



**inż. Piotr Kornaszewski**

nr albumu: 38650

kierunek studiów: Automatyka i Robotyka

specjalność: systemy sterowania procesami przemysłowymi

forma studiów: studia stacjonarne

**METODY GENERACJI I REALIZACJI TRAJEKTORII RUCHU ROBOTÓW  
MOBILNYCH**

**METHODS OF GENERATION AND IMPLEMENTATION OF THE TRAJECTORY  
OF MOBILE ROBOTS MOVEMENT**

Praca dyplomowa magisterska

napisana pod kierunkiem:

**dr. hab. inż. Pawła Dworaka, prof. ZUT**

Katedra Automatyki i Robotyki

Data wydania tematu pracy: 19.11.2020 r.

Data dopuszczenia pracy do egzaminu: .....

Szczecin, 2021

Piotr Kornaszewski

## **OŚWIADCZENIE AUTORA PRACY DYPLOMOWEJ**

Oświadczam, że praca dyplomowa magisterska pn.

„Metody generacji i realizacji trajektorii ruchu robotów mobilnych”

napisana pod kierunkiem:

dr. hab. inż. Pawła Dworaka, prof. ZUT

jest w całości moim samodzielnym autorskim opracowaniem, sporządzonym przy wykorzystaniu wykazanej w pracy literatury przedmiotu i materiałów źródłowych.

Złożona w Dziekanacie Wydziału Elektrycznego treść mojej pracy dyplomowej w formie elektronicznej jest zgodna z treścią w formie pisemnej.

Oświadczam ponadto, że złożona w Dziekanacie praca dyplomowa, ani jej fragmenty nie były wcześniej przedmiotem procedur procesu dyplomowania związanych z uzyskaniem tytułu zawodowego w uczelniach wyższych.

.....  
podpis dyplomanta

Szczecin, dn. ..... r.

## **Streszczenie pracy**

W bieżącej pracy dokonano przeglądu różnych metod generowania trajektorii robotów mobilnych pod kątem ich własności oraz przykłady ich stosowania. Został również opracowany model kinematyczny trójkołowego robota mobilnego na podstawie prawdziwego robota zbudowanego w ramach pracy inżynierskiej pod tytułem „*Budowa i sterowanie ruchem trójkołowego robota mobilnego*” przez autora omawianej pracy. Na potrzeby pracy został również stworzony układ regulacji bazujący na modelu odwrotnym wraz z analizą teoretyczną jak i również praktyczną zastosowanego rozwiązania. W ramach pracy zostały przeanalizowane teoretycznie jak i również praktycznie wybrane metody generowania trajektorii robotów mobilnych z wykorzystaniem omawianego wcześniej trójkołowego robota mobilnego. Przeprowadzone zostały symulacyjne badania mające na celu wykazać własności każdej z wybranych metod oraz oszacować przydatność w późniejszej implementacji. Na koniec wybrana została konkretna metoda generowania trajektorii na podstawie wcześniejszych badań oraz podjęto próbę jej implementacji do fizycznego obiektu.

## **Słowa kluczowe**

trajektoria, robot, mobilny, generowanie, symulacja

## **Abstract**

In the current work, various methods of generating the trajectory of mobile robots in terms of their properties and examples of their use has been reviewed. A kinematic model of a three-wheeled mobile robot was also developed based on real robot a real robot built as a part of an engineering work under the title „*Budowa i sterowanie ruchem trójkołowego robota mobilnego*” by the author of the discussed work. For the needs of the work, a control system based on the inverse model was also created, along with a theoretical, as well as a practical analysis of the solution used. As part of the work, theoretically, as well as practically selected methods of generating the trajectory of mobile robots using the previously discussed three-wheeled mobile robot were analyzed. Simulation studies were carried out to demonstrate the properties of each of the selected methods and to estimate their usefulness in later implementation. Finally, a specific method of generating the trajectory was selected based on previous research, and an attempt was made to implement it into a physical object.

## **Keywords**

trajectory, robot, mobile, generating, simulation

# Spis treści

<b>Wykaz ważniejszych oznaczeń i skrótów</b>	6
<b>Wprowadzenie</b>	7
<b>1. Generowanie trajektorii ruchu dla robotów mobilnych</b>	11
1.1. Metody lokalne generowania trajektorii	12
1.2. Metody globalne generowania trajektorii	13
1.3. Metody mieszane generowania trajektorii	14
<b>2. Model oraz sposób sterowania trójkątowym robotem mobilnym</b>	15
2.1. Trójkątowy robot mobilny	15
2.2. Obiekt	17
2.3. Regulator obiektu	21
<b>3. Wybrane metody generowania trajektorii robotów mobilnych</b>	27
3.1. Teoretyczne zagadnienie generowania trajektorii robota mobilnego	27
3.1.1. Metoda generowania trajektorii płaskiej	27
3.1.2. Metoda generowania trajektorii wielomianowej	29
3.1.3. Metoda generowania trajektorii symetrycznej-wielomianowej	29
3.2. Praktyczna realizacja generowania trajektorii robota mobilnego	30
3.2.1. Trajektoria płaska	30
3.2.2. Trajektoria wielomianowa	33
3.2.3. Trajektoria symetryczna-wielomianowa	34
3.3. Testy symulacyjne	36
3.3.1. Symulacja zadanej trajektorii płaskiej	38
3.3.2. Symulacja zadanej trajektorii wielomianowej	40
3.3.3. Symulacja zadanej trajektorii symetrycznej-wielomianowej	42
3.3.4. Podsumowanie badań symulacyjnych	44
<b>4. Implementacja wybranej metody do fizycznego obiektu</b>	49
4.1. Wybrana metoda generowania trajektorii	49
4.2. Problematyka implementacji wybranej metody generowania trajektorii oraz niezbędne modyfikacje w robocie mobilnym	50
<b>Zakończenie</b>	51
<b>Spis tabel</b>	54
<b>Spis rysunków</b>	55
<b>Spis kodów źródłowych</b>	57

## **Wykaz ważniejszych oznaczeń i skrótów**

$E(s)$	Sygnal błędu
$K(s)$	Sygnal błędu uwzględniający różnicę modelu oraz obiektu
$P$	Środek osi napędowej robota
$P'$	Punkt śledzenia trajektorii robota mobilnego
$U(s)$	Sygnal sterujący
$X(s)$	Sygnal wejściowy
$Y(s)$	Sygnal wyjściowy
$l$	Odległość koła od środka osi napędowej robota
$m$	Odległość koła sterowego od środka osi napędowej robota
$r$	Promień koła robota
$\beta$	Kąt obrotu koła sterowego
$\dot{\varphi}_1, \dot{\varphi}_2, \dot{\varphi}_3$	Prędkości kątowe kół robota
$\theta$	Położenie kątowe robota

# **Wprowadzenie**

Dzisiejsze wymagania dotyczące wydajności procesów technologicznych stale rosną co jest równoznaczne z poszukiwaniem nowych metod optymalizacji procesów lub udoskonalaniem obecnych. Niewątpliwie czynnikiem spowalniającym wydajność w firmach podczas wykonywania żmudnych czy wyczerpujących zadań jest czynnik ludzki. Właśnie z tych względów podejmowane są próby odciążenia ludzi przez automatyzację procesów. Przykładem takiego systemu może być autonomiczny system transportu towarów w magazynach w firmie Amazon, która wykorzystuje w tym celu roboty mobilne. Takie rozwiązanie usprawnia proces oraz odciąża ludzi i pozwala wyeliminować pomyłki związane ze zmęczeniem przy wykonywaniu ciężkiej i monotonnej pracy fizycznej. Innym przykładem może być transport publiczny i stosowanie w nim autonomicznych pojazdów mobilnych. Wdrożenie autonomicznych pojazdów podobnie jak w poprzednim przypadku ma za zadanie odciążyć ludzi przy wykonywaniu monotonnej pracy i zmienić pozycję człowieka z wykonawczej na nadzorującą. Sceptyczym ludzki dotyczący autonomicznych pojazdów z roku na rok jest coraz mniejszy, w związku z tym następuje rozwój branży motoryzacyjnej w tym kierunku. Dzięki stosowaniu nowych rozwiązań i badań nad rozwojem pojazdy te stają się bezpieczniejsze i wyręczają kierującego, nie kiedy można natrafić w internecie na informacje dotyczące uniknięcia wypadku przez autonomiczny pojazd co znacząco ociepla stosunek ludzki do autonomicznych pojazdów. Elementem łączącym omawiane przypadki jest ich sposób poruszania, aściślej mówiąc sposób generowania ich trajektorii, który ma wpływ na precyzję wykonywanych ruchów. Zarówno w przypadku robota mobilnego, pracującego w magazynie jak i autonomicznego samochodu poruszającego się po ulicy musi zostać wygenerowana konkretna ścieżka zakładająca konkretne położenie, prędkość, przyspieszenie czy kąt obrotu. Te zadania realizowane są przez generator trajektorii. Istnieją różne rodzaje generatorów, w zależności czy otoczenie robota zmienia się dynamicznie lub jest statyczne, czy mamy pełną wiedzę o otoczeniu lub tylko częściową i w związku z tym dobiera się konkretny generator spełniający wymagania jakie są stawiane.

Podstawowym celem pracy była implementacja wybranej metody generowania trajektorii w robocie mobilnym. Aby tego dokonać należało dokonać przeglądu różnych metod, zweryfikować je symulacyjnie a następnie najlepszą z metod zaimplementować do robota. W ramach pracy dokonano przeglądu różnych metod generowania trajektorii robotów mobilnych, został również opracowany model kinematyczny trójkątowego robota mobilnego na podstawie prawdziwego robota zbudowanego w ramach pracy inżynierskiej pod tytułem „*Budowa i sterowanie ruchem trójkątowego robota mobilnego*” przez autora omawianej pracy. Niezbędne do przeprowadzenia symulacji okazało się zbudowanie układu regulacji bazującego na modelu odwrotnym, wraz z analizą teoretyczną jak i również praktyczną zastosowanego rozwiązania. Skupiając się na sednie problemu zostały przeanalizowane teoretycznie jak również praktycznie wybrane metody generowania trajektorii robotów mobilnych. Przeprowadzone zostały symulacyjne badania mające na celu zweryfikować własności każdej z wybranych metod oraz oszacować przydatność w późniejszej implementacji. Na koniec opierając

się o wyniki uzyskanych badań podczas symulacji wybrana została konkretna metoda generowania trajektorii oraz podjęto próbę jej implementacji do fizycznego obiektu.

Piotr Kornaszewski

## **Cel pracy**

Celem pracy było przeprowadzenie badań różnych metod generacji i realizacji trajektorii ruchu robotów mobilnych. Głównymi zadaniami pracy były: stworzenie modelu kinematycznego robota, przeanalizowanie teoretyczne jak i praktyczne wybranych metod generowania trajektorii robotów mobilnych oraz implementacja wybranej metody do robota mobilnego. Otrzymane wyniki badań miały przyczynić się do rozwoju prac nad tworzeniem wielofunkcyjnego robota mobilnego mającego za zadanie eksplorowanie pomieszczeń w celu gromadzenia informacji o otoczeniu.

## **Zakres pracy**

Zakres pracy obejmował przegląd różnych metod generowania trajektorii robotów mobilnych. Należało opracować teoretyczny model wybranego robota mobilnego oraz dokonać analizy teoretycznej i praktycznej właściwości wybranych metod generacji trajektorii ruchu. W części praktycznej należało dokonać symulacyjnych badań porównawczych wybranych metod oraz ich realizacji dla opracowanego modelu robota mobilnego. Na zakończenie pracy należało dokonać implementacji jednej z metod w robocie mobilnym.

Piotr Kornaszewski

## **ROZDZIAŁ 1**

# **Generowanie trajektorii ruchu dla robotów mobilnych**

Każdy robot mobilny niezależnie od rodzaju czy przeznaczenia musi dokładnie wykonywać powierzone zadania, w związku z tym precyza jego poruszania jest niezbędna. Kluczowym elementem w realizacji tego zadania jest generator trajektorii ruchu robota. Zadaniem generatorów trajektorii jest doprowadzenie z położenia początkowego do położenia końcowego naszego robota po przez wyznaczenie trasy, z uwzględnieniem różnych czynników (prędkość, przyspieszenie, energia , itp.).

Zapewnienie precyzji ruchu robota jest jednym z głównych kryteriów jakości jego działania, w tym celu stosuje się różne metody narzucając pewne wymagania. Wymagania te wynikają z ograniczeń otoczenia robota czy też złożoności wykonywanych zadań. Precyzę ruchu robota można uzyskać wykonując konkretną trajektorię tak, aby robot znalazł się dokładnie w wyznaczonym miejscu i pod odpowiednim kątem.

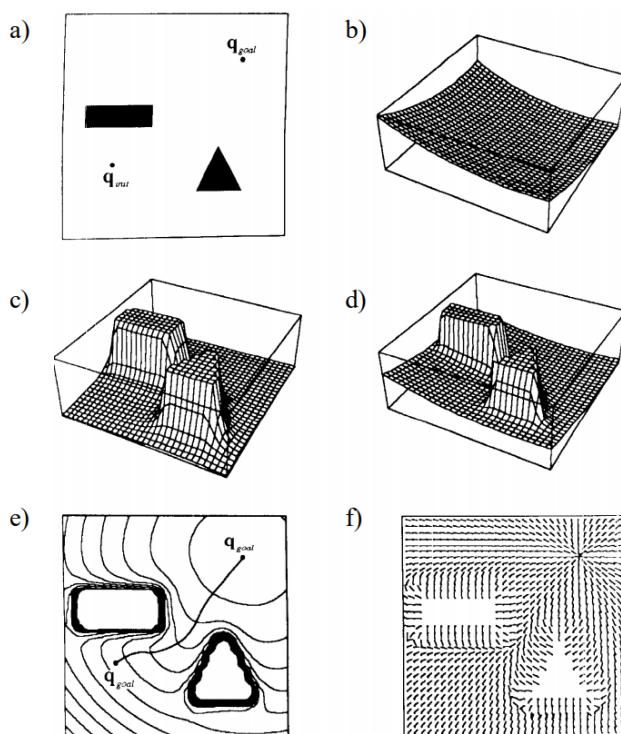
Można rozróżnić dwie zasadnicze kategorie planowania ruchu, pierwszą z nich jest planowanie w czasie rzeczywistym typu „on-line”, drugą typu „off-line”, gdzie cała trasa jest wyliczana a priori. W metodach czasu rzeczywistego wykorzystuje się różne sensory mierzące odległość jak również systemy wizyjne wyposażone w kamery z czujnikiem głębi typu Kinect, sama metoda wymaga sporej mocy obliczeniowej przez co nie zawsze możliwe jest jej zastosowanie (ze względu na koszty czy technologię). Generatory typu „off-line” bazują na uproszczonym modelu matematycznym obiektu oraz środowiska w celu ograniczenia czasu potrzebnego na wykonanie obliczenia, ścieżka wyliczana jest w całości i zostaje wysłana do układu regulacji w celu jej wykonania. Konieczne jest zastosowanie układu ze sprzężeniem zwrotnym w celu śledzenia zaplanowanej trajektorii, aby zapewnić odporność metody na niedokładności modelowania obiektu.

Znajomość środowiska w jakim pracuje robot jest kluczowym elementem budowania układów regulacji, doboru wyspecjalizowanych sensorów czy właśnie dobrania odpowiednich omawianych metod planowania ścieżki, w związku z tym stosuje się różne klasy metod. W podrozdziałach 1.1 i 1.2 zostały przedstawione lokalne rozwiązania bazujące na

ograniczonym polu widzenia oraz metody globalne bazujące na przybliżonej lub pełnej znajomości otoczenia.

## 1.1. Metody lokalne generowania trajektorii

Metody lokalne generowania trajektorii rozpatrują układ robota opisany równaniami kinematyki w obecnym punkcie przestrzeni konfiguracyjnej, planowanie ruchu odbywa się na podstawie zmniejszania odległości robota od punktu docelowego. Metody te wyznaczają trajektorię z bieżącego położenia do kolejnego wyznaczonego położenia i tworzy się w ten sposób rozwiązanie lokalne. Po „sklejeniu” wszystkich fragmentów rozwiązań lokalnych otrzymuje się rozwiązanie globalne, należy pamiętać, iż otrzymane rozwiązanie nie musi być optymalne globalnie, gdyż fragmenty otrzymanej trajektorii były optymalizowane lokalnie.



**Rysunek 1.1.** Zasada działania metod potencjalnych: a) kształt mapy przeszkód,b) potencjał przyciągający do celu, c) potencjał wypadkowy,d) nałożone podpunkty b) i c), e) warstwice pola potencjalnego i znaleziona ścieżka, f) kierunki wektorów gradientu potencjału

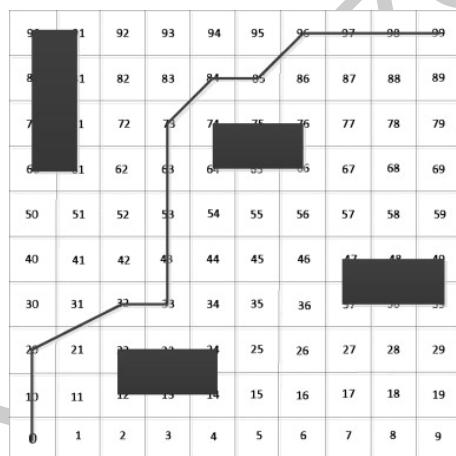
Źródło: Na podstawie [[dynamiczne\\_planowanie](#)]

Rozwiązania tego typu stosowane są w systemach czasu rzeczywistego, wszędzie tam, gdzie kluczowym elementem jest czas reakcji na zmiany w otoczeniu robota wynikających z ruchu innych obiektów czy też poruszania się w nieznanym środowisku. Przykładem takich algorytmów mogą być metody potencjalne, zakłada się w nich, że cel przyciąga robota a przeszkody odpychają go z siłą zależną od odległości dzielącej przeszkodę z robotem. Ruch robota jest wykonywany w kierunku wzdłuż linii powstałego gradientu. Przykładowa zasada działania została przedstawiona na Rysunku 1.1. Metody tego typu

są szybkie i nadają się do wykorzystywania w czasie rzeczywistym, lecz posiadają pewną wadę, mianowicie nie biorą pod uwagę całej przestrzeni robota a co za tym idzie istnieje możliwość dotarcia w ślepy zaułek. W literaturze właśnie z tego względu te metody są krytykowane i są podejmowane próby wyeliminowania ich wad. Próby eliminowania uzyskuje się przez połączenie metod globalnych z lokalnymi, skutkuje to częściowym wyeliminowaniem wad jednej i drugiej metody.

## 1.2. Metody globalne generowania trajektorii

Globalne planowanie trajektorii robota działa w pętli usprawniając każdy kolejny krok za pomocą odpowiedniego wskaźnika jakości, po wyliczeniu całej ścieżki zostaje ona wysłana w całości do wykonania. W tych rozwiązaniach niezbędna jest całkowita wiedza o otoczeniu co nie zawsze jest możliwe. Zaletą metod globalnych jest to, że optymalizują pewną funkcję kosztu, na przykład minimalizacja promienia skrętu robota i odległość od punktu docelowego. Wadą tych rozwiązań jest brak odporności na zmiany w otoczeniu pracy robota oraz duża złożoność obliczeniowa, co utrudnia ich implementację w małych pojazdach posiadających słabe jednostki obliczeniowe. Przykładem metody globalnej



**Rysunek 1.2.** Środowisko robota wraz z wyznaczoną trasą

Źródło: Na podstawie [Qu2013]

może być metoda wykorzystująca optymalizację wielokryterialną z ulepszonym algorytmem genetycznym. Otoczenie robota zostaje podzielone przez prostokąty o równej wielkości i zostają one ponumerowane (rysunek 1.2), w celu możliwości implementacji do algorytmu generowania trajektorii. Stosując algorytm genetyczny oraz dodając do niego odpowiednie funkcje kosztu (w zależności na jakim parametrze zależy nam najbardziej) przy jednoczesnym dodaniu ograniczeń w postaci numerów kwadratów pod przeszkodami, możliwe jest wygenerowanie trajektorii. Po otrzymaniu trasy należy poddać ją wygładzeniu w celu ułatwienia sterowania i zaimplementować do regulatora obiektu. Istotną kwestią jest możliwości generowania trajektorii bez uwzględnienia nieholonomicznych ograniczeń co skutkuje ułatwieniem wyznaczenia optymalnej trasy, lub branie pod uwagę tych ograniczeń co z kolei jest bardziej skomplikowanym procesem obliczeniowym lecz eliminuje niewykonalne ruchy robota. W pierwszym przypadku

układ regulacji robota w celu podążania za wyznaczoną trajektorią musi wygenerować sterowanie uwzględniające ograniczenia, które pozwoli wykonać zaplanowaną trajektorię. Jest to szybsze rozwiązanie w porównaniu z drugim przypadkiem lecz uzyskanie przez robota w całości optymalnej ścieżki może okazać się niemożliwe. Trajektorie wygenerowane z uwzględnionymi ograniczeniami nieholonomicznymi pozwalają uzyskać optymalną ścieżkę jak również ułatwiają znacząco jej wykonanie przez układ regulacji robota.

### **1.3. Metody mieszane generowania trajektorii**

Poruszanie się robotów mobilnych w nieznanym środowisku czy też znanym lecz z dynamicznie występującymi przeszkodami utrudnia skuteczne wykorzystywanie opisywanych metod lokalnych czy metod globalnych. Przykładem problemu metod lokalnych może być ich „krótkowzroczność” co może doprowadzić do sytuacji wykonania ścieżki w ślepy zaułek, z kolei metody globalne nie radzą sobie z przeszkodami występującymi dynamicznie w otoczeniu ze względu na wyliczanie trajektorii a priori, co uniemożliwia szybką reakcję przez złożony proces obliczeniowy.

Aby częściowo wyeliminować wady metod lokalnych oraz metod globalnych łączy się je tworząc metody mieszane. Eliminują one brak odporności na zmienne otoczenie robota jak i również pozwalają na optymalizowanie samej ścieżki. Metody mieszane są coraz bardziej wykorzystywane ze względu na swoje właściwości, wykorzystuje się je w branży motoryzacyjnej (asystent parkowania, autopilot, itp.) oraz w wielu innych. Wszędzie tam liczy się jak najmniejszy koszt eksploatacji jak i również bezpieczeństwo robota i ludzi pracujących w jego otoczeniu. Rozwojowi tych metod sprzyja tworzenie coraz to lepszych jednostek obliczeniowych jak i również niższe koszty czujników potrzebnych do budowy całego układu, również inne aspekty techniczne jak i rozwiązania niewykonalne kiedyś, dzisiaj dostępne przy niewielkim nakładzie finansowym. Niewątpliwie jest to jedna z przyszłościowych metod w tej dziedzinie robotyki.

## **ROZDZIAŁ 2**

# **Model oraz sposób sterowania trójkołowym robotem mobilnym**

W poprzednim rozdziale została omówiona tematyka generowania trajektorii robotów mobilnych, rodzaje robotów oraz przykłady. Niniejszy rozdział został przeznaczony na przedstawienie struktury kinematycznej wybranego robota mobilnego oraz budowy jego układu sterowania, który posłużył w dalszej części pracy do przeprowadzenia analizy poprawności działania poszczególnych metod.

### **2.1. Trójkołowy robot mobilny**

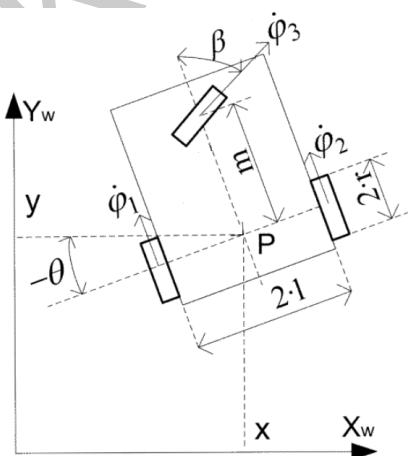
Niewątpliwą inspiracją do powstania tematu omawianej pracy był przygotowany przez autora w ramach pracy inżynierskiej trójkołowy robot mobilny(rysunek 2.1). Robot ten został zbudowany z myślą stworzenia mobilnej platformy pozwalającej na eksplorację korytarz budynku uczelni z wykorzystaniem dodatkowych urządzeń peryferyjnych.

Zagadnienie budowy czy też sterowania robotem mobilnym nie jest niczym nowym w świecie robotyki, jednak unikalność omawianego modelu polega na jego uniwersalności po przez zastosowanie układu trzech niezależnych silników BLDC. Wszystkie koła napędowe jak i koło sterowe mogą być sterowane niezależnie (patrz rysunek 2.2) lub w innej konfiguracji zależnej od sposobu sterowania. Dzięki zastosowaniu odrębnych silników do każdego koła możliwe jest opisywanie jego sposobu poruszania się za pomocą równań kinematycznych w różnym ujęciu, bazując na budowie robota. W dalszej części pracy zostaną szczegółowo omówione warianty równań kinematycznych opisu trójkołowego robota mobilnego. Zastosowanie równań kinematyki pozwoli na przeprowadzenie symulacji z wykorzystaniem metod lokalnych generowania trajektorii, gdzie właśnie omawiane równania są niezbędne.



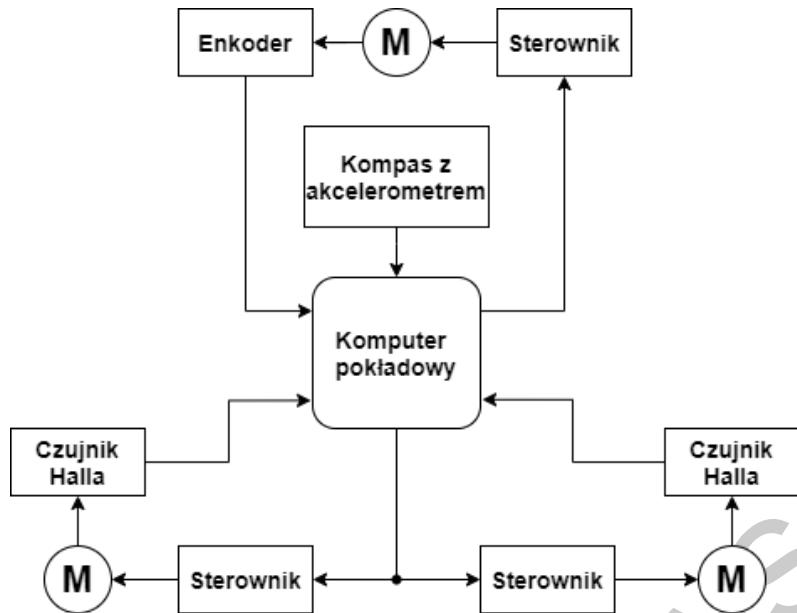
**Rysunek 2.1.** Trójkołowy robot mobilny

Źródło: Opracowanie własne



**Rysunek 2.2.** Trójkołowy robot mobilny z napędem różnicowym

Źródło: Na podstawie [Gracia2002]



**Rysunek 2.3.** Schemat ideowy układu sterowania trójkątowym robotem mobilnym

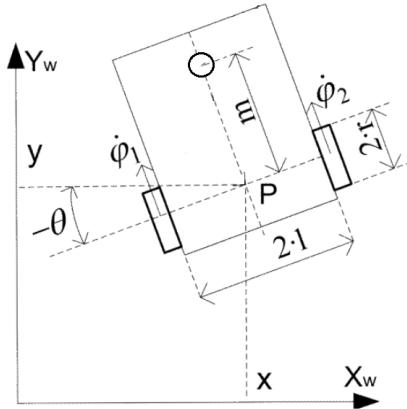
Źródło: Opracowanie własne

Układ regulacji robota wykorzystuje akcelerometr z kompasem magnetycznym, enkoder położenia koła sterowego oraz czujniki Halla siników napędowych (patrz rysunek 2.3). Zbudowany układ regulacji nie jest zaawansowany i nie pozwala na pomiar wszystkich parametrów robota lecz był budowany z myślą o łatwej modyfikacji w celu dołączenia czujników, kamer czy innych urządzeń pomiarowych. Na potrzeby przeprowadzenia symulacji układ regulacji robota jest wystarczający, co zostało przedstawione w kolejnych rozdziałach skupiających się na generowaniu trajektorii. Kolejne podrozdziały szczegółowo przedstawią różne możliwe warianty układów sterowania z wadami oraz zaletami każdego z nich.

## 2.2. Obiekt

Wybór sposobu sterowania, który jest możliwy wymaga, aby równania kinematyczne zawierały konkretne parametry, od których będą zależne dalsze etapy. Dobór odpowiednich równań kinematycznych do testów nie był najprostszy, jak już wspomniano na początku rozdziału właśnie ze względu na specyfikację fizycznego robota. Dzięki zastosowaniu niezależnych napędów tylnej osi oraz sterowanego koła przedniego wybór konfiguracji modelu kinematycznego zależy od samego użytkownika, co daje dużo możliwości na etapie projektowania sterowania jak i samym sterowaniu. Omawiany robot jest układem trójkątowca, ze względu na dużą dowolność jaką oferuje model można rozpatrywać jego kinematykę na różne sposoby. Zakładając, że tylna osi napędowa posiada dwa niezależne napędy co zostało pokazane na rysunku 2.4, model kinematyczny można przedstawić wychodząc od następującego równania:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \frac{r}{2 \cdot l} \cdot \begin{bmatrix} l \cdot \sin \theta & l \cdot \sin \theta \\ l \cdot \cos \theta & l \cdot \cos \theta \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} \dot{\varphi}_1 \\ \dot{\varphi}_2 \end{bmatrix} \quad (2.1)$$



**Rysunek 2.4.** Dwukołowy robot mobilny z napędem różnicowym

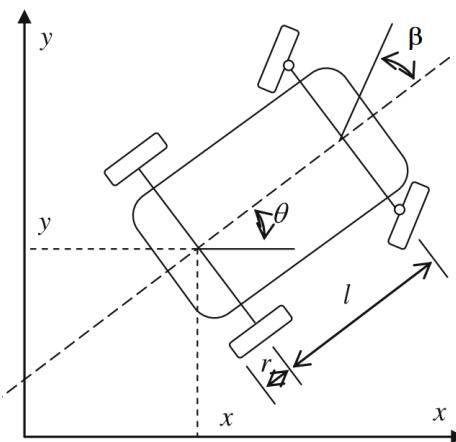
Źródło: Na podstawie [Gracia2002]

Powysze równanie przedstawia wcześniej wspomniany model kinematyczny robota dwukołowego z napędem różnicowym, praktyczne rozwiązania tego rodzaju posiadają dwa koła z niezależnymi napędami oraz punkt podparcia w postaci koła sferycznego lub koła kastora (samonastawne). Dodając do równia 2.1 pewne przekształcenie w postaci zależności kąta obrotu koła sterowego od prędkości kątowych podawanych na poszczególne koła napędowe (patrz 2.2) można uzyskać model kinematyczny trójkolowego robota (patrz rysunek 2.2). Równanie uwzględniające kąt obrotu przedniego koła sterowego wygląda następująco:

$$\beta = \arctan \left( \frac{\left( \frac{m}{l} \right) \cdot (\dot{\varphi}_1 - \dot{\varphi}_2)}{\dot{\varphi}_1 + \dot{\varphi}_2} \right) \quad (2.2)$$

Równanie na kąt obrotu koła sterowego  $\beta$  można traktować jako dodatkowy stan obiektu, na który mamy wpływ wyłącznie przez zmianę prędkości kątowych  $\dot{\varphi}_1$  i  $\dot{\varphi}_2$ .

Wcześniejsze równania kinematyki opisywały układ robota bazując na różnicowym modelu dwukołowym co jest jedną z możliwości, inną opcją jest wyjście od równań opisujących układ robota typu samochód ze sprzężonym napędem tylnej osi. Zazwyczaj w modelach kinematycznych robotów typu samochód dokonuje się pewnych uproszczeń układu skrętnego, w samochodach ma on za zadanie wychylać koła zgodnie z zasadą Ackermana [Ackerman2015], natomiast w modelach upraszcza się ją i koła skrętne wychylają się z tym samym kątem jak pokazano na rysunku 2.5. Takie rozwiązanie pozwala na opisanie trójkolowego robota za pomocą modelu czterokołowego, z punktu widzenia kinematyki ruch będzie się odbywał tak samo przy 2 kołach skrętnych jak i przy jednym. Model kinematyczny robota typu samochód przedstawia równanie 2.3.



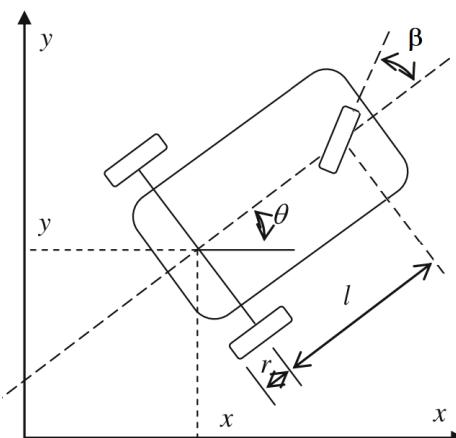
**Rysunek 2.5.** Robot mobilny typu samochód z sprężonym napędem tylnej osi  
Źródło: Na podstawie [Minh2014]

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\beta} \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \\ \frac{\tan \beta}{l} \\ 0 \end{bmatrix} \cdot r v_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \cdot v_2 \quad (2.3)$$

Przy czym  $v_1$  i  $v_2$  są odpowiednio prędkościami kątowymi robota oraz skrętu kół przednich, gdzie:

$$v_1 = \left( \frac{\sqrt{\dot{x}^2 + \dot{y}^2}}{r} \right) \text{ oraz } v_2 = \dot{\beta} \quad (2.4)$$

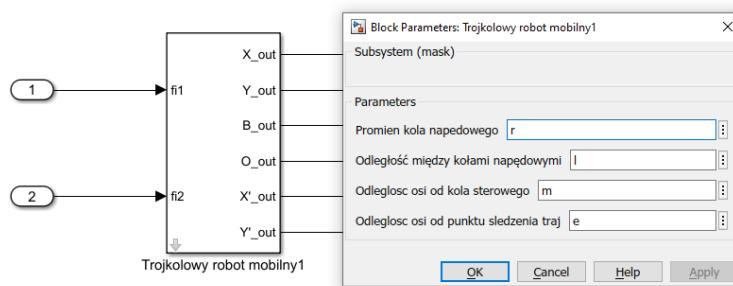
Jak można zauważyć na rysunku 2.5, w związku z tym, iż kąt obrotu kół jest taki sam dla obu stron można uprościć ten rysunek do postaci jaka została przedstawiona na rysunku 2.6, dzięki czemu możliwe jest wykorzystanie tego modelu do stworzenia modelu trójkolowca.



**Rysunek 2.6.** Trójkolowy robot mobilny z sprężonym napędem tylnej osi  
Źródło: Na podstawie [Minh2014]

Po przeprowadzeniu analiz został wybrany model opisany równaniami 2.1 oraz 2.2 ze względu na swoją uniwersalność, ponieważ podczas dalszych testów możliwe byłoby generowanie zarówno trajektorii bazującej o model samochodu z sprężonym napędem jak i model trójkółowca z napędem różnicowym. W przypadku wyboru modelu kinematycznego opisanego równaniami 2.3 mogłoby się okazać niemożliwe generowanie trajektorii bazującej na modelu trójkółowca z napędem różnicowym. Model z napędem różnicowym nadaje się do poruszania w ciasnych korytarzach, gdzie przestrzeń robocza jest znacznie ograniczona i manewrowanie robotem mobilnym z sprężonym napędem tylnym mogłoby okazać się znacznie bardziej skomplikowane.

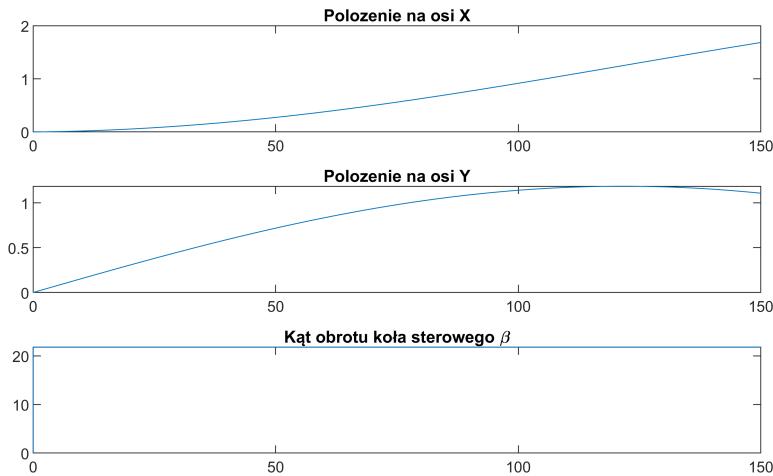
Model robota został stworzony w środowisku MATLAB 2020b, równania kinematyczne modelu zostały zaimplementowane oraz wyliczone. Po wyliczeniu przez program równań



**Rysunek 2.7.** Model robota w środowisku MATLAB Simulink

Źródło: Opracowanie własne

2.1 oraz 2.2 na poszczególne zmienne stanu zostały one zaimplementowane do MATLAB Simulink (Rysunek 2.7) w celu łatwiejszych testów symulacyjnych w dalszej części pracy. Możliwe było bezpośrednie zaimplementowanie równań do programu MATAB Simulink, lecz w obawie o możliwość popełnienia błędów podczas tworzenia modelu został wybrany inny sposób. W celu weryfikacji poprawności działania modelu zostały zadane przykładowe parametry  $\dot{\varphi}_1 = 1.5 \frac{m}{s}$  oraz  $\dot{\varphi}_2 = 1 \frac{m}{s}$ . Bazując na równaniu 2.2, kąt koła sterowego powinien wynieść około  $\beta \approx 22^\circ$  co potwierdza rysunek 2.8, na rysunku można zauważyć też, że podanie przybliżonych prędkości na koła skutkuje poruszaniem się robota po zakrzywionej trajektorii.

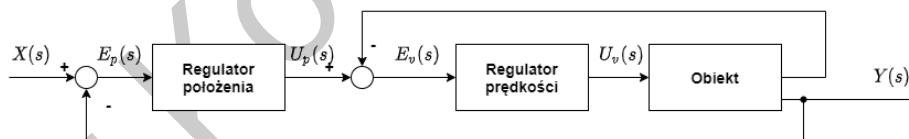


**Rysunek 2.8.** Odpowiedź robota na zadane przykładowe parametry

Źródło: Opracowanie własne

### 2.3. Regulator obiektu

Robot kołowy jako obiekt sterowania jest układem nieliniowym. Klasyczne układy regulacji typu PI, PD czy PID z ujemnym sprzężeniem zwrotnym niestety nie są odporne na wpływ tych nieliniowości. W takim przypadku stosuje się bardziej rozbudowane układy regulacji zawierające w swojej strukturze wcześniej wymienione regulatory. Przykładem takiego takiego rozwiązania może być układ regulacji kaskadowej (patrz rysunek 2.9), który posiada dwa regulatory, pierwszy steruje położeniem, drugi opowiada za prędkość. Zaletą tych

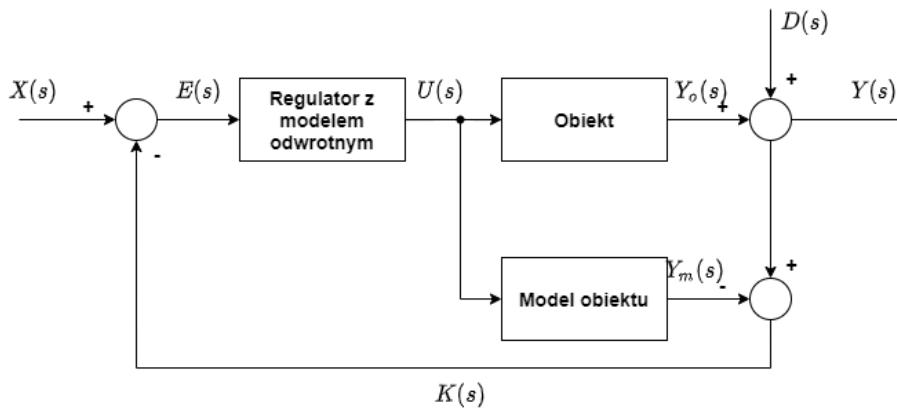


**Rysunek 2.9.** Przykład układu regulacji kaskadowej

Źródło: Opracowanie własne

rozwiązań jest to, że zakłócenia są niwelowane w pierwszym regulatorze i nie przechodzą do dalszej części układu regulacji co skutkuje lepszym sterowaniem, wymaga jednak dobrania nastaw dwóch regulatorów oraz konieczność dostępu do zmiennych obiektu potrzebnych w układzie regulacji.

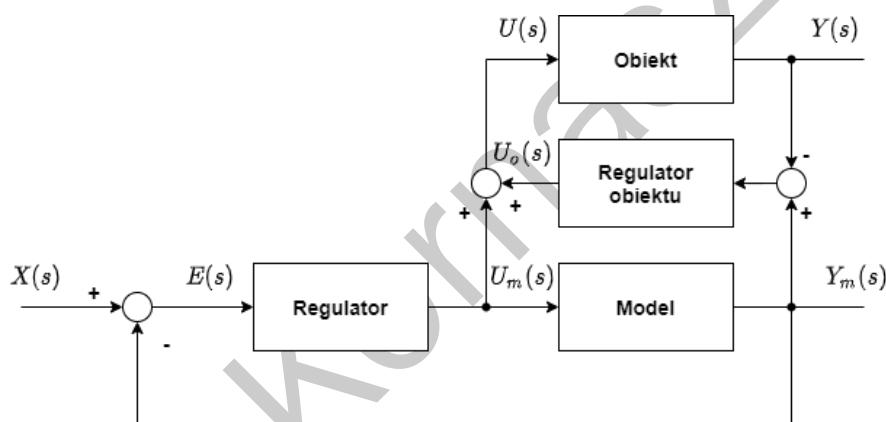
Popularnym rozwiązaniem w robotyce jest stosowanie układów regulacji bazujących na modelu obiektu tzw. MRCS (model reference control system), przykładem regulacji obmazującej na modelu obiektu może być sterowanie z wewnętrznym modelem. W idealnym przypadku układ z regulatorem bazującym na odwróconym modelu obiektu na wyjściu będzie zawsze równy wartości zadanej [**Inverse**]. W praktyce stosuje się układ z ujemnym sprzężeniem zwrotnym (patrz rysunek 2.10), dzięki czemu możliwe jest zniwelowanie wpływu



**Rysunek 2.10.** Sterowanie z odwrotnym modelem obiektu

Źródło: Opracowanie własne

niedokładności modelowania oraz wpływu zakłóceń na sterowanie. Aby możliwe było zastosowanie tego rozwiązania konieczne jest, aby model obiektu był odwracalny. Innym przykładem układu MRCS jest regulacja nadążająca za modelem (rysunek 2.11), stosuje się je tam, gdzie proces nie jest do końca znany lub model obiektu jest niedokładny. Sterowanie



**Rysunek 2.11.** Regulacja nadążająca za modelem

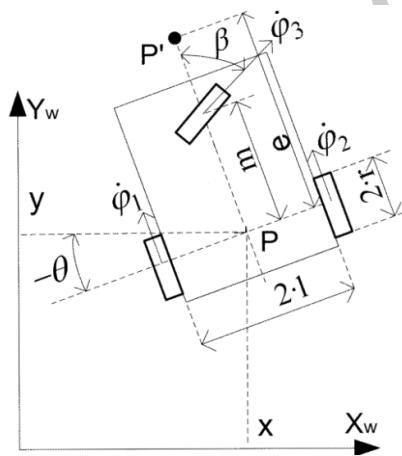
Źródło: Opracowanie własne

odbywa się po przez zadanie wartości w pętli sterowania modelem obiektu, po wygenerowaniu odpowiedzi przez model trafia ona na wejście regulatora obiektu, który ma za zadanie skorygowanie sygnału sterującego obiektem. Regulator modelu sprawia, iż model nadąży za wartością zadaną, a regulator obiektu sprawia, że obiekt nadąża za wyjściem modelu. Zaletą tego rozwiązania jest to, że wszelkie zakłócenia i niedokładności są eliminowane przez regulator obiektu co daje układ o charakterze odpornym, wadą rozwiązania jest konieczność nastrojenia dwóch regulatorów.

W omawianej pracy został wykorzystany układ regulacji z modelem odwrotnym obiektu i sterowaniem typu „point-to-point”, bazując na [Andrea1995] oraz [Gracia2002] został zaadaptowany układ regulacji bazujący na linearizacji równań stanów obiektu. W wyniku linearizacji został utracony dostęp do kąta obrotu robota  $\theta$ , jednak jest on powiązany ze stanami  $\dot{x}$  i  $\dot{y}$  w związku z czym można go uzyskać przez odpowiednią kombinację trajektorii robota. Równanie na regulator układowego przyjmuje następującą postać:

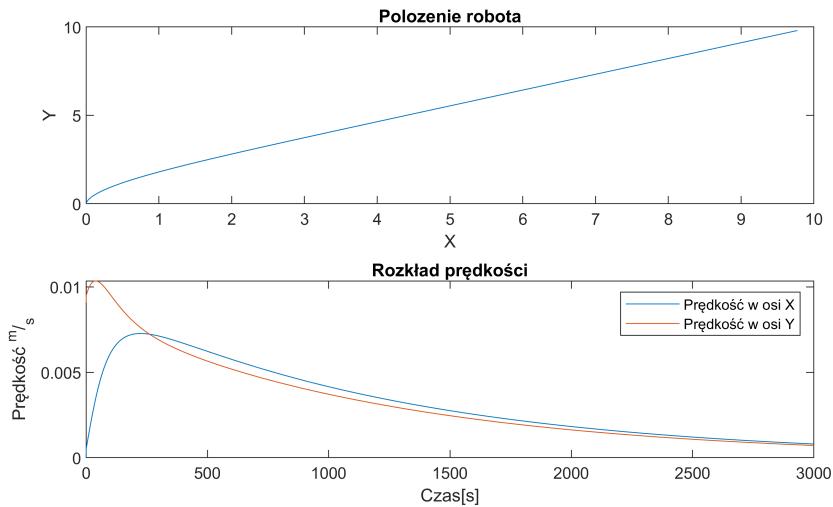
$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \end{bmatrix} = \frac{1}{r \cdot e} \cdot \begin{bmatrix} l \cdot \cos \theta & e \cdot \cos \theta \\ +e \cdot \sin \theta & -l \cdot \sin \theta \\ -l \cdot \cos \theta & e \cdot \cos \theta \\ +e \cdot \sin \theta & +l \cdot \sin \theta \end{bmatrix} \cdot \begin{bmatrix} \dot{P}'_x \\ \dot{P}'_y \end{bmatrix} - \begin{bmatrix} a_x & 0 \\ 0 & a_y \end{bmatrix} \cdot \begin{bmatrix} (x + e \cdot \sin \theta) \\ -(y + e \cdot \cos \theta) \\ (y + e \cdot \cos \theta) \\ -P'_y \end{bmatrix} \quad (2.5)$$

Regulator bazujący na równaniu 2.5 ma za zadanie wyznaczenie sterowania, które będzie śledziło punkt  $P'$  (rysunek 2.12), ograniczeniem tej metody jest to, że punkt  $P'$  nie może znajdować się w punkcie  $P$  i jeśli ten warunek jest spełniony to istnieje sterowanie zdolne osiągnąć wartość zadaną. Dzięki znajomości modelu odwrotnego oraz działaniu



**Rysunek 2.12.** Układ trójkątnego robota mobilnego z punktem śledzenia trajektorii  
 Źródło: Na podstawie [Gracia2002]

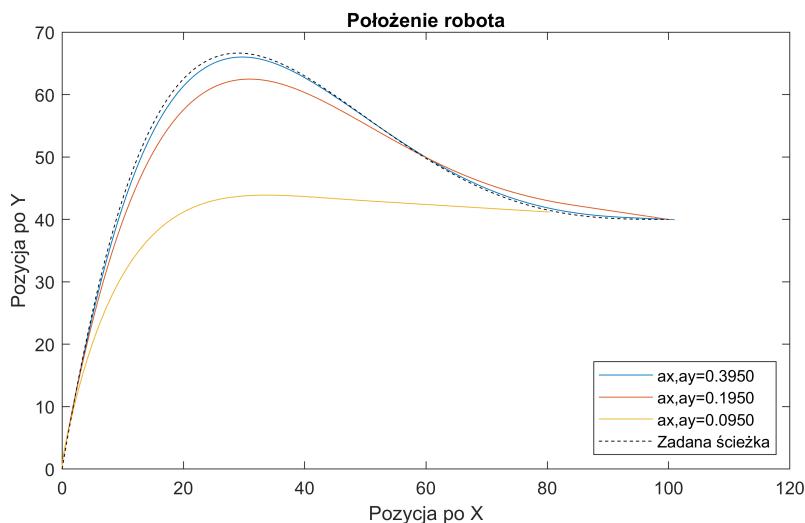
na błędzie położenia zadanego oraz położenia aktualnego robota układ jest w stanie osiągnąć zadane wartości nawet przy nieliniowościach modelu. Rysunek 2.13 przedstawia odpowiedź robota na zadane położenie końcowe  $X = 10$  i  $Y = 10$ , można zauważyć, iż robot osiąga zadane położenie a prędkości kół zbiegają do zera. Takie zachowanie jest pożąданie ponieważ robot powinien zatrzymać się po dotarciu w wyznaczone miejsce. Przez zawarcie w strukturze regulatora ograniczeń przyspieszenia za pomocą współczynników  $a_x$  i  $a_y$  możliwe jest dostosowanie prędkości dla układu w każdej osi. Takie rozwiązanie pozwala na wierniejsze odwzorowanie fizycznego obiektu bez konieczności modelowania dynamiki układu na etapie wyprowadzania równań kinematyki. Na pewno jest to niebywały atutem tego rozwiązania, w razie późniejszej implementacji do fizycznego robota możliwe będzie dostosowanie dynamiki właśnie dzięki tym parametrom i w związku z tym nie będzie konieczne przerabianie samego układu regulacji.



**Rysunek 2.13.** Odpowiedź układu regulacji na zadanie położenia końcowego

Źródło: Opracowanie własne

Dobór odpowiednich wartości współczynników  $a_x$  i  $a_y$  ma ogromny wpływ na odpowiedź robota na zadaną trajektorię. Żeby lepiej zobrazować wagę odpowiedniego doboru ograniczeń została zadana trajektoria o położeniu końcowym równym  $X = 100$  i  $Y = 40$  co obrazuje rysunek 2.14. Jak widać na załączonym rysunku przy zbyt dużym ograniczeniu prędkości robot nie dociera do zadanego położenia, jedynie przy wartościach powyżej  $a_x, a_y \approx 0.19$  robot jest w stanie dotrzeć do zadanego położenia.

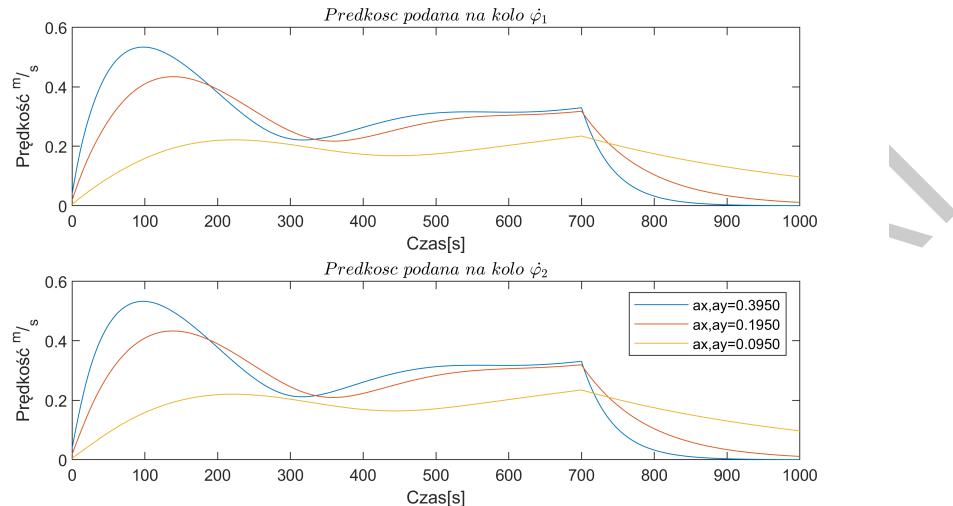


**Rysunek 2.14.** Porównanie trajektorii robota dla różnych  $a_x$  i  $a_y$

Źródło: Opracowanie własne

Oczywistym wyborem jest dobranie jak najmniejszych wartości ograniczeń, tak aby robot odwzorowywał jak najwierniej zadaną trajektorię. Jednak prędkości kół robota przy słabych ograniczeniach znaczco rosną co można zaobserwować na rysunku 2.15, widać

również, że przy zbyt dużym ograniczeniach robot nie jest w stanie ukończyć zadanej trasy w wyznaczonym czasie. W przypadku symulacji wartości prędkości nie ma większego znaczenia ale w fizycznym obiekcie może być niebezpieczne, z tego względu konieczne jest znalezienie złotego środka pomiędzy dokładnością odwzorowania trajektorii a prędkościami robota.



**Rysunek 2.15.** Porównanie sygnałów sterujących robota dla różnych  $a_x$  i  $a_y$

Źródło: Opracowanie własne

Mając na uwadze konieczność implementacji stworzonych rozwiązań wartości ograniczeń zostały dobrane, tak aby robot był w stanie osiągnąć zadane położenie przy jak najmniejszej możliwej prędkości i wynoszą one  $a_x, a_y = 0.1950$ . Prawdopodobnie istnieje możliwość lepszego doboru ograniczeń na przykład przez zastosowanie optymalizacji lecz na potrzeby symulacji takie rozwiązanie nie było konieczne.

Piotr Kornaszewski

## **ROZDZIAŁ 3**

# **Wybrane metody generowania trajektorii robotów mobilnych**

W rozdziale tym zawarty został główny cel pracy, czyli analiza wybranych metod generowania trajektorii robotów mobilnych. Zostały w nim zaprezentowane metody płaskie, wielomianowe oraz symetryczne wielomianowe. Przedstawiono je od strony teoretycznej jak i od strony praktycznej wraz z testami symulacyjnymi.

## **3.1. Teoretyczne zagadnienie generowania trajektorii robota mobilnego**

Rozdział pierwszy przedstawał w ogólny sposób różne metody planowania trajektorii robotów mobilnych począwszy od metod lokalnych, globalnych oraz metod mieszanych planowania ścieżki. W tym rozdziale zostały przedstawione metody lokalne, zastosowanie ich pozwoli na implementację gotowego rozwiązania do fizycznego robota. Zastosowanie metod globalnych w tym przypadku jest utrudnione, ponieważ robot docelowo ma poruszać się po korytarzu uczelnianym co wiąże się z możliwością dynamicznego występowania przeszkód. Z tego właśnie powodu zostały wybrane metody lokalne generowania trajektorii opisywane w rozdziale pierwszym, lecz w bardziej szczegółowy sposób skupiając się na położeniu robota, położeniu kątowym oraz kącie obrotu koła sterowego.

### **3.1.1. Metoda generowania trajektorii płaskiej**

Mówiąc o trajektorii płaskiej należy najpierw wspomnieć czym jest obiekt płaski lub płasko-różnicowy. Systemem czy też obiektem płaskim nazywamy własność obiektu w której, za pomocą zmiennych wyjściowych (pochodne skończonego rzędu) w liczbie równej ilości wejść obiektu, możliwe jest uzyskanie dowolnego stanu w obiekcie przez stworzenie funkcji różnicycznych tych zmiennych [Levine2009]. Równania powstałe dzięki takiemu zapisowi obiektu pozwalają znacznie uprościć sterowanie układami nieliniowymi jak i liniowymi a w dodatku

nie wymagają dużej mocy obliczeniowej i wyspecjalizowanych sensorów. Za pomocą tej metody można uzyskać płynne przyspieszanie robotem mobilnym bez nagłych szarpnięć czy też gwałtownych hamowań co jest groźne dla mechanizmów napędowych. Metoda ta została właśnie wybrana ze względu na możliwość dość prostej i taniej implementacji w fizycznym obiekcie.

Bazując na modelu kinematycznym 2.3 zostały wyprowadzone równania generatora trajektorii typu „off-line” [Levine2009]. Równania te pozwalają na wygenerowanie zbioru punktów w przestrzeni kartezjańskiej od położenia początkowego  $(x_0, y_0, \theta_0, \beta_0)$  robota aż do punktu końcowego  $(x_T, y_T, \theta_T, \beta_T)$  w określonym czasie  $T$ . Równania te prezentują się następująco:

dla położenia w osi  $X$  :

$$x(t) = \left( \frac{T-t}{T} \right) x_0 + \frac{t}{T} x_T + |x_T - x_0| \frac{t(t-T)}{2T^2} \quad (3.1)$$

oraz dla położenia w osi  $Y$  :

$$y(t) = y_0 + t\alpha_1 \tan \theta_0 + t^2 \frac{\alpha_2 \tan \beta_0}{2l \cos^3 \theta_0} + t^3 b_1 + t^4 b_2 + t^5 b_3 \quad (3.2)$$

gdzie  $\alpha_1 = \frac{2(x_t - x_0) - |x_t - x_0|}{2T}$ ,  $\alpha_2 = \frac{|x_t - x_0|}{T^2}$ ,  $\alpha_3 = \frac{2(x_t - x_0) - |x_t - x_0|}{2T}$  oraz dla kolejnych współczynników  $[b_1 \ b_2 \ b_3]' = A^{-1}c$  przy czym:

$$A = \begin{bmatrix} T^3 & T^4 & T^5 \\ 3T^2 & 4T^3 & 5T^4 \\ 6T & 12T^2 & 20T^3 \end{bmatrix} \quad \text{oraz} \quad c = \begin{bmatrix} y_T - y_0 - T\alpha_1 \tan \theta_0 - T^2 \frac{\alpha_2 \tan \beta_0}{2l \cos^3 \theta_0} \\ \alpha_3 \tan \theta_T \alpha_1 \tan \theta_0 - T^2 \frac{\alpha_2 \tan \beta_0}{l \cos^3 \theta_0} \\ \frac{\alpha_2 \tan \beta_T}{l \cos^3 \theta_T} - \frac{\alpha_2 \tan \beta_0}{l \cos^3 \theta_0} \end{bmatrix} \quad (3.3)$$

Powyższe równania w pełni wystarczają do wygenerowania ścieżki dla regulatora zaimplementowanego w robocie mobilnym. Ponieważ regulator steruje tylko położeniem robota w osi X i Y generowanie równań na położenie kątowe robota oraz jego koła sterowego nie jest konieczne. Jednak jeśli zostaną wygenerowane odpowie trajektorie to istnieje możliwość wpływania w sposób pośredni na pozostałe stany obiektu. W związku z tym dodane zostały równania na położenie kątowe  $\theta$  robota oraz na położenie kątowe koła sterowego  $\beta$ :

$$\theta(t) = \arctan \left( \frac{2T^2 \left( \alpha_1 \tan \theta_0 + t^2 \frac{\alpha_2 \tan \beta_0}{l \cos^3 \theta_0} + 3b_1 t^2 + 4b_2 t^3 + 5b_3 t^4 \right)}{2T(x_t - x_0) - T|x_T - x_0| + 2|x_T - x_0|} \right) \quad (3.4)$$

jak również:

$$\beta(t) = \arctan \left( \frac{\left( \frac{\alpha_2 \tan \beta_0}{l \cos^3 \theta_0} + 6b_1 t + 12b_2 t^2 + 20b_3 t^3 \right) (2t^2)^2 l \cos^3 \theta_0}{2T(x_t - x_0) - T|x_T - x_0| + 2|x_T - x_0|} \right) \quad (3.5)$$

Wyniki praktyczne powyższych równań zostaną przedstawione w sekcji 3.2 wraz z realizacją praktyczną za pośrednictwem środowiska MATLAB 2020b.

### 3.1.2. Metoda generowania trajektorii wielomianowej

Metoda generowania trajektorii wielomianowej ze sposobu wyprowadzania równań podobnie jak w poprzedniej metodzie 3.1.1, polega na stworzeniu wielomianów wyższego rzędu. Równania te bazują w tym przypadku na wejściach robota a nie jak w przypadku trajektorii płaskiej na wyjściach [Wenjie2005]. Zastosowanie tej metody pozwala na uzyskanie precyzyjnej ścieżki robota i wpływu na poszczególne stany. Metody generowania trajektorii były znane już w latach 80. jednak ze względu na dostępną technologię możliwości generowania trajektorii za pomocą wielomianów wyższego rzędu były znacznie ograniczone. Dzisiejsze technologie znaczco ułatwiają sprawę i możliwe jest wierniejsze odwzorowanie stawianych wymagań przez obliczanie znacznie bardziej skomplikowanych równań.

Podobnie jak w poprzednim przypadku tutaj również został wykorzystany ten sam model kinematyczny 2.3 na podstawie którego zostały wyznaczone równania generatora trajektorii wielomianowej [Minh2014]. Ścieżka robota zostaje wygenerowana zaczynając z położenia początkowego  $(x_0, y_0, \theta_0, \beta_0)$  robota aż do punktu końcowego  $(x_T, y_T, \theta_T, \beta_T)$  w czasie  $T$ . Równania generatora pozwalają na wygenerowanie ciągłej trajektorii i prezentują się następująco:

dla położenia w osi  $X$  :

$$x(t) = x_0 + gt \quad (3.6)$$

dla położenia w osi  $Y$  :

$$y(t) = y_0 + g\theta_0 t + \frac{1}{2}g^2\beta_0 t^2 + \frac{1}{6}g^2h_1 t^3 + \frac{1}{24}g^2h_2 t^4 + \frac{1}{60}g^2h_3 t^5 \quad (3.7)$$

dla położenia kątowego  $\theta$ :

$$\theta(t) = \arctan \left( \theta_0 + g\beta_0 t + \frac{1}{2}gh_1 t^2 + \frac{1}{6}gh_2 t^3 + \frac{1}{12}gh_3 t^4 \right) \quad (3.8)$$

jak również dla położenia koła sterowego  $\beta$ :

$$\beta(t) = \arctan \left( \beta_0 + h_1 t + \frac{1}{2}h_2 t^2 + \frac{1}{3}h_3 t^3 \right) \quad (3.9)$$

gdzie  $g = \frac{x_T - x_0}{T}$  oraz dla kolejnych współczynników  $[h_1 \ h_2 \ h_3]' = D^{-1}e$  przy czym:

$$D = \begin{bmatrix} T & \frac{1}{2}T^2 & \frac{1}{3}T^3 \\ \frac{1}{2}gT^2 & \frac{1}{6}gT^3 & \frac{1}{12}gT^4 \\ \frac{1}{6}g^2T^3 & \frac{1}{24}g^2T^4 & \frac{1}{60}g^2T^5 \end{bmatrix} \quad \text{oraz} \quad e = \begin{bmatrix} \beta_T - \beta_0 \\ \theta_T - \theta_0 - g\beta_T T \\ y_T - y_0 - g\theta_T T - \frac{1}{2}g^2\beta_T T^2 \end{bmatrix} \quad (3.10)$$

### 3.1.3. Metoda generowania trajektorii symetrycznej-wielomianowej

Bazując na założeniu, iż model robota jest płaski i możliwe jest opisanie zmiennych stanu za pomocą wyjść robota możliwe jest zastosowanie wielomianów trzeciego rzędu (najniższy rząd pozwalający na wygenerowanie optymalnej trajektorii) do wygenerowania trajektorii

**[Minh2014].** W omawianym przypadku zaletą wygenerowanej ścieżki jest zerowa prędkość w punkcie docelowym co z kolei jest pożądanym zachowaniem podczas wykonywania trajektorii przez robota mobilnego. Z racji zastosowania wielomianów niższego rzędu sama trajektoria nie będzie posiadała zakrzywień co z kolei zostanie szczegółowo pokazane w dalszej części praktycznej.

Równania generatora symetryczno-wielomianowego można zapisać w następujący sposób:

dla położenia w osi  $X$  :

$$x(t) = -\left(\frac{t}{T} - 1\right)^3 x_0 + \left(\frac{t}{T}\right)^3 x_T + a_x \left(\frac{t}{T}\right)^2 \left(\frac{t}{T} - 1\right) + b_x \left(\frac{t}{T}\right) \left(\frac{t}{T} - 1\right)^2 \quad (3.11)$$

dla położenia w osi  $Y$  :

$$y(t) = -\left(\frac{t}{T} - 1\right)^3 y_0 + \left(\frac{t}{T}\right)^3 y_T + a_y \left(\frac{t}{T}\right)^2 \left(\frac{t}{T} - 1\right) + b_y \left(\frac{t}{T}\right) \left(\frac{t}{T} - 1\right)^2 \quad (3.12)$$

dla położenia kątowego  $\theta$ :

$$\theta(t) = \arctan\left(\frac{\dot{y}}{\dot{x}}\right) \quad (3.13)$$

jak również dla położenia koła sterowego  $\beta$ :

$$\beta(t) = \arctan\left(l \frac{\cos^3 \theta \ddot{y}}{\dot{x}^2}\right) \quad (3.14)$$

gdzie współczynniki prędkości  $ax, bx$  wynoszą:

$$a_x = k \cos \theta_T - 3x_T, \quad \text{oraz} \quad b_x = k \cos \theta_0 - 3x_0 \quad (3.15)$$

analogicznie dla osi  $Y$  współczynniki wyglądają podobnie:

$$a_y = k \cos \theta_T - 3y_T, \quad \text{oraz} \quad b_y = k \cos \theta_0 - 3y_0 \quad (3.16)$$

## 3.2. Praktyczna realizacja generowania trajektorii robota mobilnego

Wcześniej podrozdział skupiał się na teoretycznym podejściu do metod generowania trajektorii robota mobilnego, w tym podrozdziale zostaną przedstawione ich praktyczne realizacje wraz z wykresami ilustrującymi ich działanie.

### 3.2.1. Trajektoria płaska

Bazując na równaniach z podrozdziału 3.1.1 zostały napisany kod generatora trajektorii płaskiej w języku programowania MATLAB 3.1.

### Kod źródłowy 3.1. Generator trajektorii płaskiej w programie MATLAB

Źródło: Opracowanie własne

```
1 %% Trajektoria płaska
2
3 clear all
4
5 syms T t beta_0 beta_T Theta_0 Theta_T x_0 y_0 x_T y_T l ;
6
7 a1 = ((2*(x_T - x_0)) - abs(x_T - x_0))/(2*T);
8 a2 = (abs(x_T - x_0))/(T^2);
9 a3 = (2*(x_T - x_0) + abs(x_T - x_0))/(2*T);
10
11 A = [T^3 T^4 T^5; 3*T^2 4*T^3 5*T^4; 6*T 12*T^2 20*T^3];
12
13 c = [(y_T - y_0 - T*a1*tan(Theta_0)-T^2 *
((a2*tan(beta_0))/(2*l*cos(Theta_0).^3)));...
14 (a3*tan(Theta_T)- a1*tan(Theta_0)-T *
((a2*tan(beta_0))/(l*cos(Theta_0).^3)));...
15 ((a2*tan(beta_T))/(l*cos(Theta_T).^3)) -
((a2*tan(beta_0))/(l*cos(Theta_0).^3))];
16
17 b = A^(-1)*c;
18
19 b1 = b(1,1);
20 b2 = b(2,1);
21 b3 = b(3,1);
22
23 xt = ((T-t)/T)*x_0 + (t/T)*x_T + abs(x_T - x_0)*(t*(t-T)/(2*T^2));
24 yt = y_0 + t*a1*tan(Theta_0) + t^2*((a2*tan(beta_0))/(2*l*cos(Theta_0).^3)) + ...
25 t^3*b1 + t^4*b2 + t^5*b3;
26
27 Theta = atan(2*T^2*(a1*tan(Theta_0) +
((a2*tan(beta_0))/(l*cos(Theta_0).^3))*t^2+3*b1*t^2 + 4*b2*t^3 + 5*b3*t^4 )...
28 /(2*T*(x_T-x_0) - T*abs(x_T - x_0) + 2*abs(x_T - x_0)*t));
29
30 beta = atan(((a2*tan(beta_0))/(l*cos(Theta_0).^3)) + 6*b1*t + 12*b2*t^2 +
20*b3*t^3)*(2*T^2)^2*cos(Theta_0).^3 )...
31 /((2*T*(x_T-x_0) - T*abs(x_T - x_0) + 2*abs(x_T - x_0)*t))^2) ;
32
33 f = matlabFunction(xt,yt,Theta,beta,'File','myfile',...
34 'Outputs',{'X','Y','O','B'});
```

Powyższy kod pozwala uzyskać równania a następnie stworzyć funkcję służącą do wygenerowania trajektorii robota. Zamiana równań z postaci symbolicznej na funkcję programu MATLAB pozwala uzyskać gotową trajektorię w mniej niż 0,02 sekundy, taki wynik jest jak najbardziej zgodny z ideą systemów czasu rzeczywistego, gdzie właśnie wykorzystuje się tego typu generatory. W celu sprawdzenia poprawności wyliczonych funkcji generowania trajektorii zostały zadane początkowe parametry położenia i orientacji robota ( $x_0, y_0, \theta_0, \beta_0$ ) jak i końcowe ( $x_T, y_T, \theta_T, \beta_T$ ) wraz z czasem symulacji  $T$ , które zostały

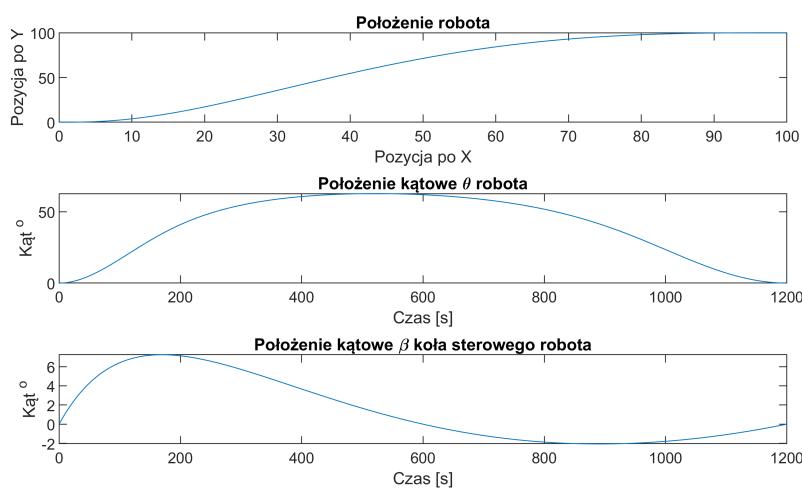
umieszczone w tabeli 3.1.

Na rysunku 3.1 można zaobserwować, iż generator wyznaczył krzywą prowadzącą z położenia początkowego aż do położenia końcowego zgodnie z zadanimi kryteriami zawartymi we wcześniej omawianej tabeli co wystarczy, aby uznać poprawność generatora na tym etapie przeprowadzonych symulacji.

**Tabela 3.1.** Tabela zbiorcza parametrów regulatora oraz generatora trajektorii

Źródło: Opracowanie własne

Nazwa	Parametr	Wartość
Początkowe położenie w osi X	$x_0$	0
Początkowe położenie w osi Y	$y_0$	0
Początkowe położenie kątowe robota	$\theta_0$	0
Początkowe położenie kątowe koła sterowego	$\beta_0$	0
Końcowe położenie w osi X	$x_T$	100
Końcowe położenie w osi Y	$y_T$	100
Końcowe położenie kątowe robota	$\theta_T$	0
Końcowe położenie kątowe koła sterowego	$\beta_T$	0
Czas potrzebny na realizację	$T$	1200
Ograniczenia prędkości w osiach X i Y	$a_x, a_y$	0.1950



**Rysunek 3.1.** Wygenerowana trajektoria płaska

Źródło: Opracowanie własne

Załączony wykres 3.1 pokazuje, że założone parametry symulacji zostały spełnione w zadany czasie, dalsze testy symulacyjne wykażą czy metoda będzie tak samo funkcjonalna podczas wykonywania przez układ regulacji robota.

### 3.2.2. Trajektoria wielomianowa

W przypadku trajektorii wielomianowej generator bazuje na równaniach z sekcji 3.1.2, jak i również tak jak poprzedni generator realizację praktyczną wykonano w programie MATLAB.

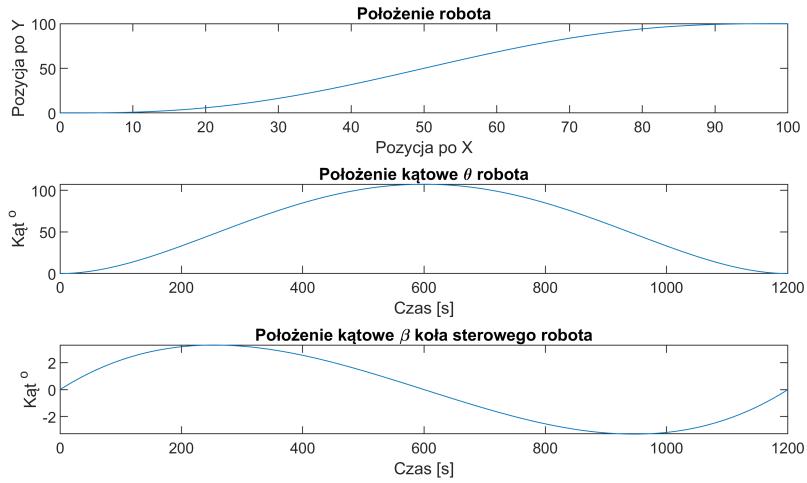
**Kod źródłowy 3.2.** Generator trajektorii wielomianowej w programie MATLAB

Źródło: Opracowanie własne

```
1 %% Trajektoria wielomianowa
2
3 clear all
4
5 syms T t beta_0 beta_T Theta_0 Theta_T x_0 y_0 x_T y_T;
6
7 g =((x_T-x_0)/(T));
8 D = [T (1/2)*T^2 (1/3)*T^3; (1/2)*g*T^2 (1/6)*g*T^3 (1/12)*g*T^4;...
9 (1/6)*g^2*T^3 (1/24)*g^2*T^4 (1/60)*g^2*T^5];
10
11 e = [(beta_T - beta_0);(Theta_T - Theta_0 - (g*beta_0*T));...
12 (y_T - y_0 - g*Theta_0*T - (0.5*g^2*beta_0*T^2))];
13
14 h = D^(-1)*e;
15 h1 = h(1,1);
16 h2 = h(2,1);
17 h3 = h(3,1);
18
19 xt1 = x_0 + g*t;
20 yt1 = y_0 + g*Theta_0*t + (1/2)*g^2*beta_0*t^2 + (1/6)*g^2*h1*t^3 +...
21 (1/24)*g^2*h2*t^4 + (1/60)*g^2*h3*t^5;
22
23 Theta = Theta_0 + g*beta_0*t + (1/2)*g*h1*t^2 + (1/6)*g*h2*t^3 + (1/12)*g*h3*t^4;
24
25 Beta = beta_0 + h1*t + (1/2)*h2*t^2 + (1/3)*h3*t^3;
26
27 f1 = matlabFunction(xt1,yt1,Theta,Beta,'File','poly',...
28 'Outputs',{'X1','Y1','0','B'});
```

Kod 3.2 pozwala na stworzenie funkcji umożliwiającej generowanie trajektorii wielomianowej, tutaj również czas potrzebny na wykonanie obliczeń nie przekracza 0,02 sekundy co jest w pełni satysfakcyjnym wynikiem i może posłużyć jako generator trajektorii czasu rzeczywistego.

W celu weryfikacji zadano położenie początkowe oraz położenie końcowe (patrz tabela 3.1) i otrzymano rezultaty widoczne na rysunku 3.2, wyniki są zgodne z założeniami i pozwalają potwierdzić zgodność równań z realizacją praktyczną.



**Rysunek 3.2.** Wygenerowana trajektoria wielomianowa

Źródło: Opracowanie własne

Rysunek 3.1 pokazuje, że założone parametry symulacji zostały spełnione w zadanym czasie, dalsze testy symulacyjne wykażą czy metoda będzie tak samo funkcjonalna podczas wykonywania przez układ regulacji robota.

### 3.2.3. Trajektoria symetryczna-wielomianowa

Ostatnim z zrealizowanych generatorów w niniejszej pracy był generator trajektorii symetryczno-wielomianowej i analogicznie do wcześniejszych on również został zrealizowany bazując na części teoretycznej z sekcji 3.1.3, jak i również w języku programowania matlab.

**Kod źródłowy 3.3.** Generator trajektorii symetryczno-wielomianowej w programie MATLAB

Źródło: Opracowanie własne

```

1  %% Trajektoria symetryczno-wielomianowa
2
3  clear all
4
5
6  syms T t beta_0 beta_T Theta_0 Theta_T x_0 y_0 x_T y_T ks 1;
7
8  ax = ks*cos(Theta_T)-3*x_T;
9  bx = ks*cos(Theta_0)-3*x_0;
10
11 ay = ks*sin(Theta_T)-3*y_T;
12 by = ks*sin(Theta_0)-3*y_0;
13
14 xt = -((t/T)-1)^3*x_0 + (t/T)^3*x_T + ax*(t/T)^2*((t/T)-1)...
15      + bx*(t/T)*((t/T)-1)^2;
16
17 yt = -((t/T)-1)^3*y_0 + (t/T)^3*y_T + ay*(t/T)^2*((t/T)-1)...

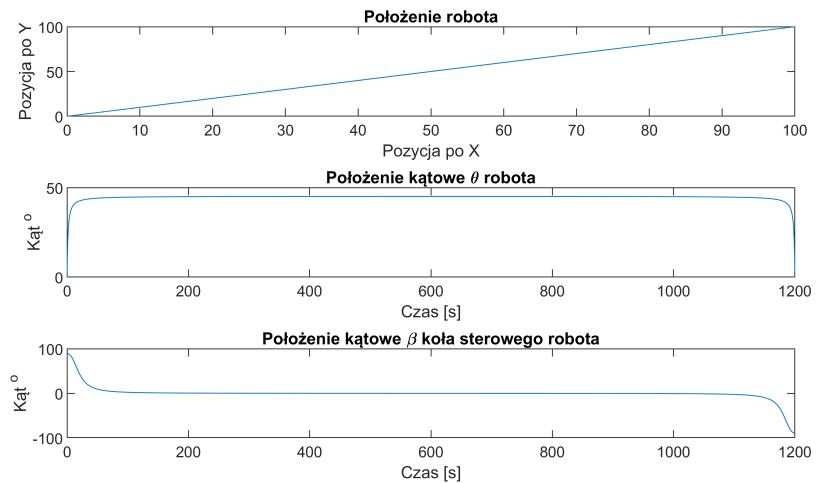
```

```

18      + by*(t/T)*((t/T)-1)^2;
19
20
21      xt_dot = -3*((t/T)-1)^2*x_0 + 3*(t/T)^2*x_T + ax*2*(t/T)*((t/T)-1)+ ax*(t/T)^2 ...
22          + bx*((t/T)-1)^2 + bx*2*(t/T)*((t/T)-1);
23
24      yt_dot = -3*((t/T)-1)^2*y_0 + 3*(t/T)^2*y_T + ay*2*(t/T)*((t/T)-1)+ ay*(t/T)^2 ...
25          + by*((t/T)-1)^2 + by*2*(t/T)*((t/T)-1);
26
27      xt_dot2 = -6*((t/T)-1)*x_0 + 6*(t/T)*x_T + ax*2*(2*(t/T)-1)+ ax*2*(t/T) ...
28          + bx*2*((t/T)-1) + bx*2*(2*(t/T)-1);
29
30      yt_dot2 = -6*((t/T)-1)*y_0 + 6*(t/T)*y_T + ay*2*(2*(t/T)-1)+ ay*2*(t/T) ...
31          + by*2*((t/T)-1) + by*2*(2*(t/T)-1);
32
33 Theta = atan(yt_dot/xt_dot);
34 Beta = atan(l*((cos(Theta).^3*yt_dot2)/(xt_dot)^2));
35 f = matlabFunction(xt,yt,Theta,Beta,'File','spoly',...
    'Outputs',{'X','Y','0','B'});

```

Kod 3.3 daje możliwość na stworzenie funkcji umożliwiającej wygenerowanie trajektorii symetryczno-wielomianowej, z racji, iż równania nie są skomplikowane czas i moc potrzebny na ich obliczenie jest mniejszy niż w poprzednich przypadkach. W celu weryfikacji zadano położenie początkowe oraz położenie końcowe (patrz tabela 3.1) i otrzymano rezultaty widoczne na rysunku 3.2, jak można zauważyć metoda ta otrzymuje inne położenia koła w skrajnych punktach trajektorii. Wyniki są częściowo zgodne z zadanymi położeniami robota z wyjątkiem położenia koła sterowego  $\beta$ .



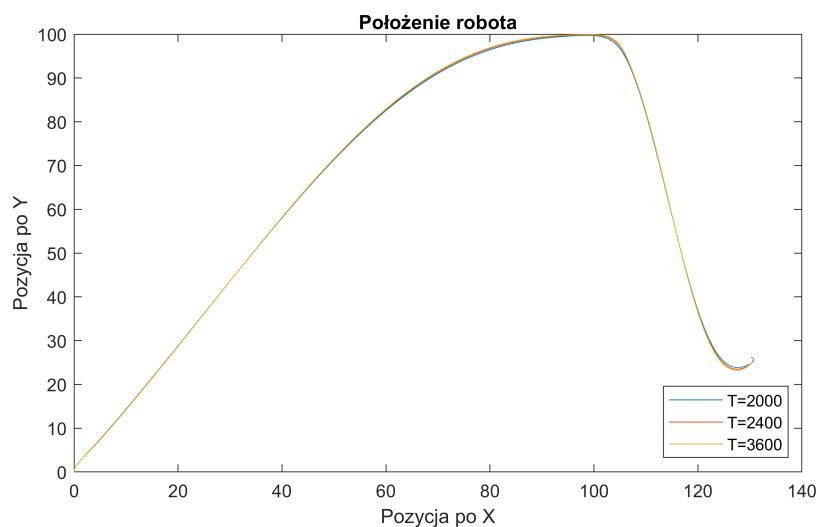
**Rysunek 3.3.** Wygenerowana trajektoria symetryczno-wielomianowa  
 Źródło: Opracowanie własne

### 3.3. Testy symulacyjne

Wcześniejsze podrozdziały skupiały się na teoretycznym jak i praktycznym generowaniu trajektorii do robota mobilnego. Na załączonych wykresach zostały wykreślone przykładowe trajektorie do każdego omawianego przykładu. Obecny podrozdział skupił się na testach generowanych trajektorii wykorzystując obiekt w postaci modelu kinematycznego robota (patrz 2.2) oraz układu regulacji bazującego na modelu odwrotnym (2.5). Wyniki symulacji były oceniane pod kątem osiągnięcia zadanego położenia, położenia kątowego robota, położenia koła sterowego oraz prędkości liniowej na wyjściu robota.

Bazując na przygotowanym modelu kinematycznym robota mobilnego oraz jego układzie sterowania zostały przeprowadzone symulacje mające na celu wskazać najlepszy algorytm do zaadaptowania w prawdziwym robocie mobilnym. Do celów symulacji zostały pominięte tarcia, masa robota czy też wpływ niedokładności modelowania obiektu. Kryterium oceny jakości wygenerowanej trajektorii opierało się na jak najmniejszym promieniu skrętu koła sterowego oraz możliwie gładkiej krzywej wygenerowanej trajektorii przy jednoczesnym osiągnięciu zadanego położenia. Ważnym czynnikiem podczas generowania trajektorii robota okazała się długość trasy, niektóre generatory wykazały nie pożądane właściwości. Wyniki symulacji oraz pewne niekorzystne właściwości zostaną pokazane w dalszej części.

Zaczynając testy symulacyjne należało przyjąć pewne wspólne parametry pozwalające na analizę otrzymanych wyników. W rozdziale drugim przy omawianiu regulatora 2.3 zostały przyjęte parametry ograniczeń prędkości  $a_x, a_y$ . Kolejnym niezbędnym parametrem podczas generowania trajektorii jest czas  $T$  w jakim ma zostać zrealizowana trajektoria. W zależności od czasu trwania trajektorii przyjmie ona różne wartości parametrów  $(x(t), y(t), \theta(t), \beta(t))$  co ma wpływ na prędkości kół robota  $\dot{\phi}_1(t)$  i  $\dot{\phi}_2(t)$ . Aby dobrać odpowiedni czas trwania symulacji przyjęte zostały parametry opisane w tabeli 3.2, na ich podstawie zostały przeprowadzone poszukiwania optymalnej wartości parametru  $T$ .



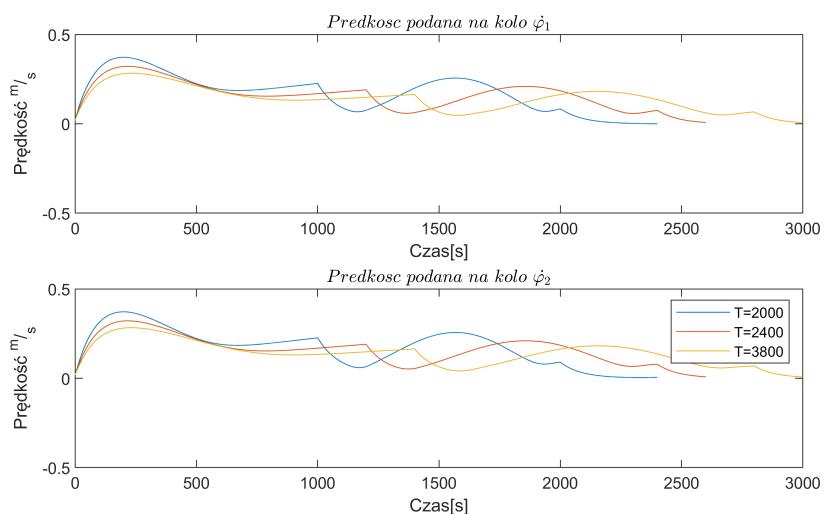
**Rysunek 3.4.** Porównanie trajektorii robota przy różnym czasie  $T$

Źródło: Opracowanie własne

**Tabela 3.2.** Tabela zbiorcza parametrów regulatora oraz generatora trajektorii - dobór parametru czasu generowania trajektorii  $T$

Źródło: Opracowanie własne

Nazwa	Parametr	Wartość
Początkowe położenie w osi X	$x_0$	0
Początkowe położenie w osi Y	$y_0$	0
Początkowe położenie kątowe robota	$\theta_0$	0
Początkowe położenie kątowe koła sterowego	$\beta_0$	0
Pośrednie położenie w osi X	$x_{T0.5}$	105
Pośrednie położenie w osi Y	$y_{T0.5}$	105
Pośrednie położenie kątowe robota	$\theta_{T0.5}$	0
Pośrednie położenie kątowe koła sterowego	$\beta_{T0.5}$	0
Końcowe położenie w osi X	$x_T$	130
Końcowe położenie w osi Y	$y_T$	25
Końcowe położenie kątowe robota	$\theta_T$	0
Końcowe położenie kątowe koła sterowego	$\beta_T$	0
Ograniczenia prędkości w osiach X i Y	$a_x, a_y$	0.1950



**Rysunek 3.5.** Porównanie prędkości podawanych na koła przy różnym czasie  $T$

Źródło: Opracowanie własne

Analizując rysunek 3.4 przedstawiający tą samą trajektorię dla różnych wartości czasu trwania trajektorii można stwierdzić, że zmiana tego parametru ma znikomy wpływ na położenie robota i ciężko wskazać, który czas jest najlepszy. Jednakże rysunek 3.5 przedstawia sygnały sterujące wygenerowane przez regulator, można zauważać znaczącą różnicę w czasie trwania przejazdu jak i samej wartości prędkości podawanych na koła robota. Bazując na otrzymanych wynikach optymalnym wyborem między czasem potrzebnym na przejechanie przez robota wygenerowanej trajektorii a prędkością robota będzie wartość współczynnika  $T = 2400$ . Dobrana wartość współczynnika  $T$  jest wystarczająca na potrzeby symulacji, możliwe, że zastosowanie bardziej wyszukanych metod pozwoliłoby lepiej dobrać jego wartość. W dalszych etapach zostaną przedstawione wyniki badań w odniesieniu do konkretnych przypadków.

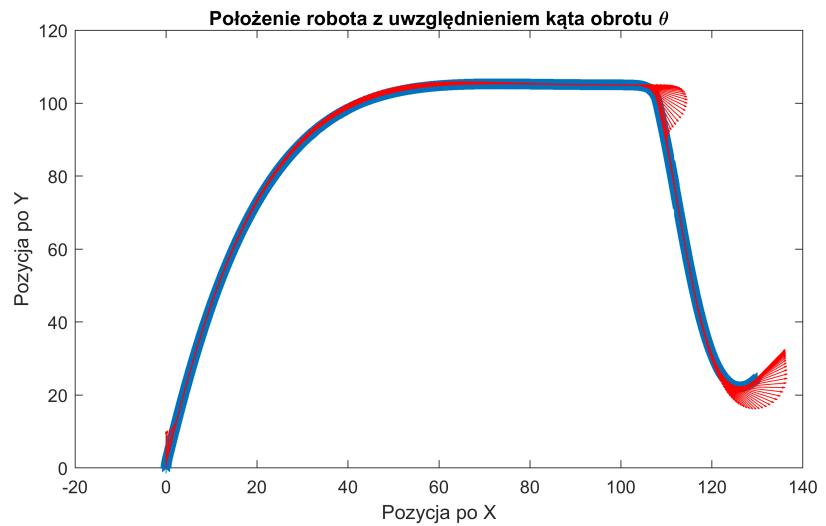
### 3.3.1. Symulacja zadanej trajektorii płaskiej

Pierwszym analizowanym przypadkiem był generator trajektorii płaskiej, aby wykazać funkcjonalność wygenerowanej trajektorii w tym przypadku zostały wygenerowane dwie trajektorie i połączono je w jedną uzyskując całą trasę. Dopiero podczas takich symulacji możliwe było jednoznaczne stwierdzenie czy rozpatrywany przypadek będzie prosty w dalszej implementacji. Na potrzeby nowej trajektorii zostały zadane nowe parametry symulacji, które widnieją w tabeli 3.3.

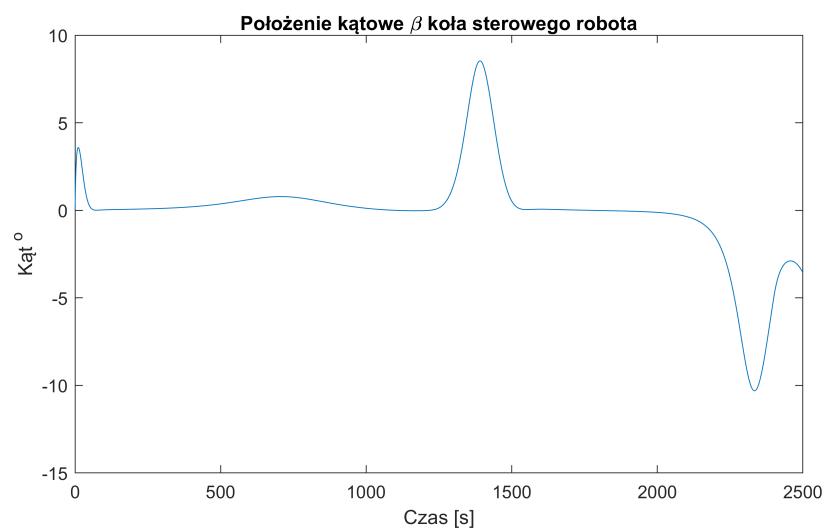
**Tabela 3.3.** Tabela zbiorcza parametrów regulatora oraz generatora trajektorii - generowanie trajektorii z symulacją na modelu

Źródło: Opracowanie własne

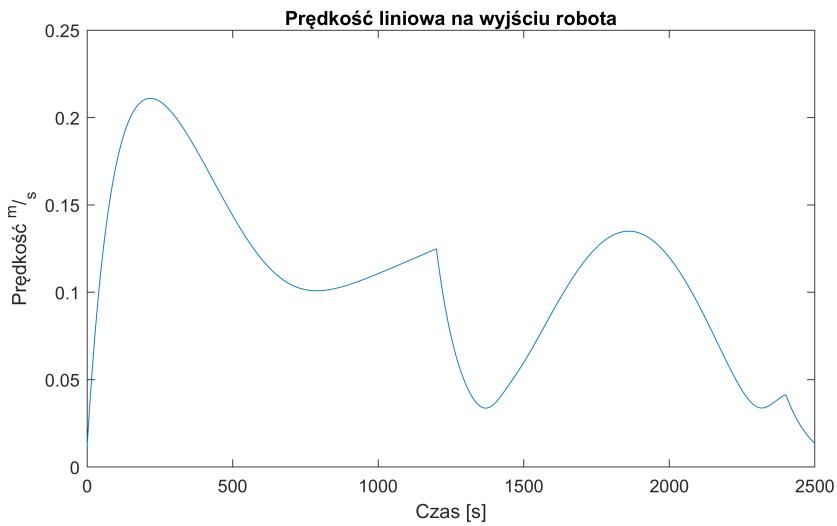
Nazwa	Parametr	Wartość
Początkowe położenie w osi X	$x_0$	0
Początkowe położenie w osi Y	$y_0$	0
Początkowe położenie kątowe robota	$\theta_0$	0
Początkowe położenie kątowe koła sterowego	$\beta_0$	0
Pośrednie położenie w osi X	$x_{T0.5}$	105
Pośrednie położenie w osi Y	$y_{T0.5}$	105
Pośrednie położenie kątowe robota	$\theta_{T0.5}$	0
Pośrednie położenie kątowe koła sterowego	$\beta_{T0.5}$	0
Końcowe położenie w osi X	$x_T$	130
Końcowe położenie w osi Y	$y_T$	25
Końcowe położenie kątowe robota	$\theta_T$	0
Końcowe położenie kątowe koła sterowego	$\beta_T$	0
Ograniczenia prędkości w osiach X i Y	$a_x, a_y$	0.1950
Czas potrzebny na realizację	$T$	2400



**Rysunek 3.6.** Symulacja trajektorii płaskiej - położenie robota z uwzględnieniem kąta obrotu  
 Źródło: Opracowanie własne



**Rysunek 3.7.** Symulacja trajektorii płaskiej - położenie koła sterowego robota  
 Źródło: Opracowanie własne



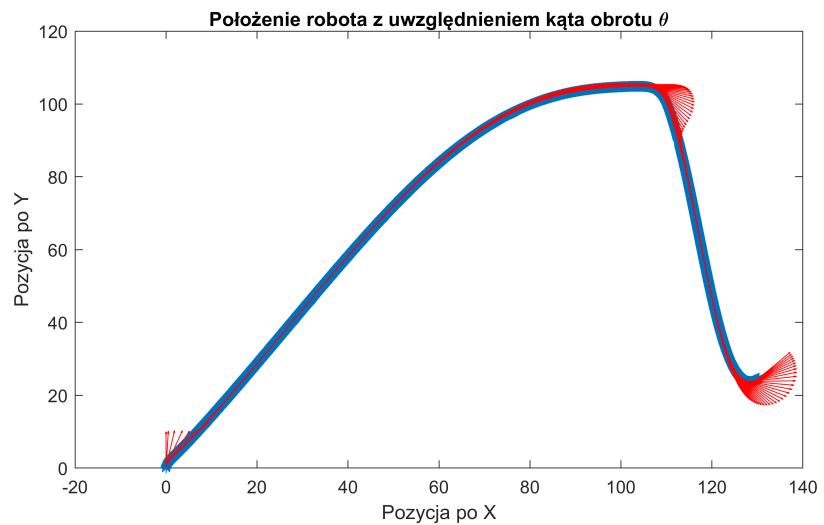
**Rysunek 3.8.** Symulacja trajektorii płaskiej - prędkość liniowa robota

Źródło: Opracowanie własne

Analizując załączone wyniki symulacji (rysunek 3.6 i rysunek 3.7) można zaobserwować, iż robot w pełni osiąga zadane położenia w osiach  $X$  i  $Y$  w zadanym czasie  $T$ , lecz nie osiąga zadanych  $\theta_T$  oraz  $\beta_T$  co jest niewątpliwie wadą tej metody. Warto również zaznaczyć, że wygenerowana trajektoria jest gładka i robot nie wykonuje nagłych ruchów. Robot dokładnie podąża za wyznaczoną trajektorią aby osiągnąć zadane położenie końcowe, jest to dość ważne podczas poruszania się w ciasnych pomieszczeniach lub w zmiennym otoczeniu robota. Kolejny rysunek 3.8 przedstawia prędkość liniową robota, która maleje wraz z dystansem potrzebnym do osiągnięcia zadanego położenia.

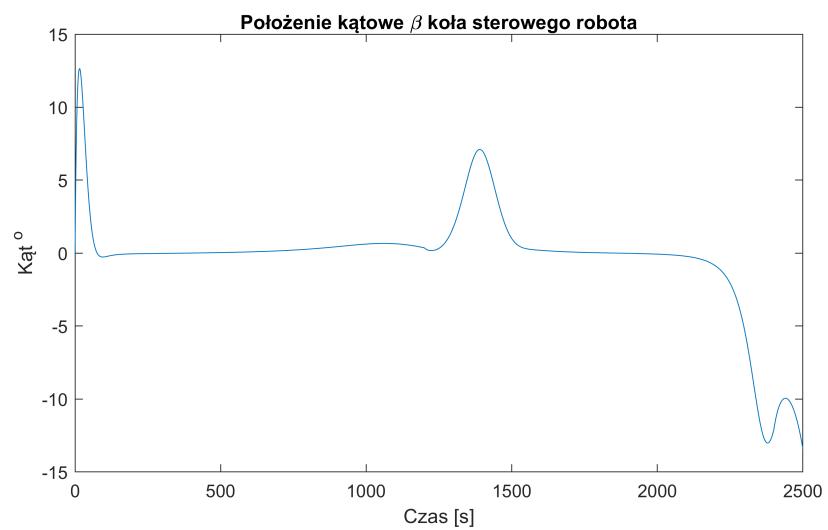
### 3.3.2. Symulacja zadanej trajektorii wielomianowej

Symulacyjne badanie wygenerowanej trajektorii wielomianowej przebiegło w takich samych krokach jak badanie trajektorii płaskiej, mianowicie zostały wygenerowane dwie trajektorie i połączono je w całość uzyskując pełną ścieżkę. Należy wspomnieć, iż otrzymana trajektoria jest optymalna w lokalnie (dwie osobne trajektorie przed połączeniem) a nie globalnie (po połączeniu). Badanie własności generatora trajektorii wielomianowej miało na celu sprawdzenie możliwości późniejszej implementacji do fizycznego obiektu. Podobnie jak w poprzednim przypadku podczas wykonywania symulacji zostały zadane dokładnie te same parametry opisane w tabeli 3.3, wyniki zostały przedstawione w dalszej części.



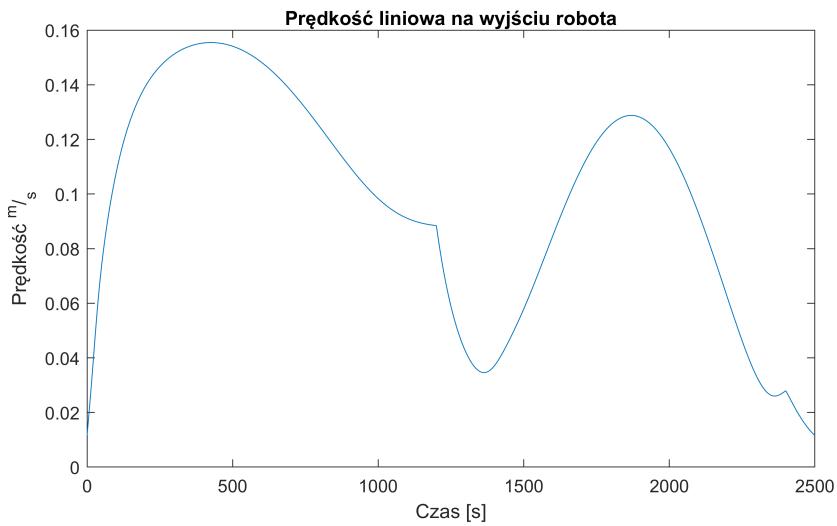
**Rysunek 3.9.** Symulacja trajektorii wielomianowej - położenie robota z uwzględnieniem kąta obrotu

Źródło: Opracowanie własne



**Rysunek 3.10.** Symulacja trajektorii wielomianowej - położenie kątowe robota

Źródło: Opracowanie własne



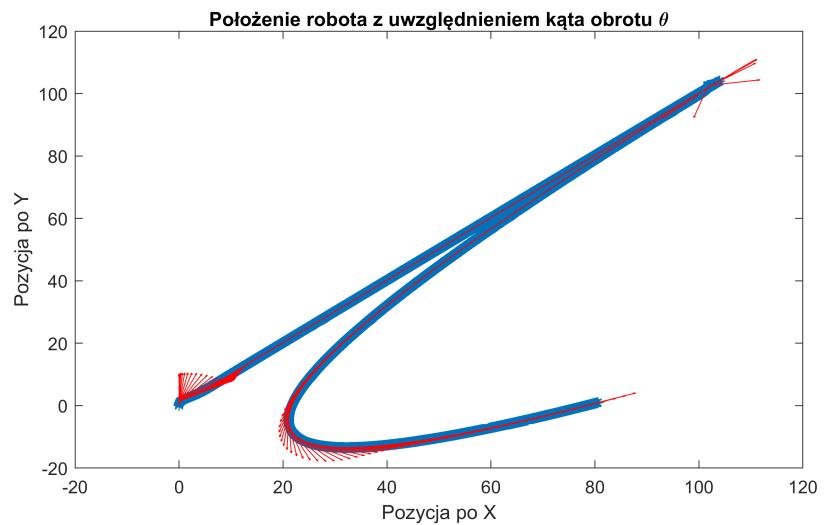
**Rysunek 3.11.** Symulacja trajektorii wielomianowej - prędkość liniowa robota

Źródło: Opracowanie własne

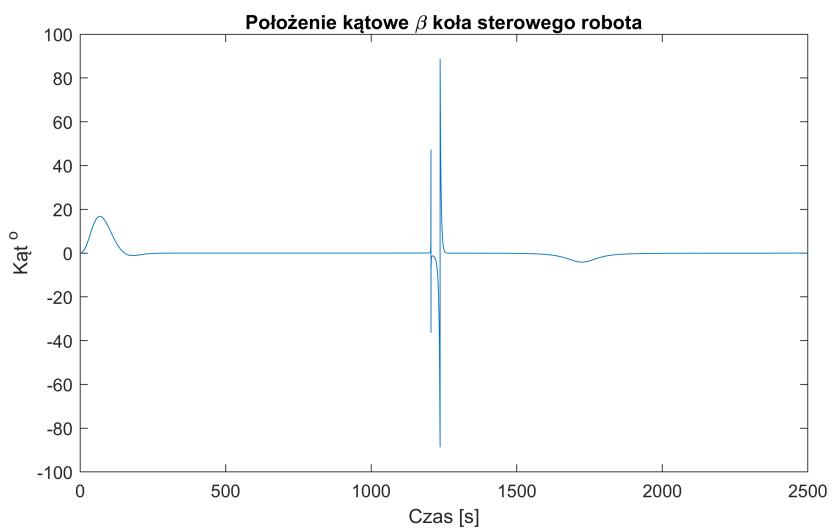
Badając załączone wyniki symulacji (rysunek 3.9 i rysunek 3.10) można zauważyć, iż robot nie osiąga w pełni zadanego położenie w osiach  $X$  i  $Y$  w zadanym czasie  $T$ , znacznie lepiej (w porównaniu do poprzedniej metody) udaje się osiągnąć położenie kątowe  $\theta_T$  jednak z pogorszeniem kąta  $\beta_T$ . Do plusów tej metody można zaliczyć dokładniejsze sterowanie położeniem kątowym robota, w przypadku zamontowania na robocie kamery bez możliwości jej obrotu konieczne jest sterowanie położeniem kątowym robota tak, aby kamera mogła uchwycić pożądany obraz. Kolejny rysunek 3.11 przedstawia prędkość liniową robota, prędkość spada wraz z coraz mniejszą odległością do punktu docelowego co jest pożądanym efektem.

### 3.3.3. Symulacja zadanej trajektorii symetrycznej-wielomianowej

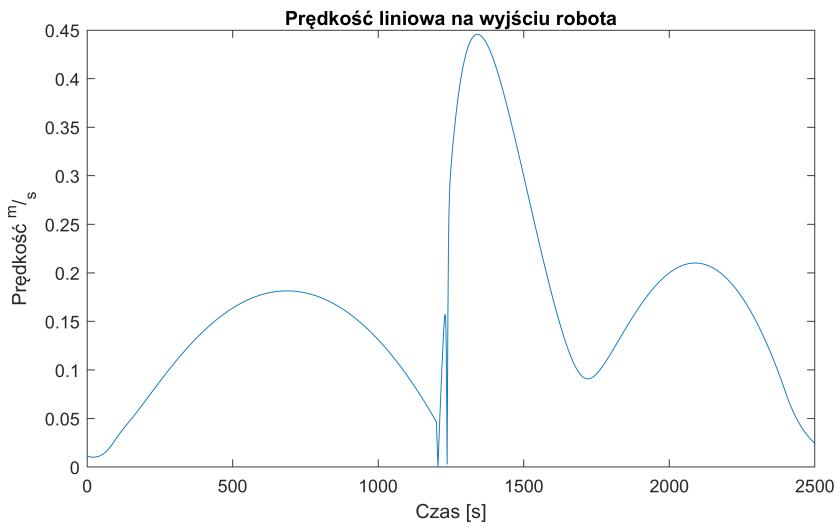
Finalną częścią prowadzonych badań symulacyjnych była wygenerowana trajektoria symetryczno-wielomianowa, podobnie jak w innych przypadkach badania miały na celu przeanalizowanie możliwości implementacji metody do robota mobilnego. Trajektoria została wygenerowana z parametry opisanymi w tabeli 3.3. Symulacyjne wyniki badań prezentują się następująco:



**Rysunek 3.12.** Symulacja trajektorii symetryczno-wielomianowej - położenie robota z uwzględnieniem kąta obrotu  
 Źródło: Opracowanie własne



**Rysunek 3.13.** Symulacja trajektorii symetryczno-wielomianowej - położenie kątowe robota  
 Źródło: Opracowanie własne



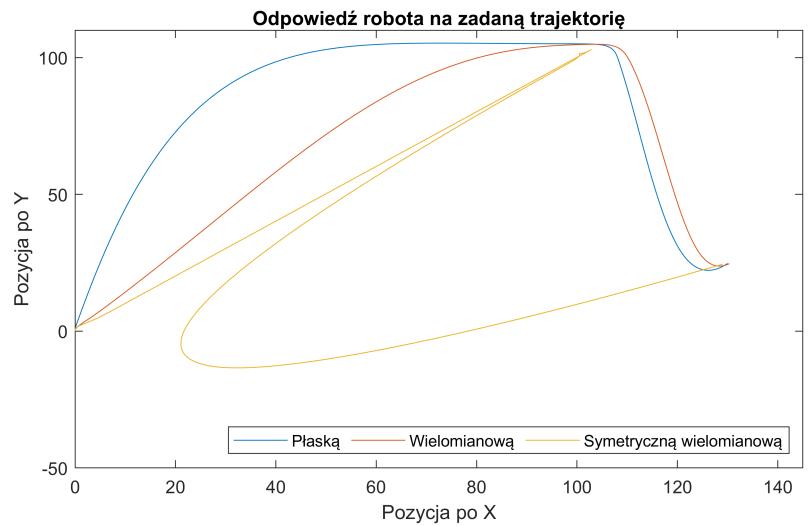
**Rysunek 3.14.** Symulacja trajektorii symetryczno-wielomianowej - prędkość liniowa robota

Źródło: Opracowanie własne

Załączone wykresy (rysunek 3.12 i rysunek 3.13) wskazują, iż robot nie osiąga zadanego położenia w osiach  $X$  i  $Y$  w zadanym czasie  $T$  jak i również położenie kątowe  $\theta_T$  oraz położenie koła sterowego  $\beta_T$  przyjmują duże wartości skokowe. Tak wygenerowana trajektoria nie dość, że nie spełnia założonych wymagań to układy wykonawcze są narażone na uszkodzenie mechaniczne w skutek skokowych zmian wartości. Wadą tej metody jest konieczność przyjmowania na początku każdej symulacji założenia, że obecny punkt położenia robota jest punktem o współrzędnych  $X = 0$  i  $Y = 0$ . W związku z tym metoda ta jest trudna do skojarzenia z globalnym układem współrzędnych, co eliminuje ją jako kandydata do implementacji w robocie mobilnym. Kolejny rysunek 3.14 przedstawia prędkość liniową robota, prędkość przyjmuje kilkukrotnie większą niż pozostałe wygenerowane trajektorie 3.3.1 czy 3.3.2. Z pewnością nagłe zmiany prędkości oraz duże przyspieszenia wpływają negatywnie na układy wykonawcze robota.

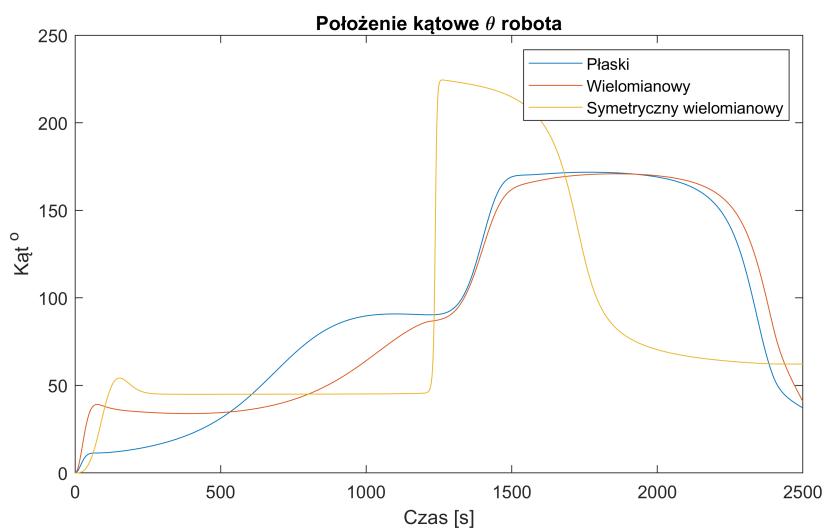
### 3.3.4. Podsumowanie badań symulacyjnych

Przeprowadzone symulacje miały na celu wskazanie pewnych własności każdej z metod generowania trajektorii, aby lepiej porównać i wybrać najlepszą z dostępnych możliwości trajektorie zostały naniesione na wspólne wykresy i prezentują się następująco:



**Rysunek 3.15.** Podsumowanie symulacji - położenie robota

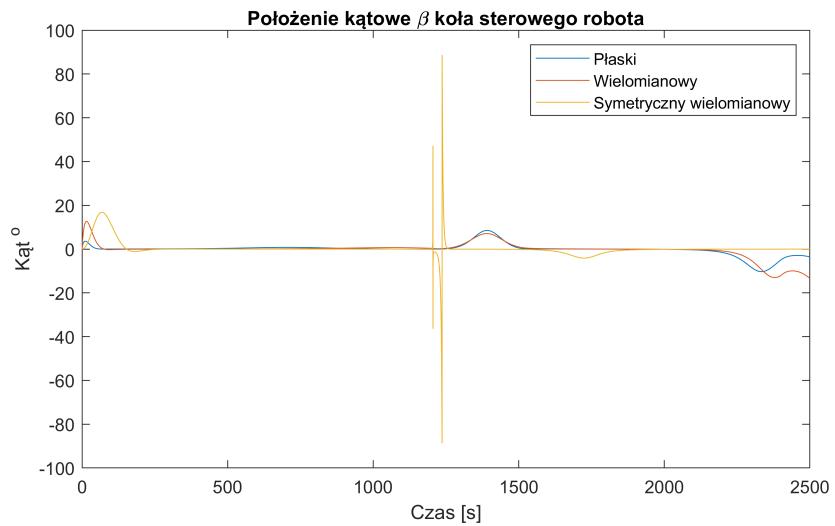
Źródło: Opracowanie własne



**Rysunek 3.16.** Podsumowanie symulacji - położenie kątowe robota

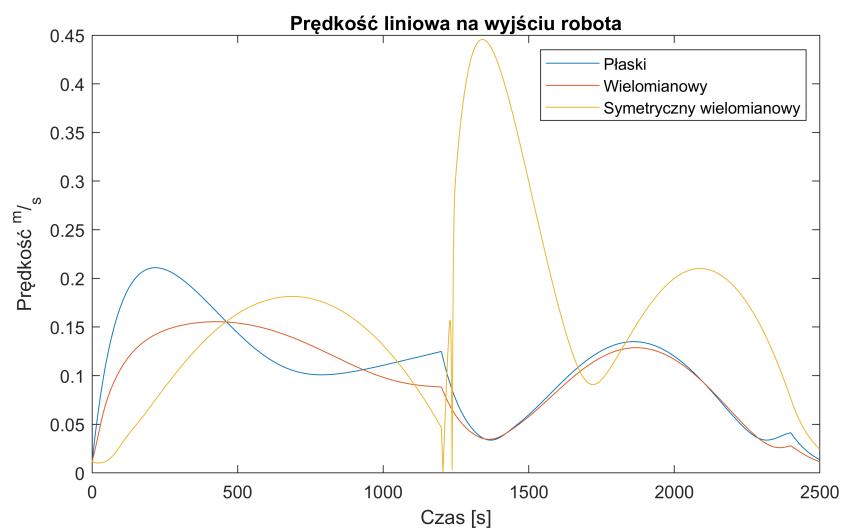
Źródło: Opracowanie własne

UKI



**Rysunek 3.17.** Podsumowanie symulacji - położenie koła sterowego robota

Źródło: Opracowanie własne



**Rysunek 3.18.** Podsumowanie symulacji - prędkość liniowa robota

Źródło: Opracowanie własne

Tak jak zostało wcześniej wspomniane metoda generowania trajektorii płaskiej oraz metoda generowania trajektorii wielomianowej spełniają warunki implementacji do robota mobilnego i w zależności czy ważniejsze jest dokładne położenie robota czy też jego położenie kątowe z mniej dokładnym położeniem, można wybrać jedną z metod. Wyniki badań wyeliminowały metodę generowania trajektorii symetryczno-wielomianowej ze względu na nie spełnienie założeń takich jak dotarcie do wyznaczonego celu czy wygenerowanie smukłej ścieżki przy możliwie minimalnym skręceniu koła sterowego oraz małych prędkościach. Konieczność dodatkowych przekształceń w celu uzyskania skojarzenia układu współrzędnych lokalnych z globalnym układem jest jedną z kolejnych wad tej metody, niewykluczone jednak, że po dodaniu odpowiednich metod sama metoda mogłaby być funkcjonalna i możliwa do implementacji w fizycznym obiekcie.

Piotr Kornaszewski

## **ROZDZIAŁ 4**

# **Implementacja wybranej metody do fizycznego obiektu**

Rozdział ten zawiera uzasadnienie wyboru konkretnej metody generowania trajektorii jak i zagadnienie implementacji wybranego generatora trajektorii oraz związaną z tym problematykę.

## **4.1. Wybrana metoda generowania trajektorii**

W poprzednich rozdziałach pracy dość szczegółowo zostały omówione zagadnienia generowania trajektorii robotów mobilnych. Generatory trajektorii zostały omówione na konkretnym przykładzie trójkółowca. Za obiekt służący przeprowadzeniu symulacji został wybrany model kinematyczny trójkółowca z napędem różnicowym, model ten pozwala na przybliżone odwzorowanie zachowanie prawdziwego robota. W celu zmniejszenia różnic między obiektem a robotem konieczne byłoby dodanie dynamiki obiektu lecz wymaga to bardziej skomplikowanych równań. Otrzymany model w pełni wystarczył do przeprowadzenia badań. Pomimo braku dynamiki w modelu przez zastosowanie regulatora z odwrotnym modelem możliwe było dodanie ograniczeń przyspieszenia w strukturze regulatora co w pewnym stopniu pozwoli wierniej odwzorować zachowanie fizycznego obiektu.

Kolejny rozdział skupiał się na teoretycznym jak i praktycznym zagadnieniu generowania trajektorii płaskiej, wielomianowej i symetryczno-wielomianowej bazując na modelu trójkółowca co miało na celu sprawdzenie uniwersalności otrzymanego układu równań kinematyki robota oraz jego układu regulacji. Załączone symulacje wykazały, iż na początku przy krótkich dwupunktowych (początek i koniec) trajektoriach wszystkie generatory poradziły sobie z zadowalającymi efektami, rozpatrując wygenerowane trajektorie pod kątem dotarcia do celu, kąta obrotu robota oraz kota obrotu koła sterowego. Symulacje wygenerowanych trajektorii już z bardziej złożoną ścieżką (początek, punkt pośredni i koniec) pozwoliły zauważyc, iż metoda symetryczno-wielomianowa nie nadaje się do prostego powiązania z globalnym układem współrzędnych co dyskwalifikuje ją na tym etapie. Dwie pozostałe

metody cechowały się pewnymi właściwościami. Generator płaski znacznie dokładniej osiągał zadane położenie w osiach X i Y ale słabiej położenie kątowe, natomiast generator wielomianowy cechował się odwrotnymi właściwościami. Wybór odpowiedniego generatora nie był oczywisty czy też prosty lecz ze względu na lepszą dokładność położenia kątowego robota został wybrany generator wielomianowy ponieważ na fizycznym robocie będzie znajdowała się kamera typu Kinect z sztywnym statywem więc dokładna pozycja kątowa jest niezbędna do odpowiedniego przetwarzania obrazu otoczenia robota. Jednakże generator płaski przy zastosowaniu dodatkowego wyposażenia w postaci ruchomego statywów typu „gimbal” może okazać się lepszy ze względu na dokładniejsze osiąganie zadanego punktu.

## **4.2. Problematyka implementacji wybranej metody generowania trajektorii oraz niezbędne modyfikacje w robocie mobilnym**

Niestety w wyniku przedłużających się prac symulacyjno-projektowych nie było możliwe terminowe dokończenie prac oraz finalne przetestowanie otrzymanych wyników badań na fizycznym obiekcie. Prowadzone prace badawcze wykazały również konieczność modyfikacji struktury układu skrętnego robota, ponieważ zastosowane rozwiązanie nie pozwalało na precyzyjne ustawienie koła sterowego. Podczas badań okazało się, iż silnik sterujący kołem sterowym wymiany z powodu na niedokładność w otrzymywanych położeniu. W celu wyeliminowania tego problemu powinno się zastosować silnik krokowy lub serwomechanizm dzięki czemu byłoby możliwe uzyskanie dokładnego położenia przy dość dużej sztywności. Kolejnym elementem byłoby dołączenie algorytmu fuzji wykorzystując kompas magnetyczny oraz akcelerometr służący do kompensacji błędu kompasu w celu pomiaru kąta obrotu robota. Do określenia poprawności działania potrzebne byłby testy środowiskowe w celu wyeliminowania zakłóceń z otoczenia wynikających z pracy urządzeń elektrycznych czy poprowadzonych przewodów w ścianach budynku. Ostatnim elementem wartym przeprowadzenia modyfikacji są regulatory silników napędowych, komunikacja z nimi następuje po przez wygenerowanie sygnału PPM (Pulse Position Modulation), wygenerowany sygnał pozwala określić przybliżoną wartość prędkości obrotowej silnika a nie jej dokładną wartość. W celu wyeliminowania tego błędu możliwe byłoby zastosowanie dodatkowych enkoderów na oś wału silnika lub stworzenie sterowania za pomocą komunikacji szeregowej.

# Zakończenie

W ramach pracy zostały omówione różne metody generowania trajektorii robotów mobilnych, metody te zostały opisane wraz z przykładami oraz możliwościami.

Następnie został opracowany model kinematyczny robota mobilnego na podstawie trójkołowego robota mobilnego wykorzystującego napęd różnicowy. Pokazane zostały właściwości uzyskanego modelu oraz jego uniwersalność przy dalszych etapach prowadzonych badań. Niezbędne okazało się zaprojektowanie regulatora bazującego na modelu odwrotnym trójkołowego robota mobilnego. Choć projektowanie regulatora nie było wymienione w zakresie pracy jego stworzenie okazało się niezbędne. Aby możliwe było wygenerowanie konkretnych sygnałów sterujących na poszczególne koła co umożliwiło poprawne przeprowadzenie badań symulacyjnych.

Dokonano analizy teoretycznej właściwości wybranych metod generowania trajektorii robotów mobilnych jak i również przeprowadzono praktyczną analizę otrzymanych generatorów wykorzystując tym celu program MATLAB 2020b, zaimplementowano algorytmy uzyskując generatorów pozwalające wygenerować pożądane trajektorie. Wyniki badań nie wskazywały jednoznacznie na przewagę którejkolwiek z metod, każda z nich cechowała się innymi właściwościami lecz wszystkie generowały trajektorie zgodnie z założeniami.

Symulacyjne badania wygenerowanych trajektorii w połączeniu z opracowanym wcześniej modelem kinematycznym robota mobilnego wykazały, iż funkcjonalność metody generowania trajektorii symetryczno-wielomianowej jest mocno ograniczona i nie pozwala na proste powiązanie lokalnego układu współrzędnych z globalnym układem przez co metoda ta została wyeliminowana. Ostatecznie wyniki badań wskazały, że metoda generowania trajektorii płaskiej uzyskuje doskonałe położenia robota w pozycji  $X$  i  $Y$  lecz znacznie słabiej radzi sobie z położeniem kątowym  $\theta$ , gdzie z kolei metoda generowania trajektorii wielomianowej nie radzi sobie tak dobrze z położeniem robota w pozycji  $X$  i  $Y$  ale znacznie lepiej uzyskuje położenie kątowe  $\theta$ . Po analizie funkcjonalności oraz dostępnym fizycznym sprzęcie został wybrany generator trajektorii wielomianowej, ponieważ położenie kątowe w przypadku późniejszej implementacji kamery na sztywnym statywie znacznie poprawi jej funkcjonalność.

Niebyvale trudnym elementem podczas wykonywania pracy okazała się obecna sytuacja pandemiczna, znaczco utrudniały wykonywanie testów oraz implementacji wybranej metody generowania trajektorii. Podczas samej implementacji okazało się, iż struktura robota wymaga przebudowy ze względu na brak dokładnej możliwości sterowania kołem sterowym. Również konieczne do ukończenia implementacji potrzebne były badania działania kompasu magnetycznego z akcelerometrem a do tego niezbędne są testy środowiskowe co też jest znacznie utrudnione. Ostatnim elementem wymagającym uwagi okazała się prędkość silników napędowych, wyliczona na podstawie zadawanego sygnału PPM nie była zbyt dokładna i wymaga zastosowania dodatkowych metod pomiarowych czy też stworzenia innej metody sterowania silnikami.

W przyszłości dalsze prace nad projektem można skupić na poprawie wcześniejszych wymienionych komponentów czy też poprawie ich funkcjonalności. Fizyczne testy mogą

znacznie poprawić pogląd na niektóre opisane rozwiązania w omawianej pracy i w związku z tym poprawienie czy ulepszenie stworzonych rozwiązań. Dodatkowym udogodnieniem mogłaby być implementacja kamery i uwzględnienie jej na etapie generowania trajektorii w celu zastosowania bardziej wyspecjalizowanych algorytmów.

Podsumowując projekt przybliża metody generowania trajektorii robotów mobilnych za pomocą modelu trójkołowca, posiada również wskazówki do dalszej pracy oraz testów badawczych. Sama praca wykazuje różne właściwości omawianych generatorów przez co wybór konkretnego jest ściśle powiązany z konkretnie omawianym przypadkiem. Dzięki badaniom możliwe jest dokładniejsze wybranie odpowiadającego algorytmu generowania trajektorii.

Piotr Kornaszewski

## **Spis tabel**

3.1. Tabela zbiorcza parametrów regulatora oraz generatora trajektorii . . . . .	32
3.2. Tabela zbiorcza parametrów regulatora oraz generatora trajektorii - dobór parametru czasu generowania trajektorii $T$ . . . . .	37
3.3. Tabela zbiorcza parametrów regulatora oraz generatora trajektorii - generowanie trajektorii z symulacją na modelu . . . . .	38

Piotr Kornaszewski

# Spis rysunków

1.1. Zasada działania metod potencjalnych: a) kształt mapy przeszkodeb, b) potencjał przyciągający do celu, c) potencjał wypadkowy, d) nałożone podpunkty b) i c), e) warstwice pola potencjalnego i znaleziona ścieżka, f) kierunki wektorów gradientu potencjału . . . . .	12
1.2. Środowisko robota wraz z wyznaczoną trasą . . . . .	13
2.1. Trójkołowy robot mobilny . . . . .	16
2.2. Trójkołowy robot mobilny z napędem różnicowym . . . . .	16
2.3. Schemat ideowy układu sterowania trójkołowym robotem mobilnym . . . . .	17
2.4. Dwukołowy robot mobilny z napędem różnicowym . . . . .	18
2.5. Robot mobilny typu samochód z sprzężonym napędem tylnej osi . . . . .	19
2.6. Trójkołowy robot mobilny z sprzężonym napędem tylnej osi . . . . .	19
2.7. Model robota w środowisku MATLAB Simulink . . . . .	20
2.8. Odpowiedź robota na zadane przykładowe parametry . . . . .	21
2.9. Przykład układu regulacji kaskadowej . . . . .	21
2.10. Sterowanie z odwrotnym modelem obiektu . . . . .	22
2.11. Regulacja nadążająca za modelem . . . . .	22
2.12. Układ trójkołowego robota mobilnego z punktem śledzenia trajektorii . . . . .	23
2.13. Odpowiedź układu regulacji na zadanie położenia końcowego . . . . .	24
2.14. Porównanie trajektorii robota dla różnych $a_x$ i $a_y$ . . . . .	24
2.15. Porównanie sygnałów sterujących robota dla różnych $a_x$ i $a_y$ . . . . .	25
3.1. Wygenerowana trajektoria płaska . . . . .	32
3.2. Wygenerowana trajektoria wielomianowa . . . . .	34
3.3. Wygenerowana trajektoria symetryczno-wielomianowa . . . . .	35
3.4. Porównanie trajektorii robota przy różnym czasie $T$ . . . . .	36
3.5. Porównanie prędkości podawanych na koła przy różnym czasie $T$ . . . . .	37
3.6. Symulacja trajektorii płaskiej - położenie robota z uwzględnieniem kąta obrotu . . . . .	39
3.7. Symulacja trajektorii płaskiej - położenie koła sterowego robota . . . . .	39
3.8. Symulacja trajektorii płaskiej - prędkość liniowa robota . . . . .	40
3.9. Symulacja trajektorii wielomianowej - położenie robota z uwzględnieniem kąta obrotu . . . . .	41
3.10. Symulacja trajektorii wielomianowej - położenie kątowe robota . . . . .	41
3.11. Symulacja trajektorii wielomianowej - prędkość liniowa robota . . . . .	42
3.12. Symulacja trajektorii symetryczno-wielomianowej - położenie robota z uwzględnieniem kąta obrotu . . . . .	43
3.13. Symulacja trajektorii symetryczno-wielomianowej - położenie kątowe robota . . . . .	43

3.14. Symulacja trajektorii symetryczno-wielomianowej - prędkość liniowa robota	44
3.15. Podsumowanie symulacji - położenie robota	45
3.16. Podsumowanie symulacji - położenie kątowe robota	45
3.17. Podsumowanie symulacji - położenie koła sterowego robota	46
3.18. Podsumowanie symulacji - prędkość liniowa robota	46

Piotr Kornaszewski

## **Spis kodów źródłowych**

3.1. Generator trajektorii płaskiej w programie MATLAB . . . . .	31
3.2. Generator trajektorii wielomianowej w programie MATLAB . . . . .	33
3.3. Generator trajektorii symetryczno-wielomianowej w programie MATLAB . . . . .	34

Piotr Kornaszewski