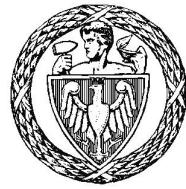


Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH



Instytut Automatyki i Informatyki Stosowanej

Praca dyplomowa magisterska

na kierunku Automatyka i Robotyka

Chwyтанie przedmiotów
w impedancyjnym prawie sterowania

Jakub Postępski
Numer albumu 257947

promotor
dr inż. Tomasz Winiarski

Warszawa 2019

Streszczenie

Tytuł: *Chwytyanie przedmiotów w robocie sterowanym impedancyjnie*

Celem pracy magisterskiej była 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 upustaciowanego i zaimplementowane przy użyciu struktury ramowej FABRIC. Robot korzysta z impedancyjnego prawa sterowania i samodzielnie kompensuje siłę grawitacji ramion.

W pracy zaprezentowano możliwe rozwiązywanie opisanego problemu wraz z testami. Zaproponowano możliwe kierunki rozwoju w kontekście przeprowadzonych badań.

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

Abstract

Title: *Grasping objects in impedance controll robot*

The aim of this thesis is algorithm with gravity compensation

Keywords: *grasping, localization, IRp-6 robot, ROS, DisCODE, OpenCV*



„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 upustaciowanego	19
3 Środowisko badawcze	21
3.1 Budowa ogólna	21
3.2 Pakiety oprogramowania	21
3.2.1 ROS	21
3.2.2 Orocoss	22
3.2.3 FABRIC	22
3.3 Symulator robota	23
3.4 Agenty	23
4 Specyfikacja systemu	27
4.1 Wymagania	27
4.2 Założenia	28
4.3 Przypadki użycia	28
5 Implementacja systemu	31
5.1 Kompensowanie wpływu grawitacji narzędzia	31

SPIS TREŚCI

6 Działanie systemu	33
6.1 Ocena jakości sterowania	33
6.1.1 Podnoszenie przedmiotu	33
6.1.2 Ruch ósemkowy	34
6.1.3 Ruch w bok	36
6.1.4 Ruch do gory	37
6.1.5 Ruch do dolu	39
6.1.6 Ruch do przodu	41
6.1.7 Obrot koncowki	42
6.1.8 Ocena jakości kolizji	42
6.1.9 Podsumowanie	43
7 Podsumowanie	57
7.1 Wnioski	57
7.2 Perspektywy rozwoju	57

Rozdział 1

Wstęp

Współczesne roboty z powodzeniem zastępują człowieka przy wielu żmudnych i niewdzięcznych czynnościach. W zakładach produkcyjnych sprawują się znakomicie. 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 robotyki 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 chwytanych 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 opisanym środowisku. Do badań posłuży robot Velma oraz jego symulator skonstruowane przez Zespół Rrogramowania 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 [10], [15] w przestrzeni operacyjnej sprawia, że chwytek robota zachowuje się jak przytwierdzony do układu ze sprężyną i amortyzatorem. Pojawienie się nieprzewidzianych sił zewnętrznych powoduje, że uchyb pozycji nie jest minimalizowany za wszelką cenę. Przy kontakcie z otoczeniem stawy robota w pewnym stopniu sprężyste i miękkie. W konsekwencji robot uginie się przed otaczającym go środowiskiem.

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

Można sformułować prawo sterowania 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} . Można uzyskać porządane wartości wektora momentów $\boldsymbol{\tau}$ które zadajemy silnikom w stawach stawie wyliczamy z jacobianu \mathbf{J} :

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

2.2 Sterowanie PID

PID (ang. Proportional Integral Derivative) [8], [14] jest bardzo 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 jest postaci:

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

gdzie:

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

Prawo sterowania możemy zapisać w formie transmitancji:

$$G(s) = P + \frac{1}{Is} + Ds \quad (2.6)$$

2.2.2 Przypadek wielowymiarowy

TODO: przepisać wzór z artykułu. Rozpatrując wektor siły uogólnionej możemy założyć że prawo sterowania rozpatruje każdą z wartości wektora niezależnie. Prawo sterowania można zapisać w postaci:

$$\mathcal{G}() = \mathbf{P}\mathbf{e}_x + \int_0^t \mathbf{I}\mathbf{e}_x dt + \mathbf{D}\dot{\mathbf{e}}_x \quad (2.7)$$

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:

$$\mathcal{F}_m(\mathbf{x}, \dot{\mathbf{x}}, \ddot{\mathbf{x}}, \mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\Lambda}(\mathbf{q})\ddot{\mathbf{x}} + \boldsymbol{\mu}(\mathbf{x}, \dot{\mathbf{x}}) + \boldsymbol{\gamma}(\mathbf{q}) + \boldsymbol{\eta}(\mathbf{q}, \dot{\mathbf{q}}) + \mathcal{F}_{ext} \quad (2.8)$$

gdzie:

- \mathcal{F} to wektor sił wynikowych
- Λ to dodatnio określona macierz inercji w przestrzeni zadań
- μ 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:

$$\hat{\mathcal{F}} = \mathcal{F} + \mathcal{F}_m(x, \dot{x}, \ddot{x}, q, \dot{q}) \quad (2.9)$$

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) [12]. Metryka jest popularnym wskaźnikiem testowania algorytmów SLAM (ang. Simultaneous Localization and Mapping) ale może być też użyta do porównania trajektorii zadanej przez interpolator i rzeczywistej (rys. 2.1).

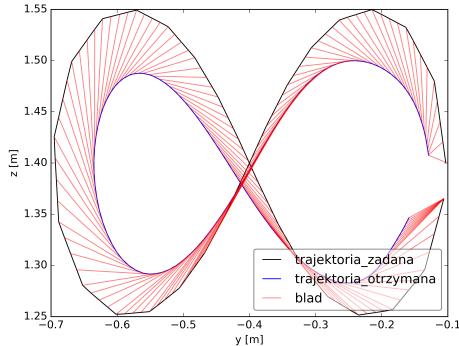
Dwie trajektorie są opisane w postaci list wektorów sił uogólnionych $\mathbf{P}_{1..n}$ oraz $\mathbf{Q}_{1..n}$ gdzie n to ilość próbek. Dla każdej chwili czasowej i jest wyliczany błąd postaci:

$$\mathbf{E}_i = \mathbf{Q}_i^{-1} \mathbf{S} \mathbf{P}_i \quad (2.10)$$

Macierz \mathbf{S} jest optymalnym w sensie metody najmniejszych kwadratów rzutowaniem wektora \mathbf{Q}_i na wektor \mathbf{P}_i znalezionym za pomocą metody Horna [7].

Błąd całkowity jest wyliczany jako błąd średniokwadratowy:

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



Rysunek 2.1: Rysunek przedstawia trajektorie zrzutowane na rysunek 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 średniokwadratowy:

$$RMSE(\mathbf{R}_{1..n}) = \sqrt{\frac{1}{n} \sum_{i=1}^n \|\mathbf{q}_d - \mathbf{q}_i\|^2} \quad (2.12)$$

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 [19] [18]. Agent upostaciowany cechuje się tym czym zwykły agnet oraz posiada fizyczne ciało.

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 a 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.

ROZDZIAŁ 2. PODSTAWY TEORETYCZNE

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

Podsystemy składają się z komponentów czyli algorytmów. Każdy z podsystemów może mieć wiele komponentów i nie wszystkie muszą być uruchomione w konkretnej chwili. Agent posiada też zdefiniowaną maszynę stanów które mogą się zmieniać przy pomocy funkcji przejścia (predykatów). Stan maszyny stanów definiuje tak zwane zachowanie agenta czyli sposób reakcji podsystemu sterowania i uruchomione komponenty. Dodatkowo agent ma możliwość wymiany danych innymi agentami oraz rzeczywistymi efektorami i receptorami poprzez bufory transmisyjne T .

Rozdział 3

Środowisko badawcze

3.1 Budowa ogólna

Środowiskiem badawczym jest robot usługowy Velma [?]. Został zaprojektowany i wykonany przez Zespół Rrogramowania Robotów i Systemów Rozpoznających [5]. Do obrotowego korpusu przytwierdzono dwa ramiona robotyczne Kuka LWR-4+ [1] [6]. Mają siedem stopni swobody i udźwig 7 kg. Na ich końcach znajdują się chwytaki Barretta oraz nadgarstkowe czujniki FTS. Głowa robota umieszczona jest na dedykowanej konstrukcji która za pomocą dwóch silników elektrycznych pozwala na zginanie i obrót głowy [11] [16]. Robot wyposażony jest w czujnik wizyjny Microsoft Kinect oraz dwie kamery połączone w stereoparę. Do głowy zamocowano mikrofon. Ramiona robota są połączone z komputerem sterującym przy pomocy magistrali FRI a pozostałe przy pomocy magistrali EtherCAT.

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 Pakiety oprogramowania

3.2.1 ROS

Struktura ramowa ROS (Robot Operating System) [4] 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 liczące kinematykę i wizualizujące pracę robota. Daje możliwość akwizycji danych. Podstawowymi narzędziami w pakiecie są:

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE

- **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 bierzącego stanu całego systemu.
- **Zarządca** - odpowiada za komunikację i nadzoruje pracę innych narzędzi pakiety.

3.2.2 Orocos

Orocos [3] jest wolnym oprogramowaniem napisanym w C++ służącym do pisania aplikacji zgodnych z wymaganiami czasu rzeczywistego. Pozwala na tworzenie komponentów odwzorowujących modele stworzone przy pomocy teorii agentowej. Zestaw bibliotek składa się między innymi z:

- **RTL (ang. Real-Time Toolkit)** - Biblioteki służące do tworzenia komponentów w aplikacjach czasu rzeczywistego.
- **OCL (ang. Orococos 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 trakcie pracy.

3.2.3 FABRIC

Framework for Agent-Based Robot Control Systems - FABRIC [9] wykorzystuje Orocosa oraz strukturę ramową ROS zapewniając interfejs programistyczny pozwalający na tworzenie komponentów zgodnych z założeniami teorii agentowej.

Ma zaimplementowane algorytmy komunikacji pomiędzy poszczególnymi podsystemami poprzez zdefiniowane wiadomości. Posiada narzędzie wizualizujące stan predykatów, zachowań i podsystemów *rqt_agent* [17]. Pokazuje też przepływ danych między komponentami.

3.3 Symulator robota

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

Orginalny kod Gazebo został przystosowany do pakietu oprogramowania DART (ang. Dynamic Animation and Robotics Toolkit) [2] 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 daje dostęp do wielu przydatnych informacji takich jak jakobian bądź macierz inercji przemiotów. Zastosowanow w nim algorytm LPC (ang. Linear complementarity problem) [13] który sprowadza model fizyczny świata do zestawu równań kwadratowych. DART symuluje działanie wielu sił w tym Coriolisa i tarcia dynamicznego. Dzięki temu możliwe jest zaawansowane wykrywanie kolizji.

3.4 Agenty

System sterowania zbudowany jest z dwóch agentów. 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 pisanyimi w ROS oraz agentem *velma_core*. Zawiera tylko jeden podsystem o tej samej nazwie.

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

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE

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 obsługi 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 magistrali. 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 chwytnika 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*.

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE

- **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.

ROZDZIAŁ 3. ŚRODOWISKO BADAWCZE

Rozdział 4

Specyfikacja systemu

Sterowanie impedancyjne w przestrzeni operacyjnej zastosowane w robocie Velma kompensuje jedynie masę członów własnych. Gdy manipulator chwyci narzędzie powstaje znaczny uchyb, który nie jest kompensowany. W przypadku chwycenia przedmiotu o dużej masie ramię robota jest wręcz unieruchomione. Należy skonstruować i zaimplementować algorytm kompensacji siły grawitacji narzędzia o nieznanych parametrach.

4.1 Wymagania

Wymagania są narzucone przez system środowiska badawczego.

- Do działania systemu wymagane jest ramię robotyczne sterowane prawem sterowania impedancyjnego w przestrzeni kartezjańskiej
- 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

4.2 Założenia

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.

- Wyliczone przez prawo sterowania momenty zadawane w stawach zostaną zmodyfikowane przez algorytm kompensacji grawitacji
- Algorytm kompensacji ma minimalizować uchyb statyczny wynikający z chwycenia przedmiotu o nieznanej masie i inercji.
- Trajektoria końcówki ramienia powinna być zbliżona do trajektorii analogicznego ruchu bez uchwyconego przedmiotu.
- Algorytm kompensacji nie może zaburzać cech sterowania impedancyjnego pozwalających na uginanie się robota w momencie kolizji

4.3 Przypadki użycia

Algorytm kompensujący ma działać przez cały czas pracy robota gdy pracuje z narzędziem. Przypadki użycia algorytmu (rys.) są zbieżne z przypadkami użycia robota kiedy pracuje w trybie sterowania impedancyjnego.

Aktorami korzystającymi z systemu są:

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

TODO: dorobic obrazek

Można wyszczególnić następujące przypadki użycia:

- **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 Chwytywanie 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.

ROZDZIAŁ 4. SPECYFIKACJA SYSTEMU

- **PU4 Odkładanie narzędzia** - Robot odkłada narzędzie a następnie rozwiera chwytak.
- **PU5 Kompensacja grawitacji** - Robot kompensuje siłę grawitacji chwyconego narzędzia

ROZDZIAŁ 4. SPECYFIKACJA SYSTEMU

Rozdział 5

Implementacja systemu

W ramach pracy zmodyfikowano istniejący system robotyczny opisany w rozdziale 3. Dzięki temu możliwe było spełnienie założeń i wymagań z rozdziału 4.

5.1 Kompensowanie wpływu grawitacji narzędzia

Kiedy chwytak robota chwyci narzędzie zmienia się łańcuch kinematyczny. W pracy zajmujemy się przypadkiem w którym po uchwyceniu narzędzie zostaje nieruchome w stosunku do chwytaka. Można przyjąć, że zmieniają się wtedy parametry ostatniego członu związane z masą i inercją.

Głównym problemem w trakcie manipulacji z nieskompensowaną grawitacją jest znaczny uchyb statyczny. Aby wyeliminować ten efekt można do istniejącego algorytmu sterowania impedancyjnego dodać algorytm PID.

Należy mieć na uwadze, że niektóre czlony 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:

$$\mathcal{F} = \mathbf{K}_x \mathbf{e}_x + \mathbf{D}_x \dot{\mathbf{e}}_x + \int_0^t \mathbf{I} \mathbf{e}_x dt \quad (5.1)$$

gdzie:

- \mathcal{F} to wektor sił wynikowych
- \mathbf{K}_x to diagonalna macierz sprężystości
- \mathbf{D}_x to diagonalna macierz sztywności
- \mathbf{I} to diagonalna macierz członu całkującego
- \mathbf{e}_x to wektor uchybu

ROZDZIAŁ 5. IMPLEMENTACJA SYSTEMU

Rozdział 6

Działanie systemu

Algorytm kompensacji grawitacji zaimplementowano modyfikując komponent *cart_imp* odpowiedzialny za wyliczanie prawa sterowania oraz inne komponenty. W celu weryfikacji wymagań systemu został napisany program komunikujący się z agentem *velma_ros_interface*. Program nakazujący robotowi manipulowanie chwytnikiem w trybie sterowania impedancyjnego w przestrzeni operacyjnej uruchomiono w trybie symulacji. Program jest uruchamiany w trybie wysokiej i niskiej sprezystości oraz z włączonym i wyłączonym algorytmem kompensacji grawitacji. W trakcie wszystkich eksperymentów wszystkie parametry algorytmu sterowania były takie same. TODO: dodac osie w sekcji o robocie

6.1 Ocena jakości sterowania

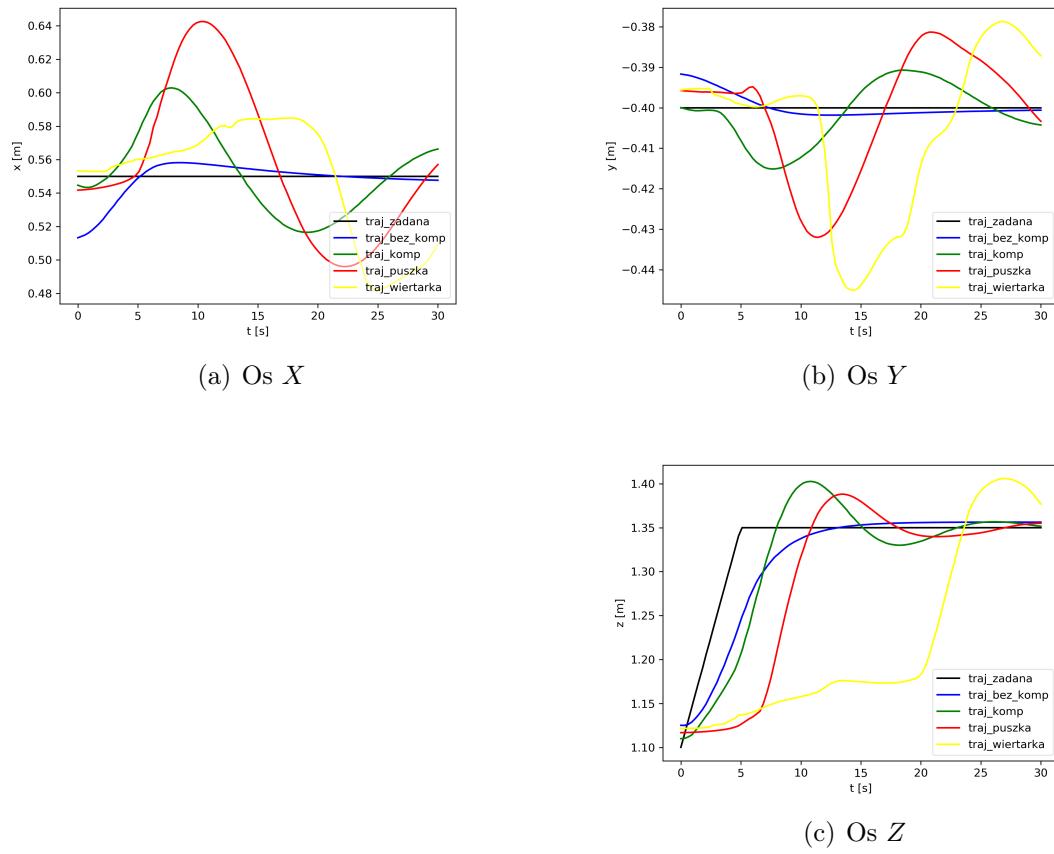
Okreslenie jakości algorytmu możliwe jest dzięki modyfikacji komponentu *TfPublisher*. Po modyfikacji udostępnia on dane o pozycji zadanej z interpolatora trajektorii oraz o pozycji osiągniętej. Ocena jakości sterowania następuje poprzez porównanie wyników obliczonych zgodnie z metryką APE opisanej w sekcji 2.4. W pracy nie zamieszczono wyników działania algorytmu sterowania bez kompensacji i z chwyconymi przedmiotami ponieważ nie były one w zadnym stopniu wykonywane. Testy polegaly na chwytaniu dwóch przedmiotów. Puszka ma wagę ok 1 kg jest walcem z rowomierne rozłożona masa. Wiertarka ma wagę ok 1,5 kg i jej rozkład mas nie jest równomierny.

6.1.1 Podnoszenie przedmiotu

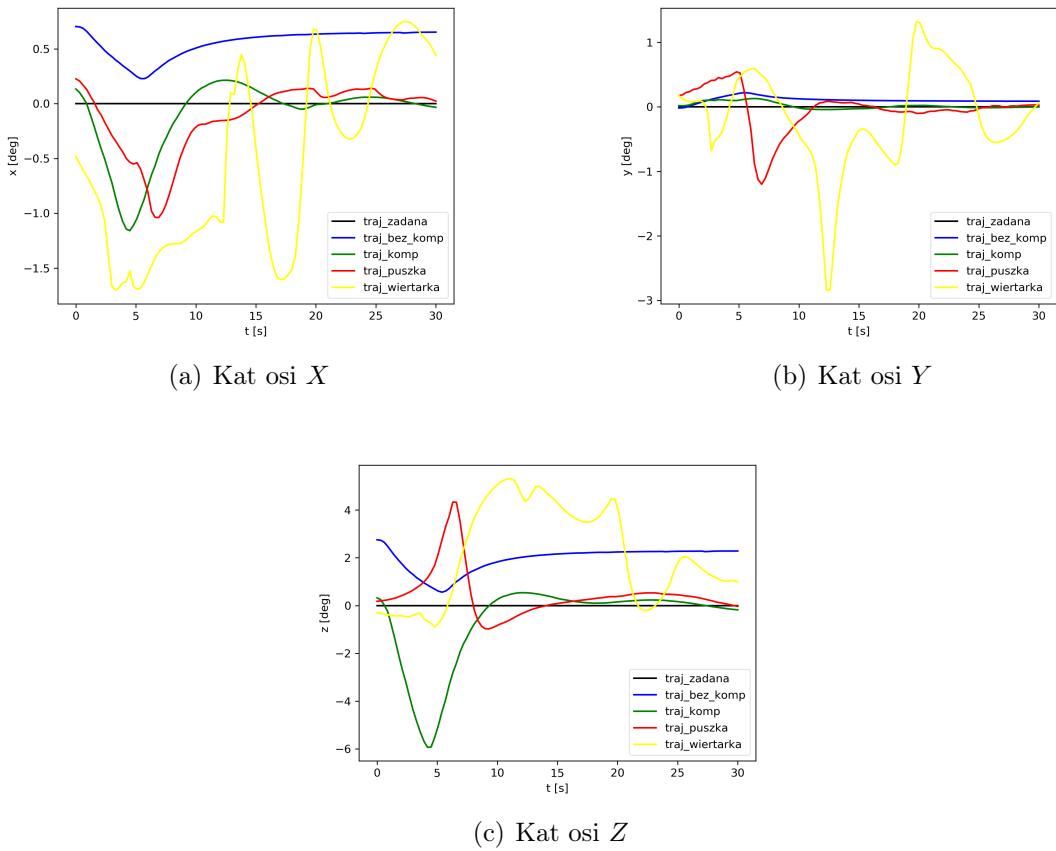
Eksperyment ma przetestować zachowanie algorytmu kompensacji przy podnoszeniu przedmiotu o nieznanych parametrach ruchach (rys. 6.1, 6.2). Ruch widoczny jest głównie w osi *Z*

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

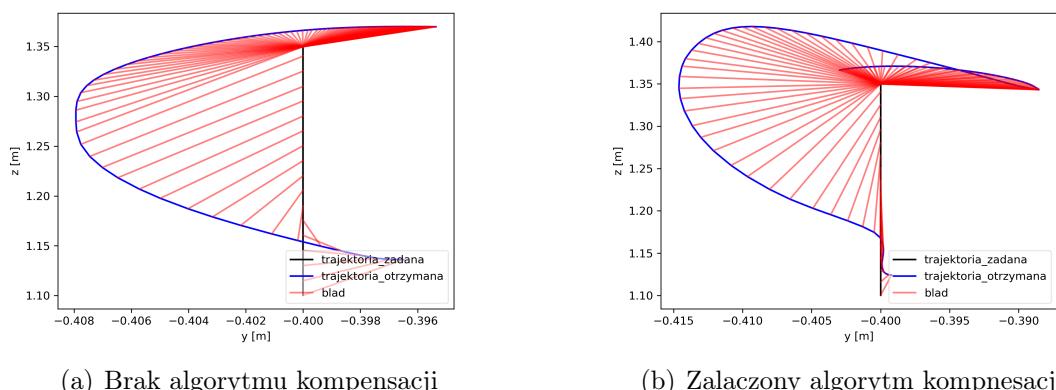
Trajektoria ruchu w rzucie na wprost ruchu zostala zaprezentowana na rys. 6.3, 6.4 i 6.7(a). Trajektoria widoczna z boku (w osiach X oraz Z) zostala zaprezentowana na rys. 6.5, 6.6 i 6.7(b).



Rysunek 6.1: Ruch osemkowy. Porownanie trajektorii pozycji w zaleznosci od czasu.

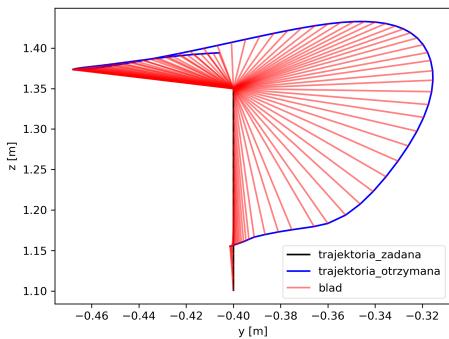


Rysunek 6.2: Ruch osemkowy. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.

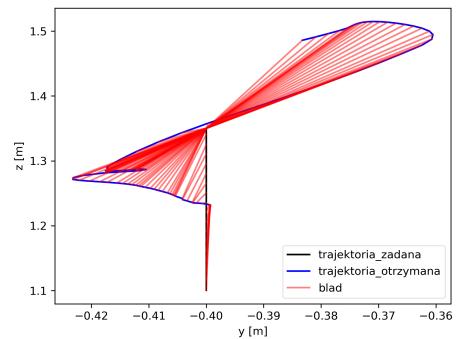


Rysunek 6.3: Porównanie trajektorii chwytaka w osiach Y i Z

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

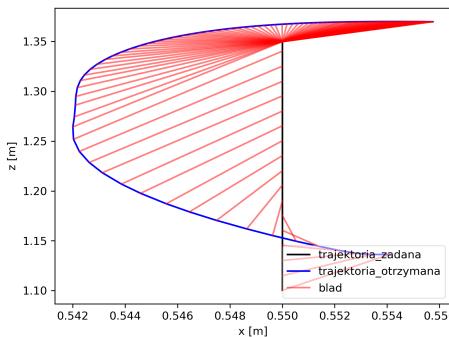


(a) Trajektoria z chwycona puszka

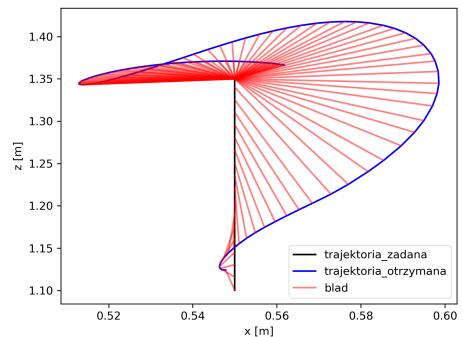


(b) Trajektoria z chwycona wiertarka

Rysunek 6.4: Porównanie trajektorii chwytaka w osiach Y i Z

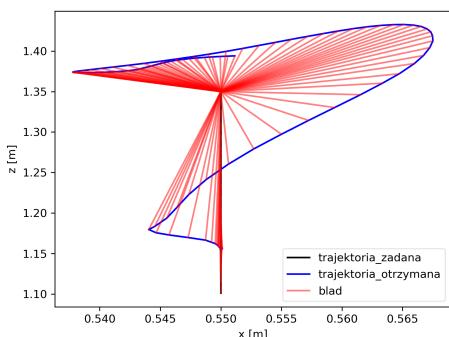


(a) Brak algorytmu kompensacji

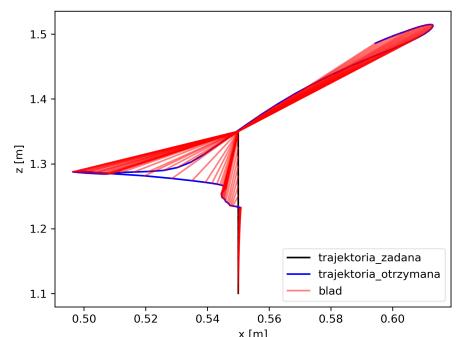


(b) Zalaczony algorytm kompensacji

Rysunek 6.5: Porównanie trajektorii chwytaka w osiach X i Z

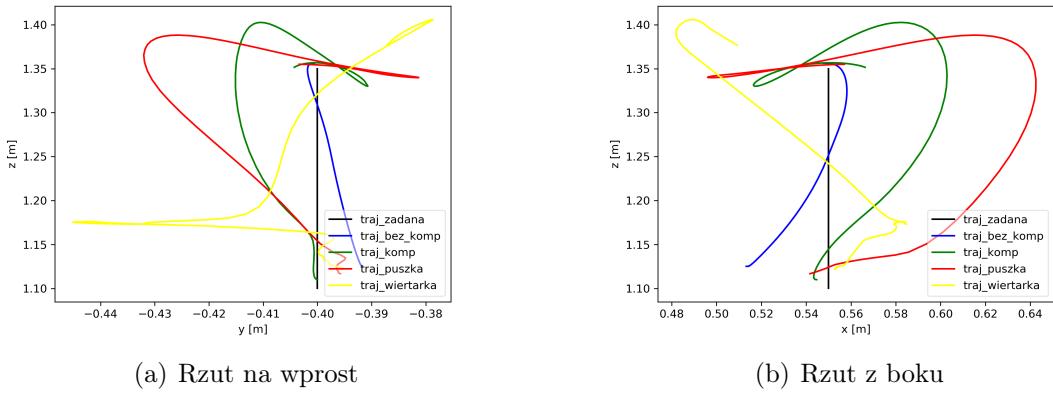


(a) Trajektoria z chwycona puszka



(b) Trajektoria z chwycona wiertarka

Rysunek 6.6: Porównanie trajektorii chwytaka w osiach X i Z



Rysunek 6.7: Porowanie wszystkich trajektorii bez zaznaczonego bledu.

6.1.2 Ruch ósemkowy

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy skomplikowanych ruchach (rys. 6.8, 6.9). Ruch ósemkowy zadany jest w osi Y oraz Z . Trajektoria jest zadana zgodnie ze wzorem lemniskaty Bernoulliego opisanej wzorem:

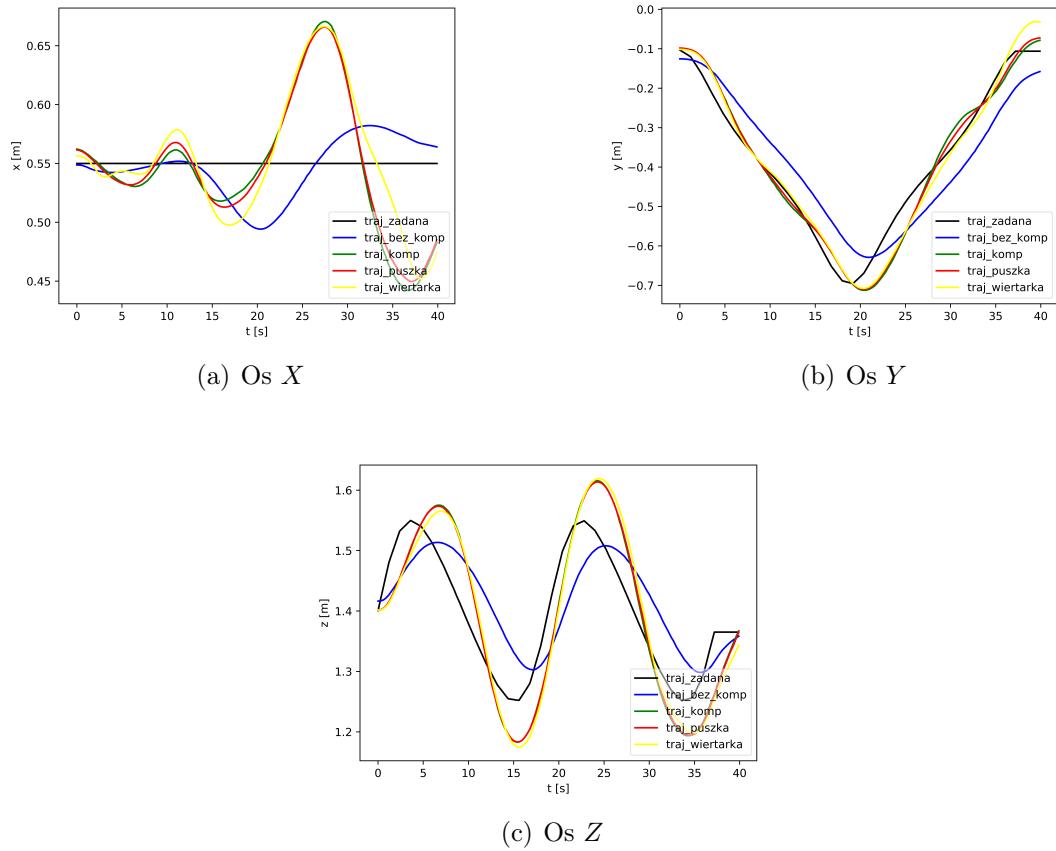
$$(y^2 + z^2)^2 = 2a^2(y^2 - z^2) \quad (6.1)$$

gdzie:

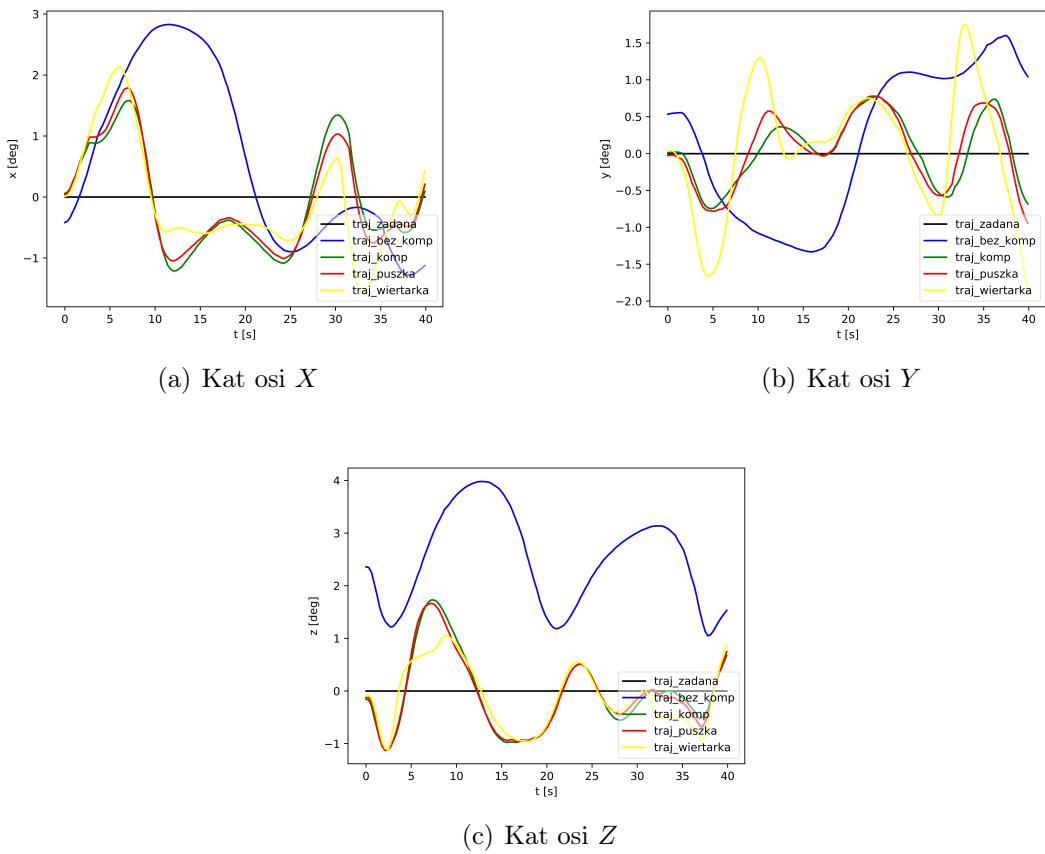
- y oraz z to współrzędne trajektorii
- a to parametr równania

Trajektoria ruchu w rzucie na wprost ruchu została zaprezentowana na rys. 6.10, 6.11 i 6.14(a). Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.12, 6.13 i 6.14(b).

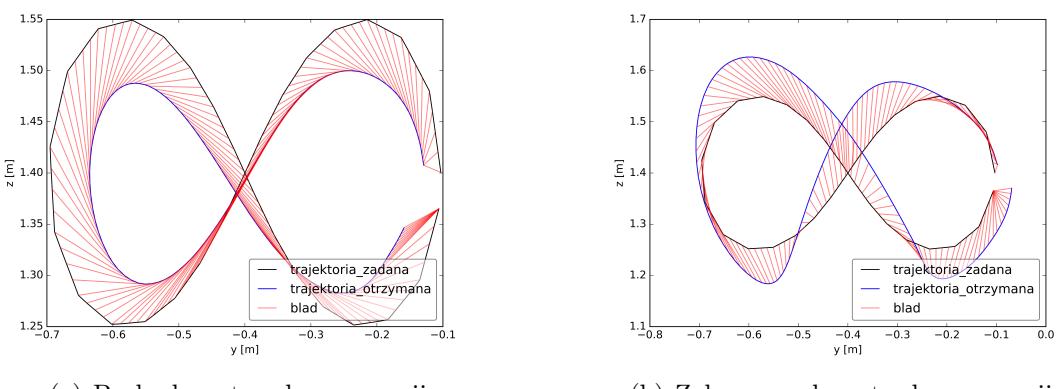
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



Rysunek 6.8: Ruch osemkowy. Porównanie trajektorii pozycji w zaleznosci od czasu.

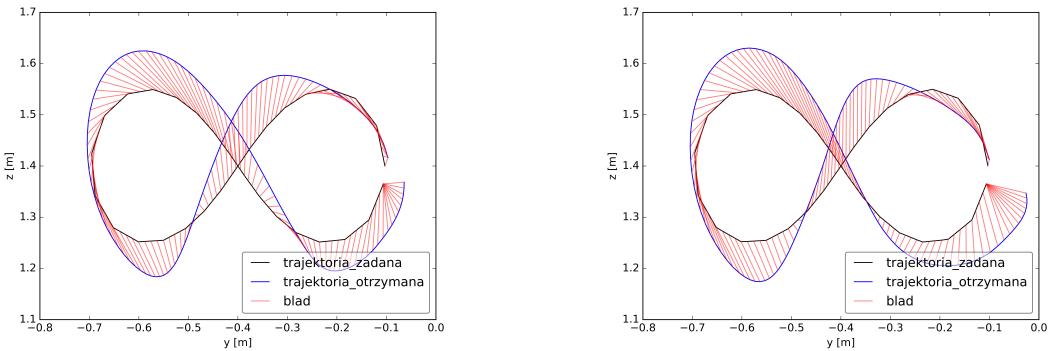


Rysunek 6.9: Ruch osemkowy. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.



Rysunek 6.10: Porównanie trajektorii chwytaka w osiach Y i Z

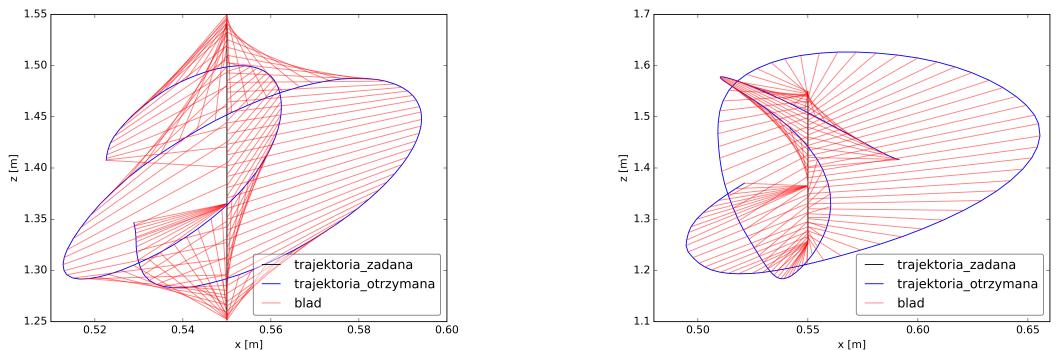
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



(a) Trajektoria z chwycona puszka

(b) Trajektoria z chwycona wiertarka

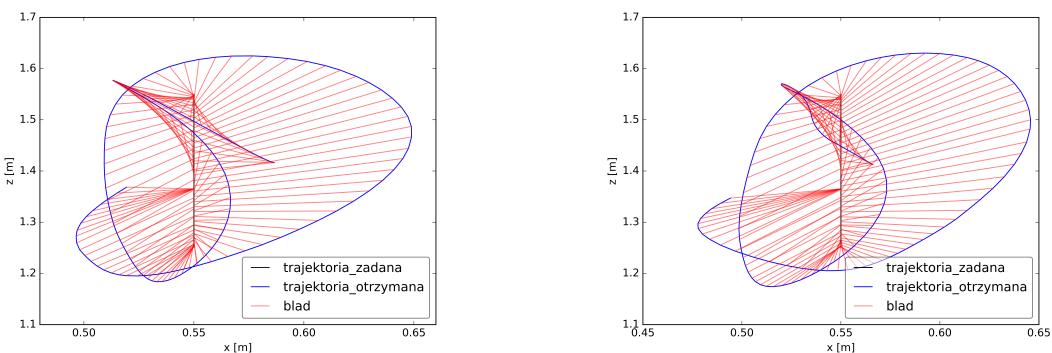
Rysunek 6.11: Porównanie trajektorii chwytaka w osiach Y i Z



(a) Brak algorytmu kompensacji

(b) Zalaczony algorytm kompensacji

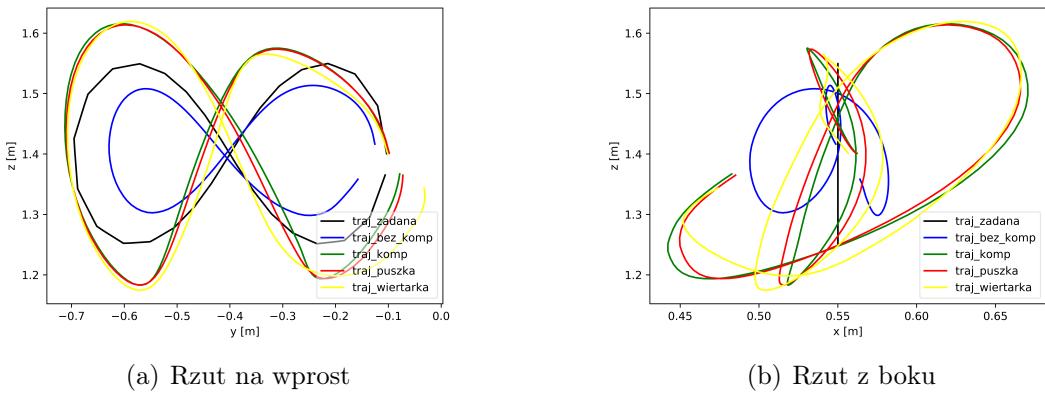
Rysunek 6.12: Porównanie trajektorii chwytaka w osiach X i Z



(a) Trajektoria z chwycona puszka

(b) Trajektoria z chwycona wiertarka

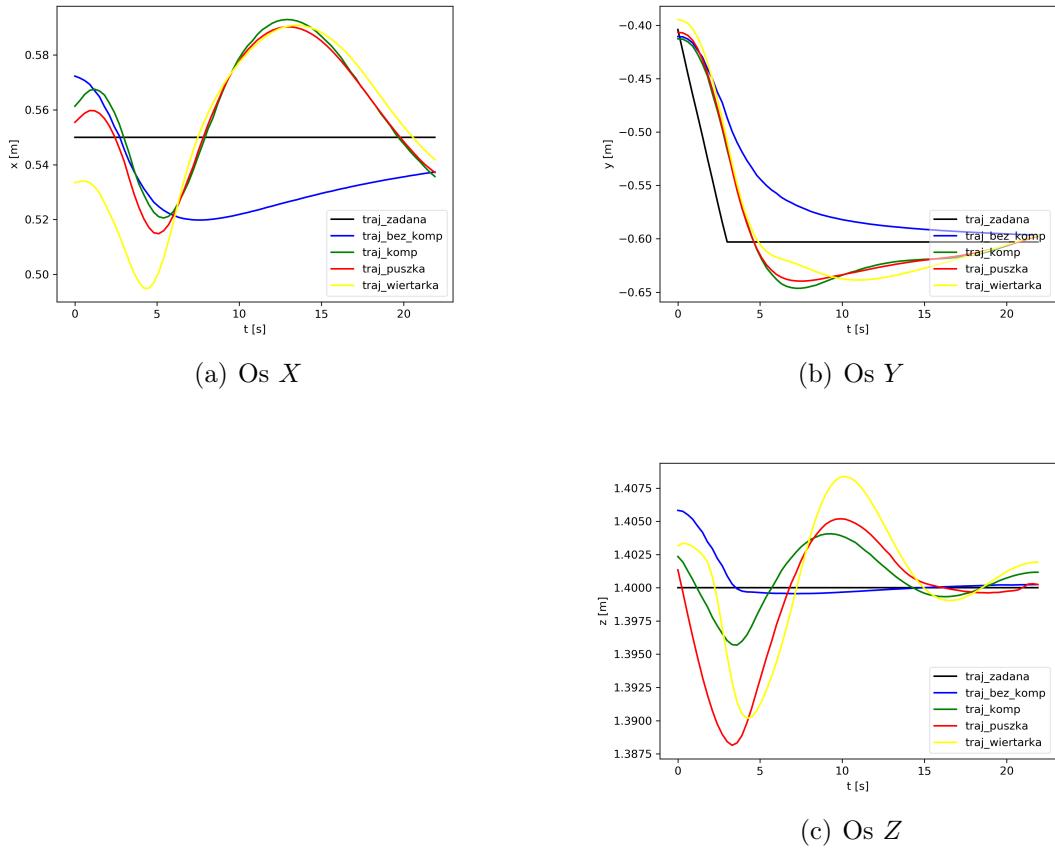
Rysunek 6.13: Porównanie trajektorii chwytaka w osiach X i Z



Rysunek 6.14: Porowanie wszystkich trajektorii bez zaznaczonego bledu.

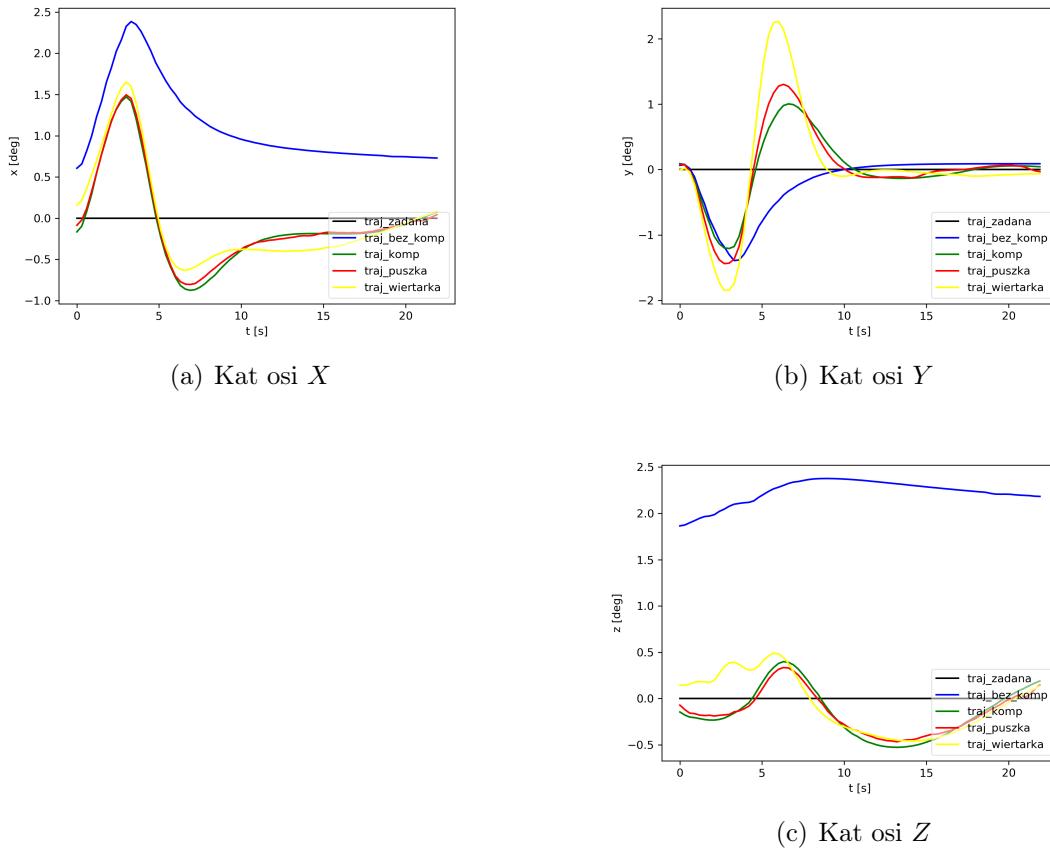
6.1.3 Ruch w bok

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu koncowki w bok. Trajektoria ruchu w rzucie na wprost ruchu została zaprezentowana na rys. 6.17, 6.18 i 6.19(a).

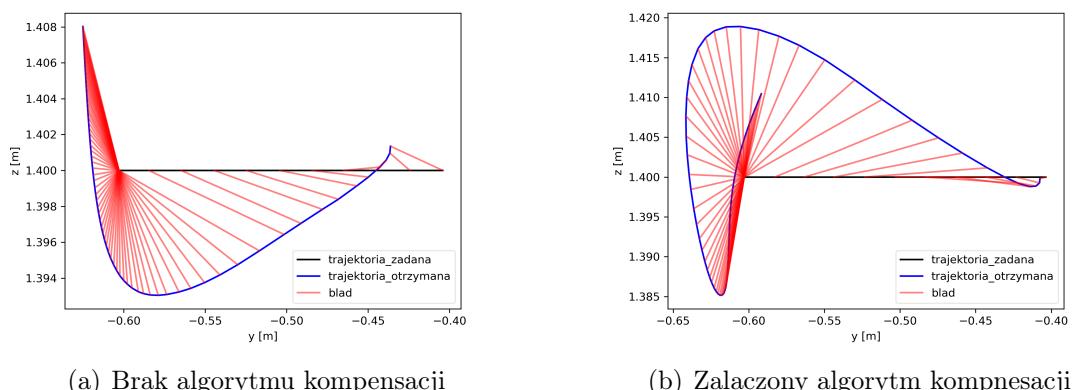


Rysunek 6.15: Ruch do gory. Porownanie trajektorii pozycji w zaleznosci od czasu.

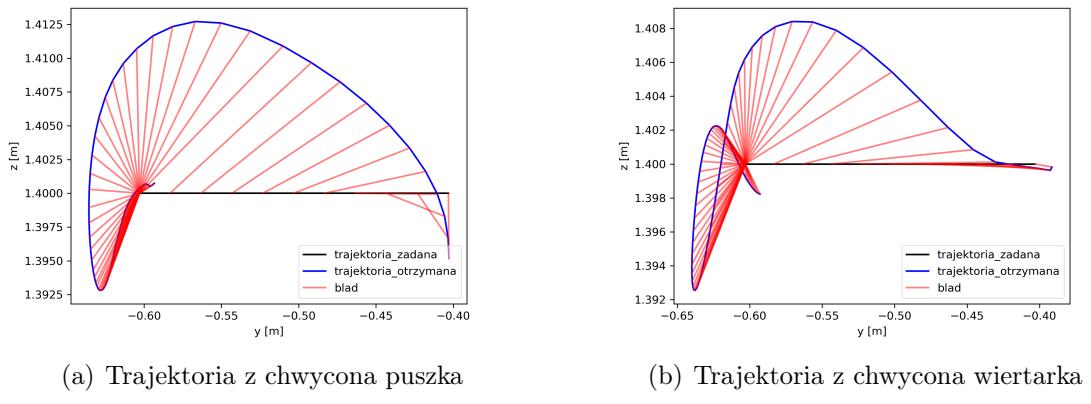
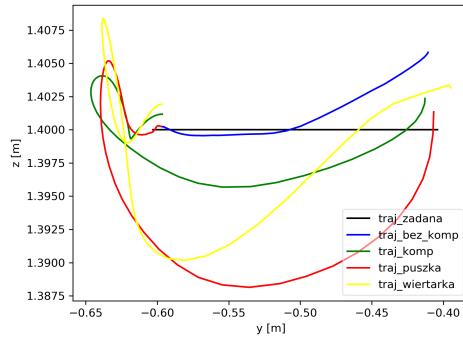
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



Rysunek 6.16: Ruch do gory. Porownanie trajektorii katow w notacji Eulera w zaleznosci od czasu.



Rysunek 6.17: Porownanie trajektorii chwytaka w osiach Y i Z

Rysunek 6.18: Porownanie trajektorii chwytaka w osiach Y i Z 

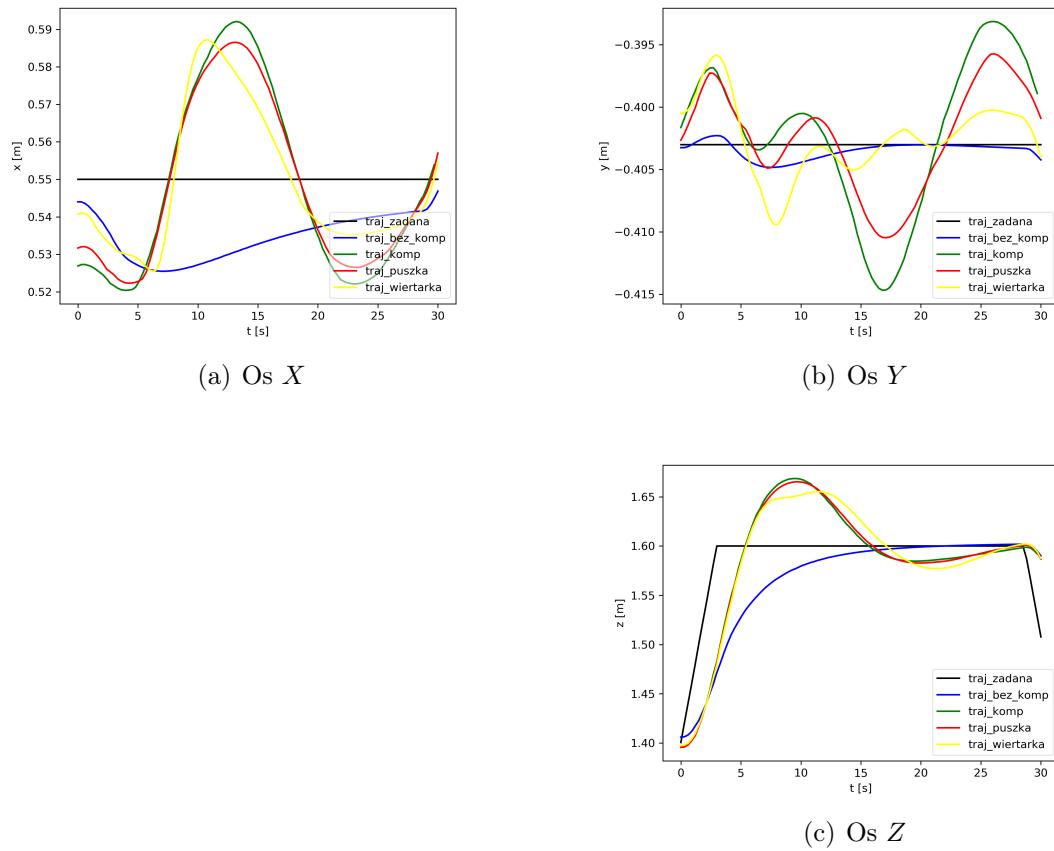
(a) Rzut na wprost

Rysunek 6.19: Porownanie wszystkich trajektorii.

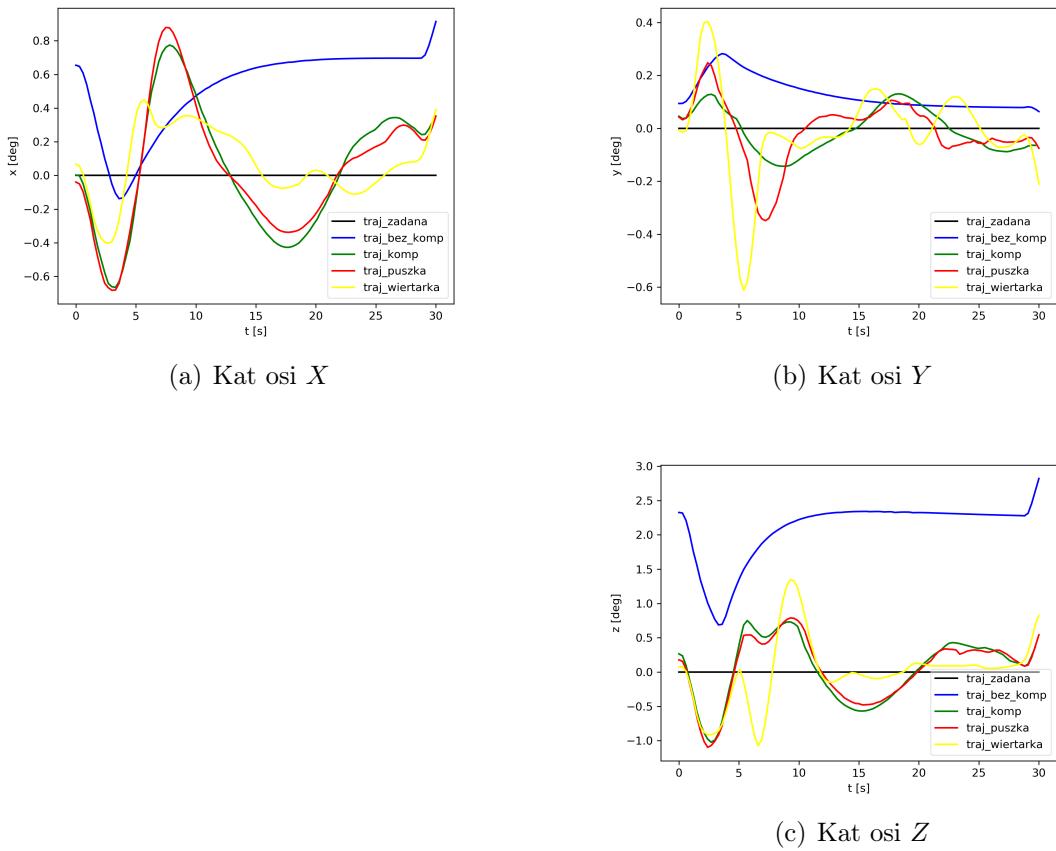
6.1.4 Ruch do gory

Eksperyment ma przetestowac zachowanie algorytmu kompensacji przy ruchu koncowki do gory (rys. 6.20, 6.21). Jest to dla algorytmu potencjalnie trudny ruch z przez sile grawitacji dzialajaca wlasnie w tej osi. Trajektoria ruchu w rzucie APE zostala zaprezentowana na rys. 6.22, 6.23 i 6.24(a).

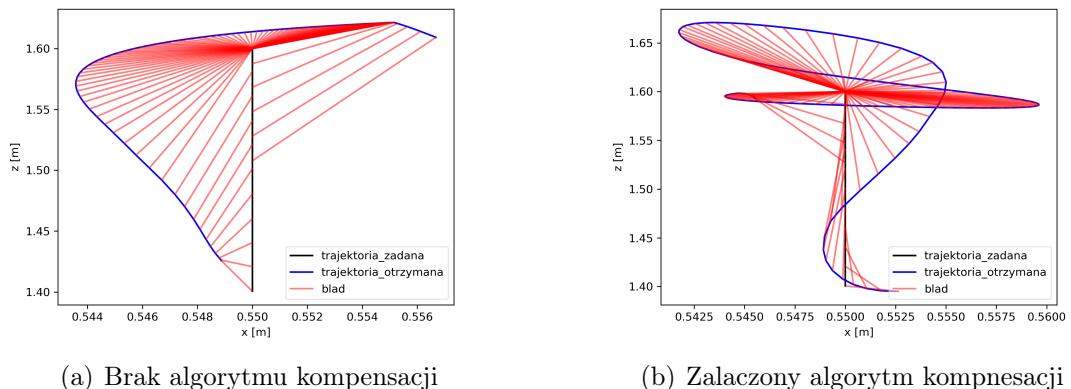
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



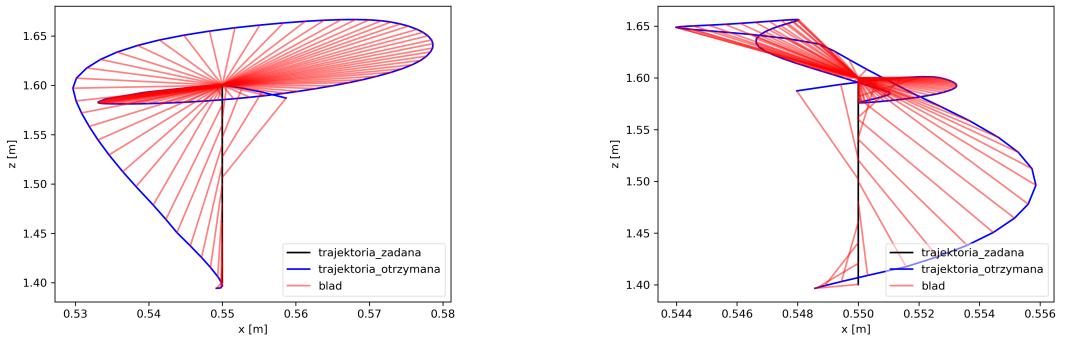
Rysunek 6.20: Ruch do gory. Porownanie trajektorii pozycji w zaleznosci od czasu.



Rysunek 6.21: Ruch do gory. Porownanie trajektorii katow w notacji Eulera w zależnosci od czasu.



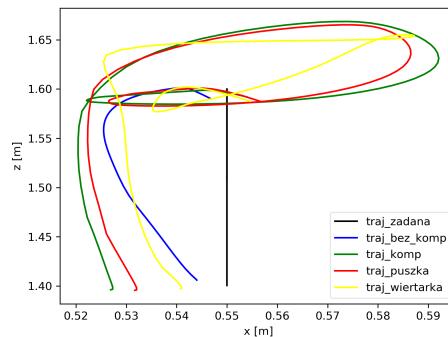
Rysunek 6.22: Porownanie trajektorii chwytaka w osiach X i Z



(a) Trajektoria z chwyconą puszka

(b) Trajektoria z chwyconą wiertarka

Rysunek 6.23: Porownanie trajektorii chwytaka w osiach X i Z

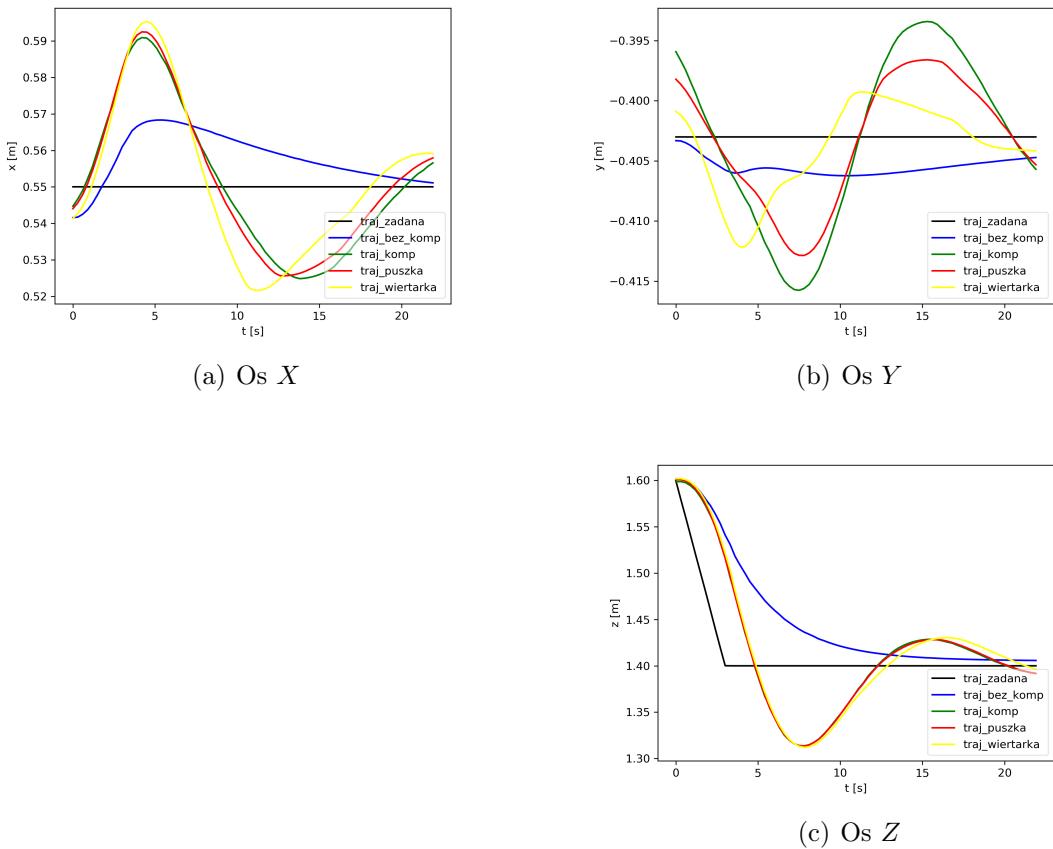


(a) Rzut na wprost

Rysunek 6.24: Porownanie wszystkich trajektorii.

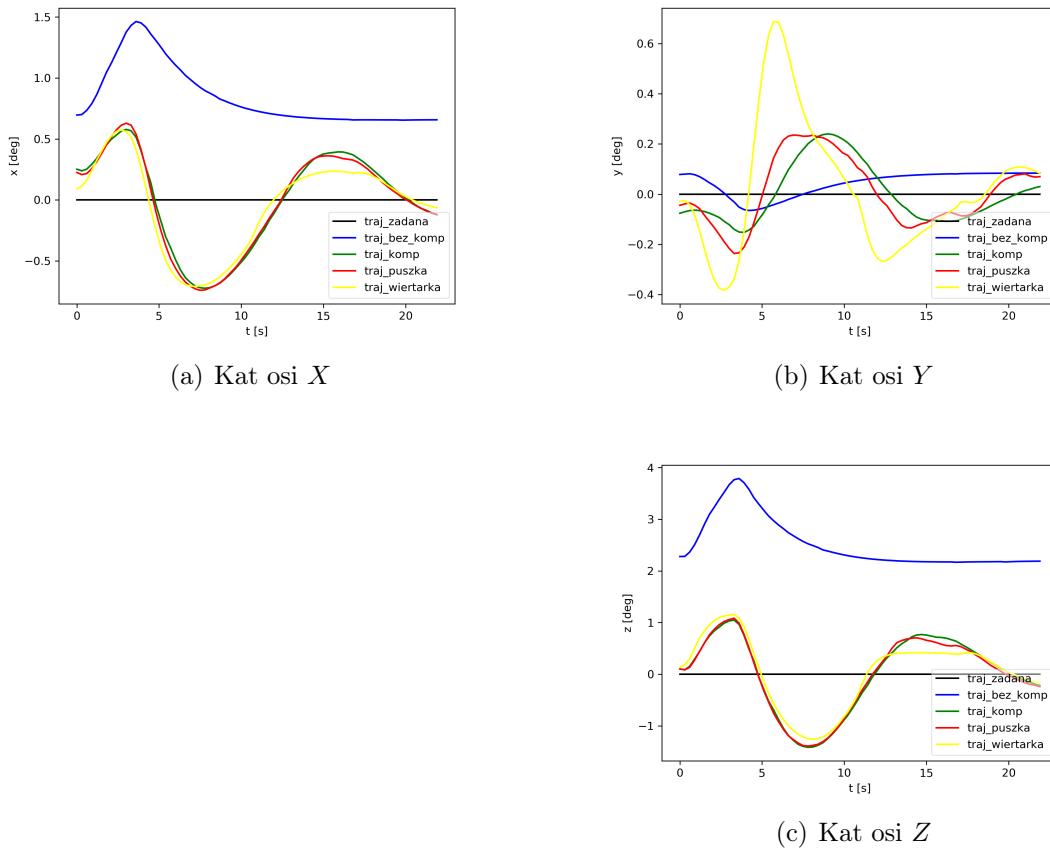
6.1.5 Ruch do dolu

Eksperyment ma przetestowac zachowanie algorytmu kompensacji przy ruchu koncowki do dolu (rys. 6.25, 6.26). Podobnie jak przy eksperymencie ruchu do gory warty uwagi jest aspekt dzialania sily grawitacji w tej samej co ruch osi. Trajektoria ruchu w rzucie APE na wprost ruchu zostala zaprezentowana na rys. ??, 6.28 i 6.29(a).

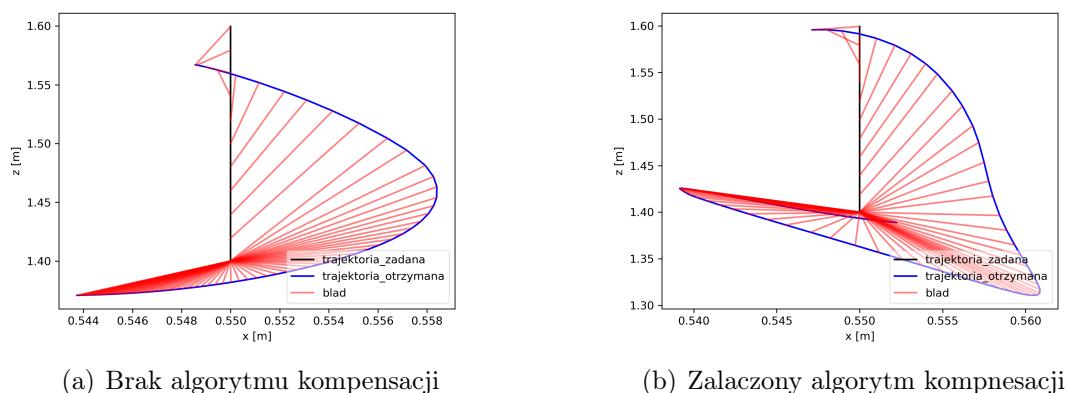


Rysunek 6.25: Ruch do dolu. Porównanie trajektorii pozycji w zaleznosci od czasu.

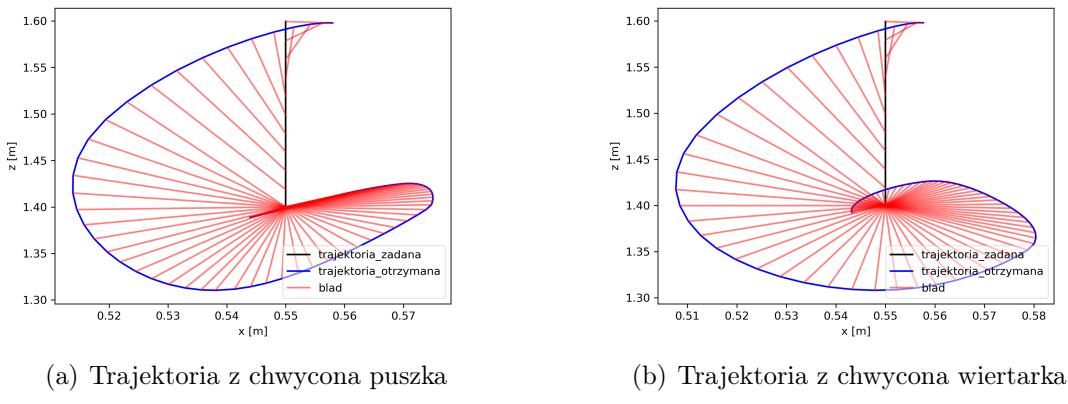
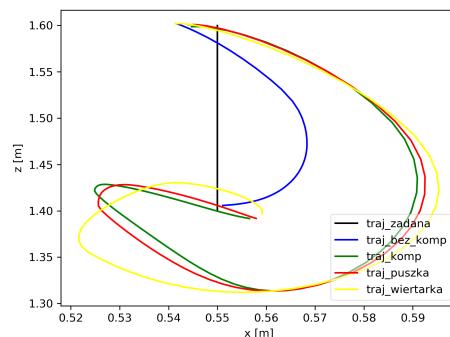
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



Rysunek 6.26: Ruch do dolu. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.



Rysunek 6.27: Porównanie trajektorii chwyptaka w osiach X i Z

Rysunek 6.28: Porównanie trajektorii chwytaka w osiach X i Z 

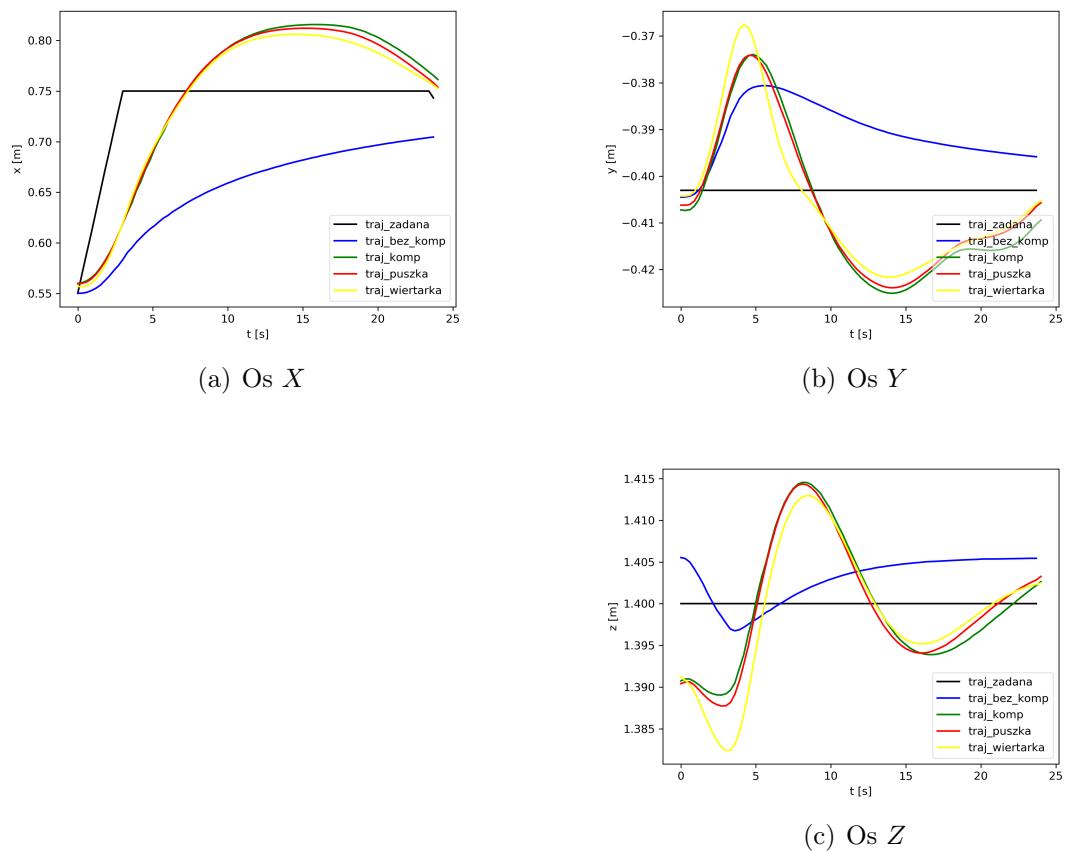
(a) Rzut na wprost

Rysunek 6.29: Porowanie wszystkich trajektorii.

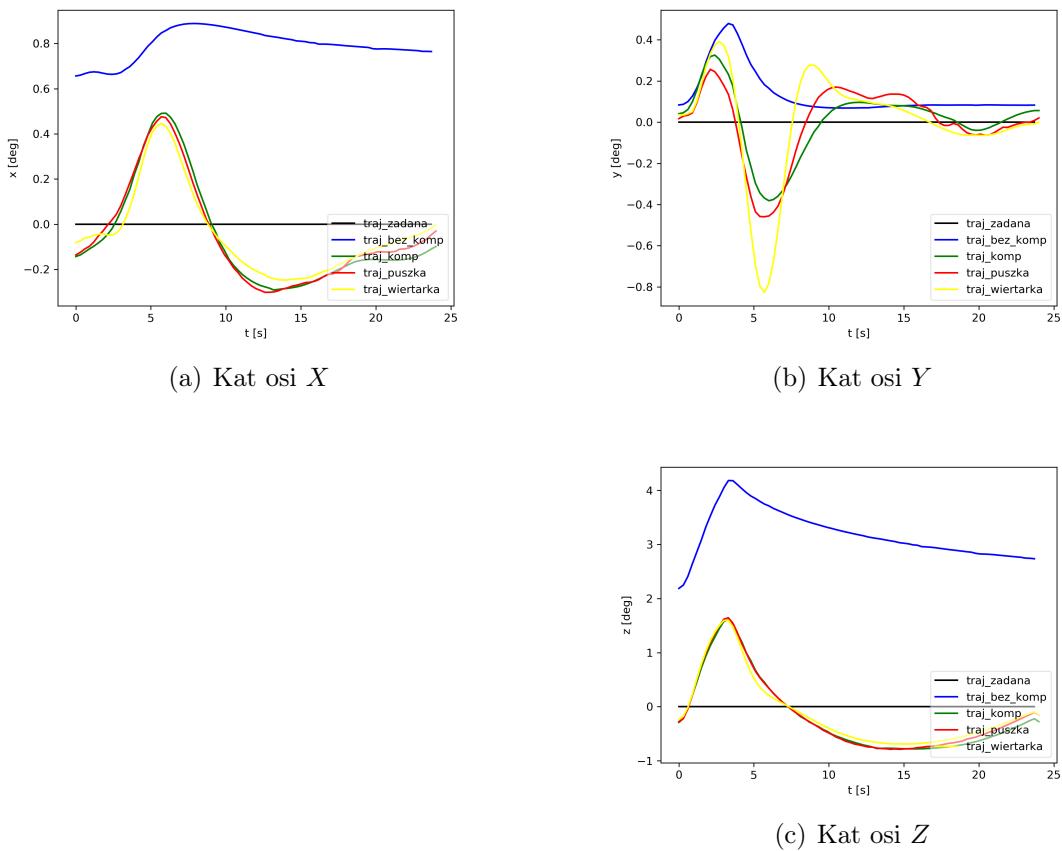
6.1.6 Ruch do przodu

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy ruchu koncowki w kierunku od robota (rys. 6.30, 6.31). Trajektoria ruchu w rzucie na wprost ruchu została zaprezentowana na rys. 6.32, 6.33 i 6.36(a). Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.34, 6.35 i 6.36(b).

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

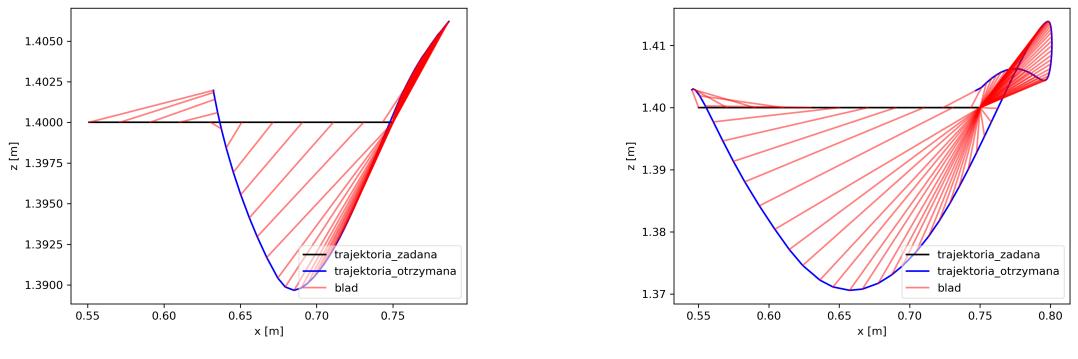


Rysunek 6.30: Ruch do przodu. Porównanie trajektorii pozycji w zależności od czasu.

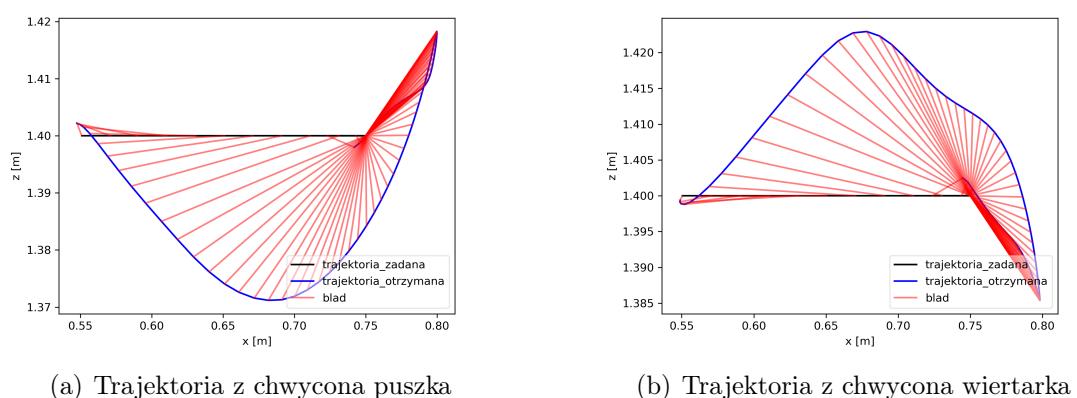


Rysunek 6.31: Ruch do przodu. Porównanie trajektorii katow w notacji Eulera w zaleznosci od czasu.

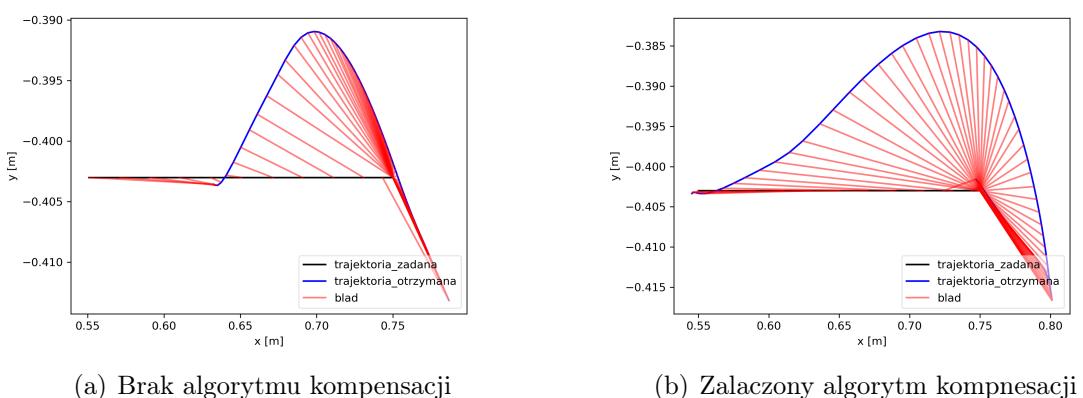
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



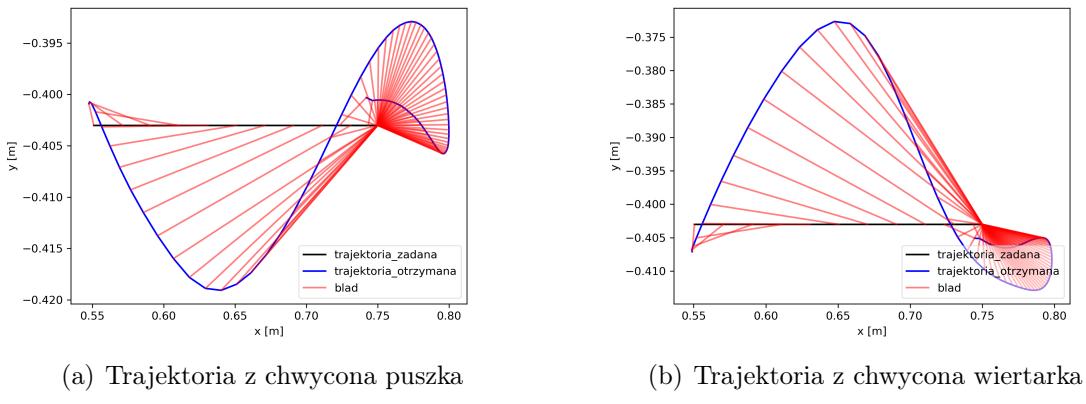
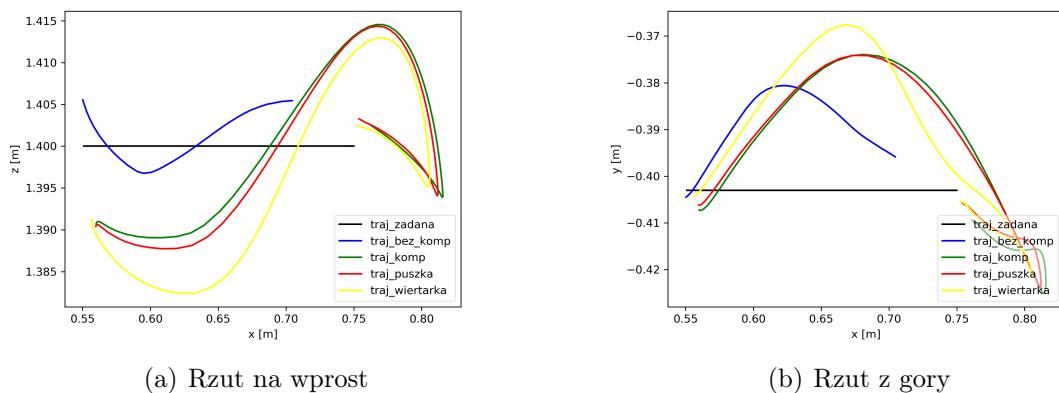
Rysunek 6.32: Ruch do przodu. Porównanie trajektorii chwytaka w osiach X i Z



Rysunek 6.33: Ruch do przodu. Porównanie trajektorii chwytaka w osiach X i Z



Rysunek 6.34: Porównanie trajektorii chwytaka w osiach X i Y

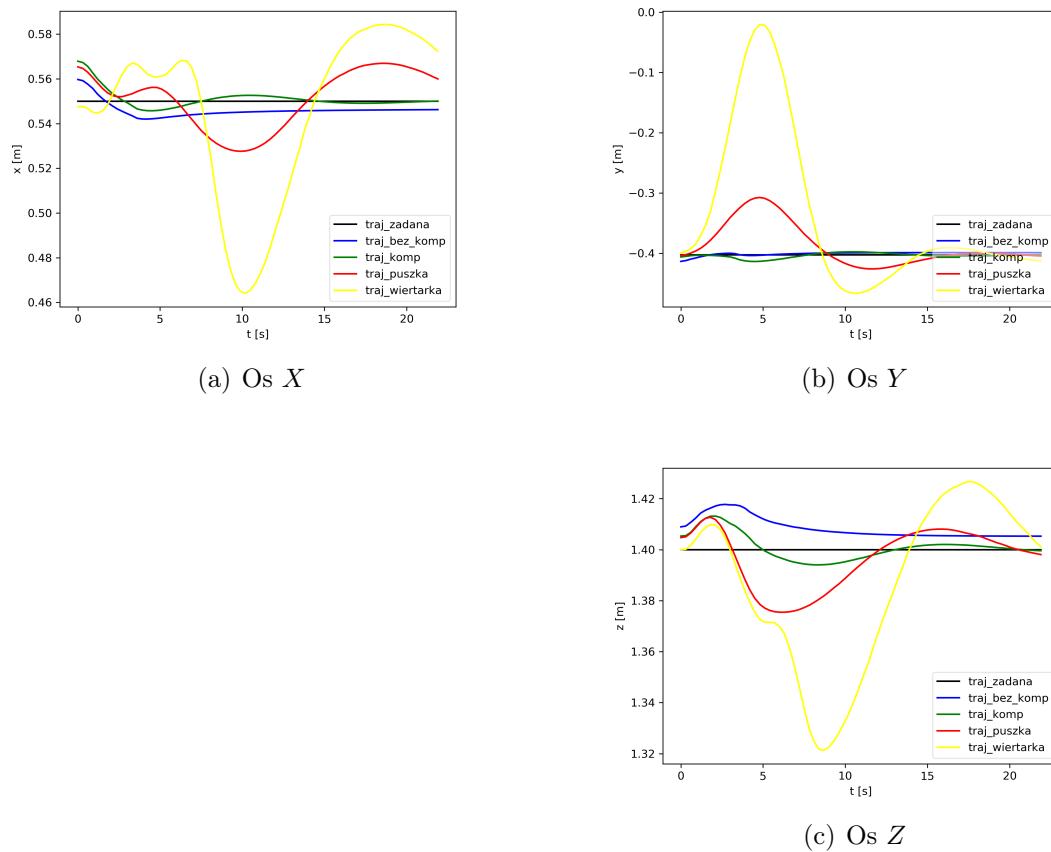
Rysunek 6.35: Porownanie trajektorii chwytaka w osiach X i Y 

Rysunek 6.36: Porowanie wszystkich trajektorii bez zaznaczonego bledu.

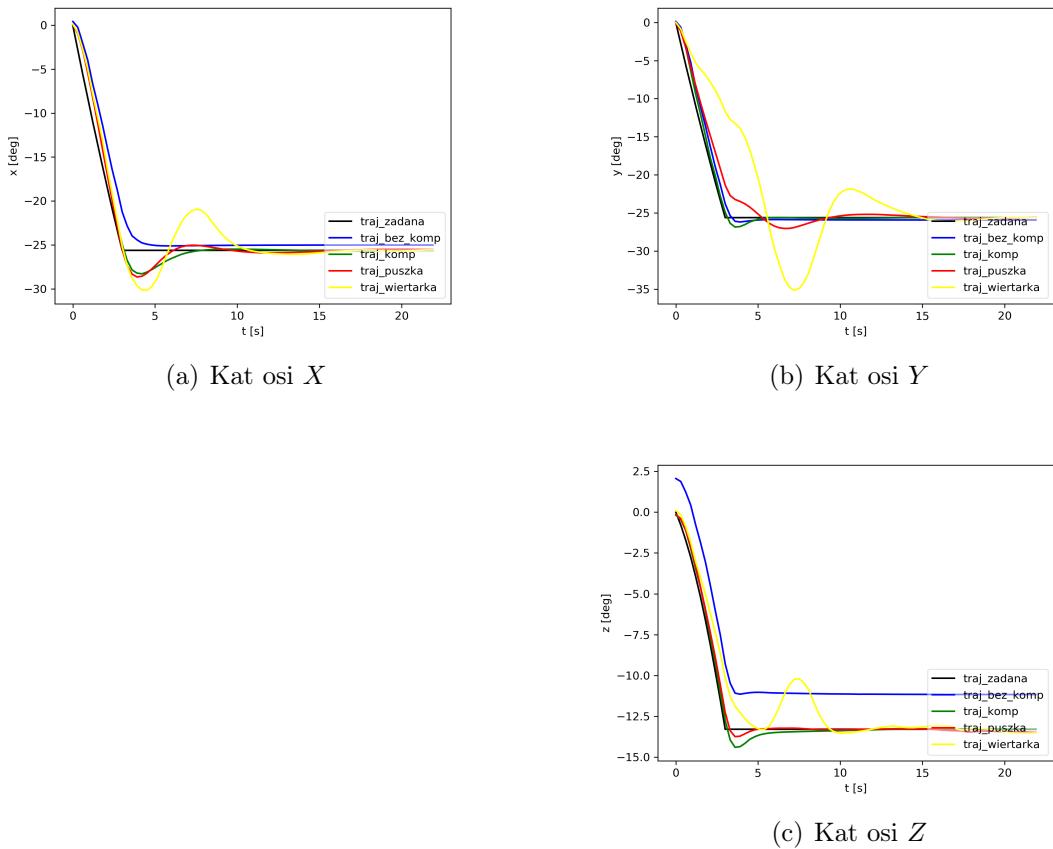
6.1.7 Obrot koncowki

Eksperyment ma przetestować zachowanie algorytmu kompensacji przy obrocie koncowki. Obserwacje polegają na zmianie położen katowych koncowki bez zmiany pozycji. Koncowka ma obrócić się o zadany kąt we wszystkich osiach (rys. 6.37, 6.38). Trajektoria ruchu (w osiach X oraz Y) została zaprezentowana na rys. 6.39, 6.40 i 6.43(a). Trajektoria widoczna z boku (w osiach X oraz Z) została zaprezentowana na rys. 6.41, 6.44 i 6.43(b).

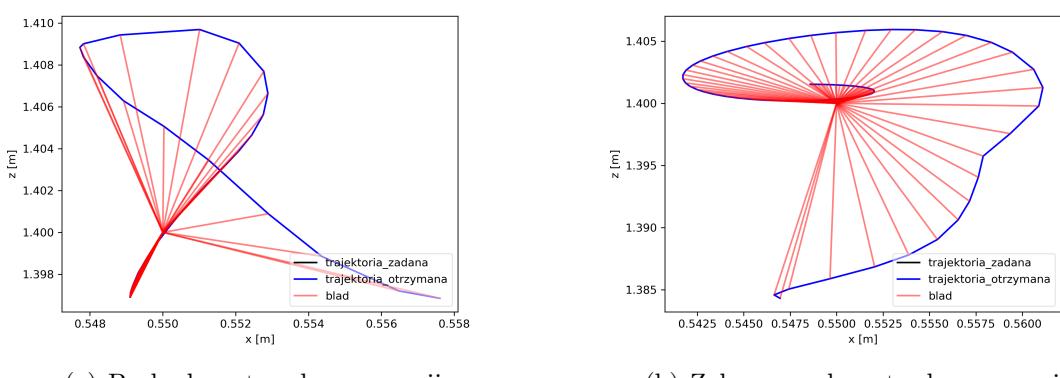
ROZDZIAŁ 6. DZIAŁANIE SYSTEMU



Rysunek 6.37: Porównanie trajektorii pozycji w zależności od czasu.

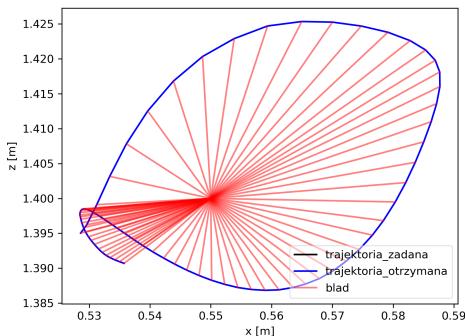


Rysunek 6.38: Porównanie trajektorii katow w notacji Eulera w zaleznosci od czasu.

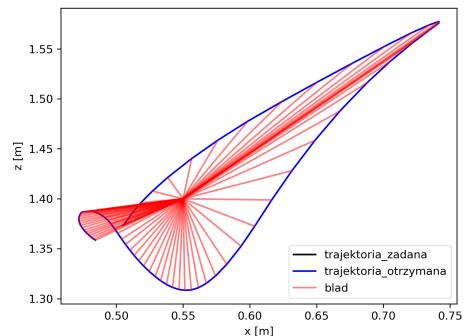


Rysunek 6.39: Porównanie trajektorii chwytaka w osiach X i Z

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

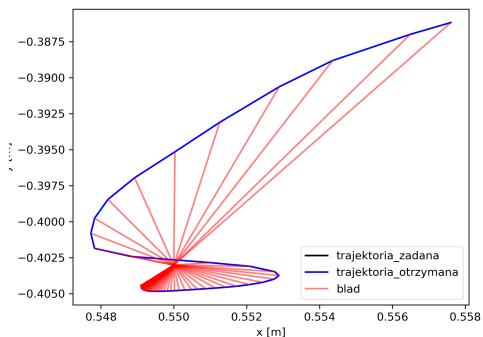


(a) Trajektoria z chwycona puszka

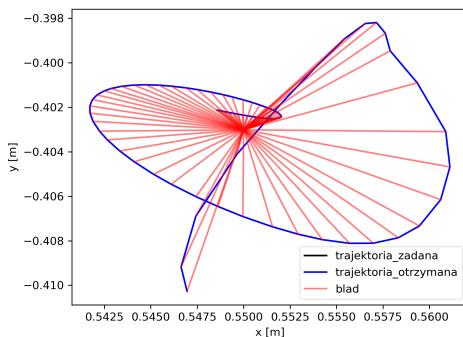


(b) Trajektoria z chwycona wiertarka

Rysunek 6.40: Porównanie trajektorii chwytaka w osiach X i Z

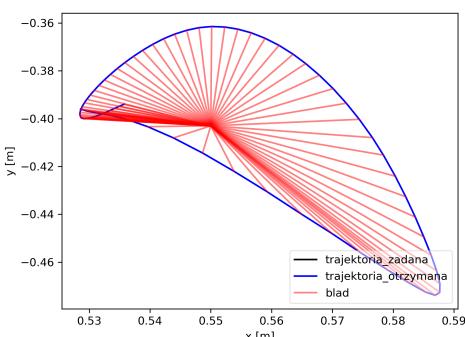


(a) Brak algorytmu kompensacji

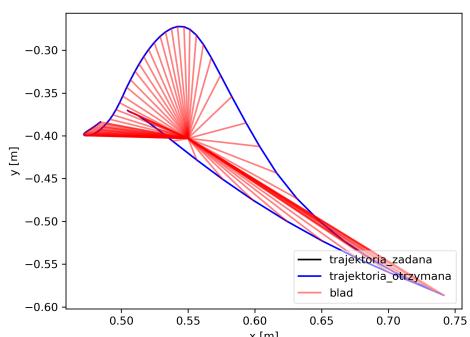


(b) Zalaczony algorytm kompnesacji

Rysunek 6.41: Porównanie trajektorii chwytaka w osiach X i Y

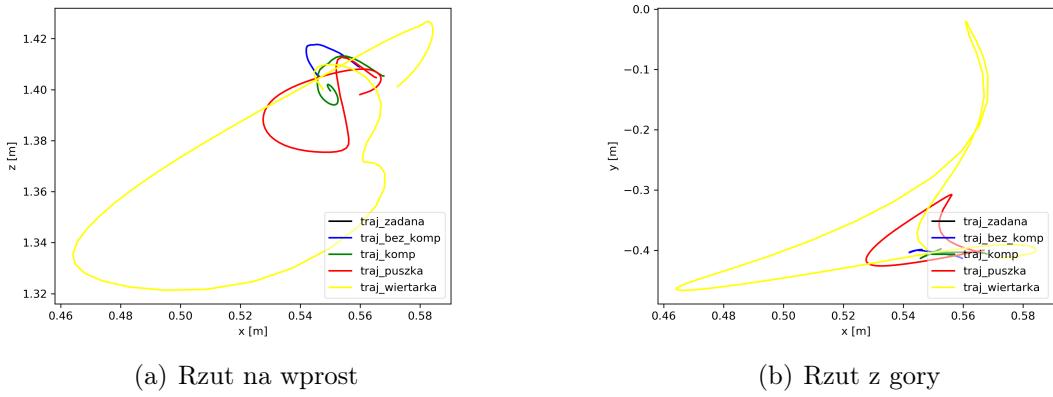


(a) Trajektoria z chwycona puszka

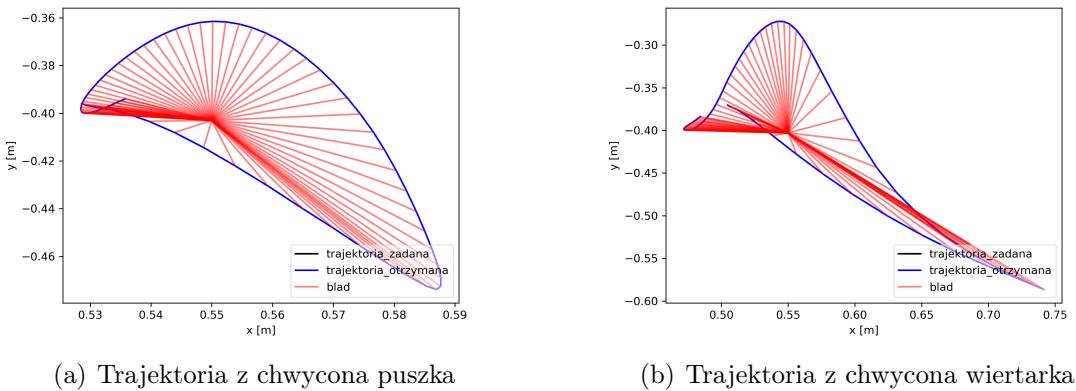


(b) Trajektoria z chwycona wiertarka

Rysunek 6.42: Porównanie trajektorii chwytaka w osiach X i Y



Rysunek 6.43: Porowanie wszystkich trajektorii bez zaznaczonego bledu.

Rysunek 6.44: Porownanie trajektorii chwytyaka w osiach X i Y

6.1.8 Ocena jakosci kolizji

Ocena jakosci kolizji nastepuje poprzez przylozenie skokowych momentow do stawow ramienia oraz obserwacji zachowania robota. Ocena jakosci uginania robota pod wplywem symulowanej kolizji nastepuje zgodnie z metodą opisana w sekcji ??.

6.1.9 Podsumowanie

ROZDZIAŁ 6. DZIAŁANIE SYSTEMU

Rozdział 7

Podsumowanie

7.1 Wnioski

7.2 Perspektywy rozwoju

ROZDZIAŁ 7. PODSUMOWANIE

Bibliografia

- [1] Kuka lwr4+ datasheet. https://www.kukakore.com/wp-content/uploads/2012/07/KUKA_LBR4plus_ENLISCH.pdf.
- [2] Oficjalna strona projektu dart. <https://dartsim.github.io/>.
- [3] Oficjalna strona systemu orocos. <http://orocos.org/>.
- [4] Oficjalna strona systemu ROS. <http://www.ros.org/>.
- [5] Oficjalna strona Zakładu Programowania Robotów i Systemów Rozpoznających. <https://www.robotyka.ia.pw.edu.pl/>.
- [6] 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.
- [7] Berthold Horn, Hugh Hilden, Shahriar Negahdaripour. Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices.
- [8] Tavakoli Saeed, Griffin Ian, Peter J. Fleming. Tuning of decentralised pi (pid) controllers for tito processes. *Control Engineering Practice*, strony 1069–1080. IEEE, 2006.
- [9] 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.
- [10] 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.
- [11] Maciej Stefańczyk, Michał Wałęcki, Konrad Banachowicz, Tomasz Winiarski. Robot usługowy Velma – projekt i konstrukcja głowy. *XIII Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 2, strony 451–460, 2014.

BIBLIOGRAFIA

- [12] 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.
- [13] Jie Tan, Kristin Siu, Karen Liu. Contact Handling for Articulated Rigid Bodies Using LCP.
- [14] Nie ZY. Wang QG. PID Control for MIMO Processes. Vissioli A. Vilanova R., redaktor, *PID Control in the Third Millennium. Advances in Industrial Control.*, strony 177–204. IEEE, 2012.
- [15] 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].
- [16] Tomasz Winiarski, Konrad Banachowicz. Sterowanie redundantnym systemem dwuramiennym z aktywnym korpusem. *XIII Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 2, strony 433–442, 2014.
- [17] Tomasz Winiarski, Dawid Seredyński. Wizualizacja sterowników robotów bazujących na teorii agenta upostaciowanego. *XV Krajowa Konferencja Robotyki – Postępy robotyki*, wolumen 1, strony 417–426, 2018.
- [18] Cezary Zieliński. Architektury systemów robotycznych tworzone z agentów upostaciowionych. 196:379–394, 2018. Robotic System Architectures Based on Embodied Agents (in Polish) - Zielinski.
- [19] 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.

Spis rysunków

2.1	Rysunek przedstawia trajektorie zrzutowane na rysunek 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
6.1	Ruch osemkowy. Porównanie trajektorii pozycji w zależności od czasu.	34
6.2	Ruch osemkowy. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.	35
6.3	Porównanie trajektorii chwytaka w osiach Y i Z	35
6.4	Porównanie trajektorii chwytaka w osiach Y i Z	36
6.5	Porównanie trajektorii chwytaka w osiach X i Z	36
6.6	Porównanie trajektorii chwytaka w osiach X i Z	36
6.7	Porowanie wszystkich trajektorii bez zaznaczonego bledu.	37
6.8	Ruch osemkowy. Porównanie trajektorii pozycji w zależności od czasu.	37
6.9	Ruch osemkowy. Porównanie trajektorii katów w notacji Eulera w zależności od czasu.	38
6.10	Porownanie trajektorii chwytaka w osiach Y i Z	38
6.11	Porownanie trajektorii chwytaka w osiach Y i Z	39
6.12	Porownanie trajektorii chwytaka w osiach X i Z	39
6.13	Porownanie trajektorii chwytaka w osiach X i Z	39
6.14	Porowanie wszystkich trajektorii bez zaznaczonego bledu.	40
6.15	Ruch do gory. Porownanie trajektorii pozycji w zależności od czasu.	40
6.16	Ruch do gory. Porownanie trajektorii katów w notacji Eulera w zależności od czasu.	41
6.17	Porownanie trajektorii chwytaka w osiach Y i Z	41
6.18	Porownanie trajektorii chwytaka w osiach Y i Z	42
6.19	Porowanie wszystkich trajektorii.	42
6.20	Ruch do gory. Porownanie trajektorii pozycji w zależności od czasu.	43
6.21	Ruch do gory. Porownanie trajektorii katów w notacji Eulera w zależności od czasu.	44
6.22	Porownanie trajektorii chwytaka w osiach X i Z	44

SPIS RYSUNKÓW

6.23 Porownanie trajektorii chwytaka w osiach X i Z	45
6.24 Porowanie wszystkich trajektorii.	45
6.25 Ruch do dolu. Porownanie trajektorii pozycji w zaleznosci od czasu. .	46
6.26 Ruch do dolu. Porownanie trajektorii katow w notacji Eulera w zale- znosci od czasu.	47
6.27 Porownanie trajektorii chwytaka w osiach X i Z	47
6.28 Porownanie trajektorii chwytaka w osiach X i Z	48
6.29 Porowanie wszystkich trajektorii.	48
6.30 Ruch do przodu. Porownanie trajektorii pozycji w zaleznosci od czasu.	49
6.31 Ruch do przodu. Porownanie trajektorii katow w notacji Eulera w zaleznosci od czasu.	50
6.32 Ruch do przodu. Porownanie trajektorii chwytaka w osiach X i Z . .	50
6.33 Ruch do przodu. Porownanie trajektorii chwytaka w osiach X i Z . .	51
6.34 Porownanie trajektorii chwytaka w osiach X i Y	51
6.35 Porownanie trajektorii chwytaka w osiach X i Y	51
6.36 Porowanie wszystkich trajektorii bez zaznaczonego bledu.	52
6.37 Porownanie trajektorii pozycji w zaleznosci od czasu.	52
6.38 Porownanie trajektorii katow w notacji Eulera w zaleznosci od czasu.	53
6.39 Porownanie trajektorii chwytaka w osiach X i Z	53
6.40 Porownanie trajektorii chwytaka w osiach X i Z	54
6.41 Porownanie trajektorii chwytaka w osiach X i Y	54
6.42 Porownanie trajektorii chwytaka w osiach X i Y	54
6.43 Porowanie wszystkich trajektorii bez zaznaczonego bledu.	55
6.44 Porownanie trajektorii chwytaka w osiach X i Y	55

Spis tablic