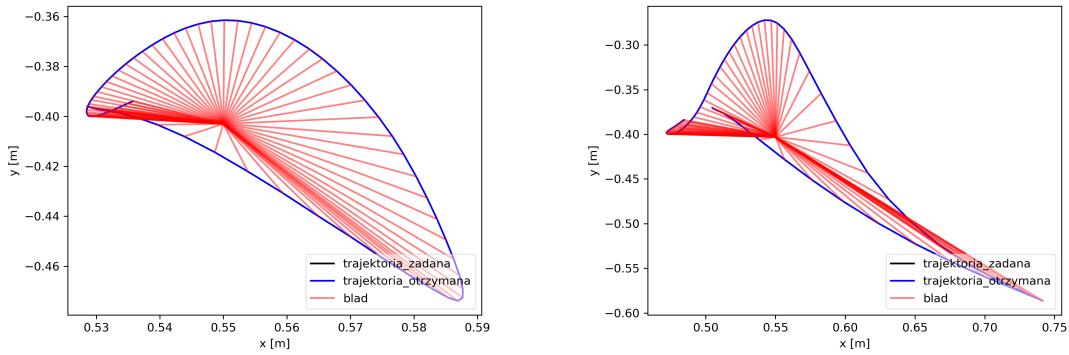
Rysunek 6.35: Porównanie trajektorii chwytaka w osiach X i Y



(a) Trajektoria z chwycona puszka

(b) Trajektoria z chwycona wiertarka

Rysunek 6.36: Porównanie trajektorii chwytaka w osiach X i Y



(a) Trajektoria z chwyconą puszka

(b) Trajektoria z chwyconą wiertarką

 Rysunek 6.37: Porównanie trajektorii chwytaka w osiach X i Y

6.8 Wnioski

Metryka APE jest dobrym narzędziem do oceny jakości trajektorii końcówki. Ruchy tego samego typu mają zbliżony do siebie współczynnik RMSE (tab. 6.1). Zaimplementowane prawo sterowania pozwoliło na skuteczną kompensację siły grawitacji chwyconego przedmiotu. Modyfikacja pozwala na kompensację nie tylko sił grawitacji ale także na ograniczanie innych niedoskonałości modelu stosowanego w prawie sterowania. Osiągane błędy średnio-kwadratowe trajektorii wykonywanych trajektorii są rzędu pojedynczych centymetrów i maksymalnie około dwa razy większe niż w przypadku braku algorytmu kompensacji (tab. 6.1).

W trakcie wszystkich eksperymentów widać, że prawo sterowania osiąga mniejsze błędy dla prostszego obiektu jakim jest puszka z jednorodnym rozkładem mas. Algorytm kompensacji wprowadza oscylacje zarówno położenia jak i rotacji. Zjawisko jest najbardziej widoczne gdy prędkość w końcówce jest wytracana. Przez właściwości zaimplementowanego prawa sterowania podnoszenie przedmiotów trwa długo.

Empiryczne eksperymenty w trakcie pracy symulatora wykazały, że wprowadzone modyfikacje nie miały dużego wpływu na uginanie się robota w trakcie kolizji z przedmiotami chyba że kolizja trwa bardzo długo. Wytlumaczeniem tego pozytywnego zjawiska może być fakt powolnej kompensacji po chwyceniu przedmiotu. Robot potrzebuje ok 15 s aby dodać do końcówki chwytaka siłę o wartości odpowiadającej sile ciężkości jednego kilograma.

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

Opis ruchu – Błąd [m]	Bez alg. komp.	Bez narzędzi	Z puszka	Z wiertarką
Podnoszenie	0.032490	0.053853	0.090244	0.104056
Ruch ósemkowy	0.102113	0.092006	0.089532	0.095919
Ruch w bok	0.030680	0.040400	0.041024	0.047411
Ruch do góry	0.035023	0.051653	0.050177	0.048357
Ruch do dołu	0.038727	0.055311	0.055371	0.057347
Ruch do przodu	0.035068	0.060216	0.058322	0.056084
Obrót końcówki	0.005984	0.007722	0.038998	0.140615

Tablica 6.1: Porównanie błędu średnio-kwadratowego RMSE w metryce APE dla poszczególnych ruchów.

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

Rozdział 7

Podsumowanie

Zaprezentowana modyfikacja prawa sterowania impedancyjnego w przestrzeni operacyjnej pozwoliła na szybkie i proste rozwiązywanie problemu kompensacji grawitacji chwytanego narzędzia. W trakcie manipulacji nawet skomplikowane ruchy z chwyconymi przedmiotami o znacznej masie nie stanowią problemu dla robota. Najbardziej wymagające dla nowego prawa sterowania jest podnoszenie przedmiotów, czyli manipulacja przy zmieniających się paraemtrach narzędzia. Potrzeba adaptacji do narzędzia o nowych parametrach trwa kilka sekund.

W przyszłości można usprawnić algorytm poprzez dodanie ograniczenia członu całkującego zapożyczonego z prawa sterowania PID. Dzięki takiej modyfikacji nie będzie miało miejsce praktycznie żadne pogorszenie uginania się robota w trakcie kolizji. Jako alternatywę można stosować algorytm który działa tylko w pionowej osi układu. Prawdopodobne jest jednak to, że spowoduje to znaczne pogorszenie jakości osiąganych trajektorii. Dalsze badania mogłyby skupić się na przyjęciu uproszczonego modelu robota wraz z narzędziem oraz estymacji jego parametrów z wykorzystaniem czujnika FTS i wyliczonych pozycji końcówki.

Zaprezentowana metodyka pracy stanowi dobre podstawy do testowania zupełnienie nowych i bardziej wysublimowanych algorytmów. Badania w obszarze metodyki powinny skupić się na stworzeniu bardziej precyzyjnych narzędzi do oceny jakości kolizji robota z otoczeniem.

ROZDZIAŁ 7. PODSUMOWANIE

Bibliografia

- [1] Barrett Hand. <https://robots.ros.org/barrett-hand/>. [Dostęp: 21 stycznia 2019 r.].
- [2] Documentation of control system of WUT Velma Robot. https://rcprg-ros-pkg.github.io/velma_docs/. [Dostęp: 21 stycznia 2019 r.].
- [3] Kuka lwr4+ datasheet. https://www.kukakore.com/wp-content/uploads/2012/07/KUKA_LBR4plus_ENLISCH.pdf.
- [4] Oficjalna strona gazebo. <http://www.gazebosim.org/>.
- [5] Oficjalna strona projektu dart. <https://dartsim.github.io/>.
- [6] Oficjalna strona systemu orocos. <http://orocos.org/>.
- [7] Oficjalna strona systemu ROS. <http://www.ros.org/>.
- [8] Oficjalna strona Zakładu Programowania Robotów i Systemów Rozpoznających. <https://www.robotyka.ia.pw.edu.pl/>.
- [9] Useful tools for the rgb-d benchmark. <https://vision.in.tum.de/data/datasets/rgbd-dataset/tools>.
- [10] Loredana Zollo Alessandro De Luca, Bruno Siciliano. PD control with on-line gravity compensation for robots with elastic joints: Theory and experiments. <https://www.sciencedirect.com/science/article/pii/S000510980500186X?via%3Dihub>. [Dostęp: 21 stycznia 2019 r.].
- [11] Rainer Bischoff1, Johannes Kurth, Günter Schreiber, Ralf Koeppe, Alin Albu-Schäffer, Alexander Beyer, Oliver Eiberger, Sami Haddadin, Andreas Stemmer, Gerhard Grunwald, Gerhard Hirzinger. The KUKA-DLR Lightweight Robot arm – a new reference platform for robotics research and manufacturing. IEEE.
- [12] Berthold Horn, Hugh Hilden, Shahriar Negahdaripour. Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices.

BIBLIOGRAFIA

- [13] SJ Huang, G Lallement. Direct estimation of rigid body properties from harmonic forced responses. *PROCEEDINGS-SPIE THE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING*, strony 175–180. SPIE INTERNATIONAL SOCIETY FOR OPTICAL, 1997.
- [14] K. Parsa, J. Angeles, A. K. Misra. Pose-and-twist estimation of a rigid body using accelerometers. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, wolumen 3, strony 2873–2878 vol.3, May 2001.
- [15] Tavakoli Saeed, Griffin Ian, Peter J.Fleming. Tuning of decentralised pi (pid) controllers for tito processes. *Control Engineering Practice*, strony 1069–1080. IEEE, 2006.
- [16] Dawid Seredyński, Tomasz Winiarski, Cezary Zieliński. FABRIC: Framework for Agent-Based Robot Control Systems. K. Kozłowski, redaktor, *12th International Workshop on Robot Motion and Control (RoMoCo)*, strony 215–222. IEEE, 2019.
- [17] P. Song, Y. Yu, X. Zhang. Impedance control of robots: An overview. *2017 2nd International Conference on Cybernetics, Robotics and Control (CRC)*, strony 51–55, July 2017.
- [18] Maciej Stefańczyk, Michał Wałecki, Konrad Banachowicz, Tomasz Winiarski. Robot usługowy Velma – projekt i konstrukcja głowy. *XIII Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 2, strony 451–460, 2014.
- [19] J. Sturm, N. Engelhard, F. Endres, W. Burgard, D. Cremers. A benchmark for the evaluation of rgb-d slam systems. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, strony 573–580, Oct 2012.
- [20] Jie Tan, Kristin Siu, Karen Liu. Contact Handling for Articulated Rigid Bodies Using LCP.
- [21] N Uyama, T Narumi. Hybrid impedance/position control of a free-flying space robot for detumbling a noncooperative satellite. *IFAC-PapersOnLine*, 49(17):230–235, 2016.
- [22] Nie ZY. Wang QG. PID Control for MIMO Processes. Visioli A. Vilanova R., redaktor, *PID Control in the Third Millennium. Advances in Industrial Control.*, strony 177–204. IEEE, 2012.

- [23] Wikipedia. Impedance control — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=Impedance%20control&oldid=904943924>, 2019. [Online; accessed 05-September-2019].
- [24] Wikipedia. PID controller — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=PID%20controller&oldid=913750967>, 2019. [Online; accessed 11-September-2019].
- [25] Tomasz Winiarski, Konrad Banachowicz. Sterowanie redundantnym systemem dwuramiennym z aktywnym korpusem. *XIII Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 2, strony 433–442, 2014.
- [26] Tomasz Winiarski, Konrad Banachowicz, Dawid Seredyński. Two mode impedance control of Velma service robot redundant arm. Roman Szewczyk, Cezary Zieliński, Małgorzata Kaliczyńska, redaktorzy, *Progress in Automation, Robotics and Measuring Techniques. Vol. 2 Robotics.*, wolumen 351 serii *Advances in Intelligent Systems and Computing (AISC)*, strony 319–328. Springer, 2015.
- [27] Tomasz Winiarski, Dawid Seredyński. Wizualizacja sterowników robotów bazujących na teorii agenta upostaciowanego. *XV Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 1, strony 417–426, 2018.
- [28] Cezary Zieliński. Architektury systemów robotycznych tworzone z agentów upostaciowionych. 196:379–394, 2018. Robotic System Architectures Based on Embodied Agents (in Polish) - Zielinski.
- [29] Cezary Zieliński. Zastosowanie agentów upostaciowionych do projektowania systemów robotycznych. J. Kacprzyk P. Kulczycki, J. Korbicz, redaktor, *Automatyka, robotyka i przetwarzanie informacji*. Wydawnictwo Naukowe PWN, 2019. Zielinski.

BIBLIOGRAFIA

Spis rysunków

2.1	Rysunek przedstawia rzut APE w dwóch wymiarach. Czarną i niebieską linią oznaczono zadaną i osiąganą trajektorię. Linie czerwone łączą odpowiednie punkty na obydwu trajektoriach i obrazują błąd pomiędzy nimi.	19
2.2	Rysunek przedstawia schemat budowy agenta upostaciowanego [?]. Typ podsystemu oznaczany jest dużą literą. Podsystemy i bufory posiadają prawy dolny indeks. Typ podsystemu jest oznaczany lewym dolnym indeksem. Typ buforu jest oznaczany lewym dolnym indeksem (x - wejściowy, y - wyjściowy).	20
3.1	Robot usługowy Velma	22
3.2	Poglądowy i uproszczony schemat agentów robota Velma zawierający także nieopisywane części systemu odpowiedzialne za nadzór Kinecta [2].	23
3.3	Robot Velma w trakcie manipulacji. Symulacja w środowisku Gazebo.	26
4.1	Diagram aktywności pokazujący przebieg sterowania ramieniem robota.	28
4.2	Diagram przypadków użycia robota z uwzględnieniem algorytmu kompensacji grawitacji	29
4.3	Schemat nowego prawa sterowania. W pierwszej fazie wyliczany jest algorytm z członem całkującym a następnie transformowany do przestrzeni stawów przez jakobian.	31
4.4	Diagram aktywności pokazujący przebieg sterowania ramieniem robota z aktualizacją modelu.	32
6.1	Podnoszenie przedmiotu. Porównanie trajektorii z różnymi parametrami członu całkującego.	40
6.2	Podnoszenie przedmiotu. Porównanie parametrów członu całkującego jako kątów w notacji Eulera w zależności od czasu.	41
6.3	Podnoszenie przedmiotu. Porównanie trajektorii pozycji w zależności od czasu.	42

SPIS RYSUNKÓW

6.4 Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach Y i Z	42
6.6 Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach Y i Z	43
6.5 Podnoszenie przedmiotu. Porównanie trajektorii kątów w notacji Eu- lera w zależności od czasu.	43
6.7 Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach X i Z	44
6.8 Podnoszenie przedmiotu. Porównanie trajektorii chwytaka w osiach X i Z	44
6.9 Ruch ósemkowy. Porównanie trajektorii pozycji w zależności od czasu.	45
6.10 Porównanie trajektorii chwytaka w osiach Y i Z	45
6.11 Ruch ósemkowy. Porównanie trajektorii kątów w notacji Eulera w za- leżności od czasu.	46
6.12 Ruch ósemkowy. Porównanie trajektorii chwytaka w osiach Y i Z . .	46
6.13 Ruch ósemkowy. Porównanie trajektorii chwytaka w osiach X i Z . .	47
6.14 Ruch ósemkowy. Porównanie trajektorii chwytaka w osiach X i Z . .	47
6.15 Ruch w bok. Porównanie trajektorii pozycji w zależności od czasu. .	48
6.16 Ruch w bok. Porównanie trajektorii chwytaka w osiach Y i Z	48
6.17 Ruch w bok. Porównanie trajektorii kątów w notacji Eulera w zależ- ności od czasu.	49
6.18 Ruch w bok. Porównanie trajektorii chwytaka w osiach Y i Z	49
6.19 Ruch do góry. Porównanie trajektorii pozycji w zależności od czasu. .	50
6.20 Ruch do góry. Porównanie trajektorii chwytaka w osiach X i Z . . .	51
6.21 Ruch do góry. Porównanie trajektorii kątów w notacji Eulera w za- leżności od czasu.	51
6.22 Ruch do góry. Porównanie trajektorii chwytaka w osiach X i Z	52
6.23 Ruch do dołu. Porównanie trajektorii pozycji w zależności od czasu. .	53
6.24 Ruch do dołu. Porównanie trajektorii chwytaka w osiach X i Z	53
6.25 Ruch do dołu. Porównanie trajektorii kątów w notacji Eulera w za- leżności od czasu.	54
6.26 Porównanie trajektorii chwytaka w osiach X i Z	54
6.27 Ruch do przodu. Porównanie trajektorii pozycji w zależności od czasu.	55
6.28 Ruch do przodu. Porównanie trajektorii chwytaka w osiach X i Z . .	55
6.29 Ruch do przodu. Porównanie trajektorii kątów w notacji Eulera w za- leżności od czasu.	56
6.30 Ruch do przodu. Porównanie trajektorii chwytaka w osiach X i Z . .	56
6.31 Porównanie trajektorii pozycji w zależności od czasu.	57
6.32 Porównanie trajektorii chwytaka w osiach X i Z	58

SPIS RYSUNKÓW

6.33 Porównanie trajektorii kątów w notacji Eulera w zależności od czasu.	58
6.34 Porównanie trajektorii chwytaka w osiach X i Z	59
6.35 Porównanie trajektorii chwytaka w osiach X i Y	59
6.36 Porównanie trajektorii chwytaka w osiach X i Y	59
6.37 Porównanie trajektorii chwytaka w osiach X i Y	60

SPIS RYSUNKÓW

Spis tablic