Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Mechatroniki i Budowy Maszyn

Kierunek studiów:

Automatyka i Robotyka

Gabriel Bracha, Dawid Pietrala

Materiały dydaktyczne do przedmiotu

KINEMATYKA I DYNAMIKA ROBOTÓW

opracowane w ramach realizacji Projektu

„Dostosowanie kształcenia

w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”

FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści

Utwór objęty licencją Creative Commons BY 4.0.

Licencja dostępna pod adresem: https://creativecommons.org/licenses/by/4.0/

1. Unified Robot Description Format (URDF)

Plik URDF (ang. Unified Robot Description Format) to plik w formacie XML używany głównie w ROS (Robot Operating System) i środowisku Matlab/Simulink do opisu budowy robota – jego geometrii, kinematyki, dynamiki oraz właściwości wizualnych i kolizyjnych [4]. Jest to swego rodzaju "schemat robota", który pozwala symulatorom(np. Matlab/Simulink), narzędziom wizualizacyjnym i algorytmom sterowania rozpoznać: jak robot wygląda, jak jest zbudowany i jakie ma ograniczenia ruchu. Plik URDF będący niejako opisem "szkieletu" robota zawiera następujące elementy: <*links*> – części robota, <*joints*> – stawy i ruch między linkami, <*inertial*> – masa i bezwładność, <*visual*>/<*collision*> – wygląd i model kolizji, <*geometry*> – kształty, wymiary, modele 3D.

1.1. Struktura pliku URDF

Plik URDF jest hierarchicznym drzewem, w którym wyróżniamy kilka podstawowych elementów:

- **<robot>** – główny element otwierający cały opis. Ma atrybut *name*, który nadaje nazwę robotowi. Poniżej pokazano przykład struktury elementy *robot*.

```
<robot name="my_robot">
  ...
</robot>
```

- **<link>** – ogniwa robota. Reprezentują sztywne elementy konstrukcji (np. segmenty ramienia, podstawa, chwytak). Każdy link może mieć: *visual* – opis wyglądu (geometria i tekstura), *collision* – model używany do obliczeń kolizji (może być uproszczony), *inertial* – masa, położenie środka ciężkości, macierz bezwładności. Poniżej pokazano przykład struktury elementy *link*.

```
<link name="base_link">
  <visual>
    <geometry>
      <box size="0.5 0.5 0.2"/>
    </geometry>
    <material name="gray">
```

```
        <color rgba="0.5 0.5 0.5 1.0"/>
      </material>
    </visual>
    <collision>
      <geometry>
        <box size="0.5 0.5 0.2"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="10.0"/>
      <origin xyz="0 0 0"/>
      <inertia ixx="0.1" ixy="0.0" ixz="0.0" iyy="0.1" iyz="0.0" izz="0.1"/>
    </inertial>
</link>
```

- **<joint>** – połączenia między ogniwami. Określają rodzaj ruchu między linkami. Atrybut *type* określa typ połączenia, np.: *fixed* – nieruchome, *revolute* – obrotowe z ograniczeniami, *continuous* – obrotowe bez ograniczeń, *prismatic* – przesuwne, *floating* – 6DOF, *planar* – ruch w płaszczyźnie. Ponadto każdy joint ma: *parent* i *child* – nazwy połączonych ogniw, *origin* – pozycja i orientacja stawu, *axis* – wektor osi ruchu, *limit* – ograniczenia ruchu (dla stawów ruchomych). Poniżej pokazano przykład struktury elementy *joint*.

```
<joint name="shoulder_joint" type="revolute">
    <parent link="base_link"/>
    <child link="upper_arm"/>
    <origin xyz="0 0 0.1" rpy="0 0 0"/>
    <axis xyz="0 0 1"/>
    <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

1.2. Roboty współpracujące – struktura kinematyczna

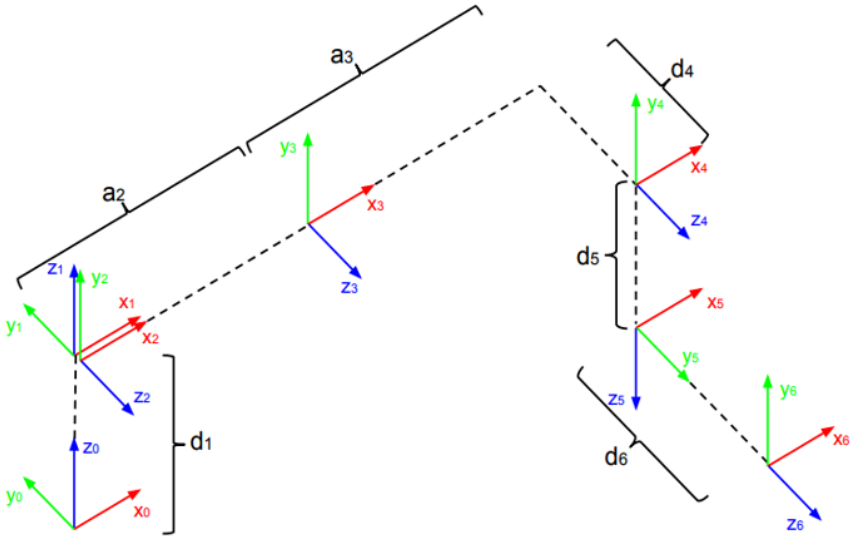
Roboty współpracujące (ang. *collaborative robots*, coboty) to urządzenia projektowane do bezpośredniej współpracy z człowiekiem, bez konieczności stosowania dużych, fizycznych barier ochronnych przy zachowaniu norm bezpieczeństwa, np. ISO/TS 15066. Przykładem takiego robota jest UR5 firmy Universal Robots. Coboty znacznie różnią się od klasycznych robotów przemysłowych, zwłaszcza w kontekście analizy kinematycznej robota. Poniżej przedstawiono najważniejsze różnice pomiędzy obydwoma rodzajami robotów.

Tabela 1 Zasadnicze różnice pomiędzy robotami współpracującymi i przemysłowymi

Cecha	**Robot współpracujący**	**Robot przemysłowy**
Bezpieczeństwo	Wbudowane czujniki momentu w każdym przegubie, ograniczenie siły i prędkości ruchu, możliwość pracy bez klatek ochronnych	Wysokie prędkości i siły, wymagane klatki/ogrodzenia, wyłączniki bezpieczeństwa
Sterowanie	Możliwość manualnego prowadzenia (tryb uczenia przez przesuwanie ramienia ręką)	Programowanie z konsoli, brak bezpośredniego prowadzenia (ze względów bezpieczeństwa)
Kinematyka	Lekka, kompaktowa konstrukcja, często otwarty łańcuch kinematyczny z mniejszą masą przegubów i mniejszym momentem bezwładności – sprzyja pracy z ograniczoną siłą	Również otwarty łańcuch kinematyczny, ale masywniejsze ogniwa i większe przekładnie – zoptymalizowane pod duże udźwigi i sztywność
Napędy	Silniki z przekładniami o dużej redukcji, ale zintegrowane czujniki momentu/siły, które pozwalają reagować na kolizje	Silniki i przekładnie zoptymalizowane pod maksymalny moment i prędkość, brak aktywnego pomiaru siły w każdym przegubie
Oprogramowanie	Prostota obsługi, skrypty wysokiego poziomu, interfejs przyjazny operatorowi	Zaawansowane języki programowania robota, większa elastyczność kosztem trudniejszej nauki
Wydajność	Niższe prędkości i przyspieszenia (normy bezpieczeństwa)	Wyższe prędkości i przyspieszenia – optymalizacja pod maksymalną wydajność

Najistotniejsze z punktu widzenia niniejszego przedmiotu są różnice wynikające z budowy kinematycznej obydwu robotów. Klasyczne roboty przemysłowe posiadają zazwyczaj sześć stopni swobody i kiść sferyczną (osie 4, 5 i 6 przecinają się w jednym punkcie). Roboty współpracujące posiadają także sześć stopni swobody, jednak cechują się równoległością osi 2, 3 i 4 i nie posiadają kiści sferycznej. Ta zasadnicza różnica wymusza inny sposób prowadzenia analizy kinematycznej takich robotów, co jest szczególnie widoczne przy analitycznym rozwiązywaniu zadania odwrotnego kinematyki. Poniżej przedstawiono schemat kinematyczny i model 3D robota UR5.

[Tekst alternatywny. Schemat kinematyczny robota UR5. Rysunek przedstawia łańcuch kinematyczny robota współpracującego z sześcioma przegubami i zaznaczonymi układami współrzędnych w każdym z nich. Widoczne są osie X, Y i Z dla układów od 0 do 6, pokazujące orientację w przestrzeni. Segmenty robota opisane są długościami a2 i a3 oraz przesunięciami d1, d4, d5 i d6 wzdłuż osi Z. Połączenia między przegubami zaznaczono liniami przerywanymi. Orientacje kolejnych układów są obrócone względem siebie, co obrazuje stopnie swobody robota. Cały rysunek przedstawia geometrię i orientację robota w przestrzeni w położeniu roboczym.]



Rys 1 Schemat kinematyczny robota UR5 [1]

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

2. Analiza kinematyczna robota współpracującego

Dysponując odpowiednio przygotowanym plikiem URDF dla danego robota, możemy bardzo łatwo dokonać jego analizy kinematycznej w zakresie kinematyki prostej, oraz analizy dynamicznej w zakresie dynamiki odwrotnej. W tym celu wykorzystamy środowisko Matlab/Simulink wraz z pakietem Simscape Multibody.

2.1. Simscape Multibody środowiska MATLAB/Simulink

Simscape Multibody to moduł środowiska MATLAB/Simulink, który służy do modelowania i symulacji mechaniki ciał sztywnych w 3D. Można go traktować jako narzędzie do tworzenia wirtualnych prototypów układów mechanicznych, gdzie zamiast pisać równania ruchu, buduje się model z gotowych elementów odwzorowujących elementy fizyczne.

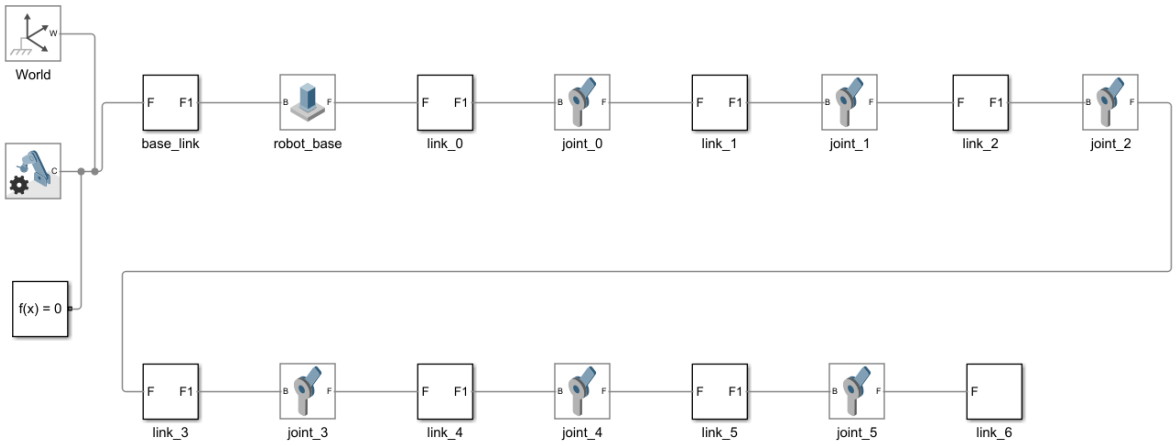
Główne cechy:

- modelowanie kinematyki i dynamiki — odwzorowanie ruchów, prędkości, przyspieszeń i sił w układach ciał sztywnych,
- środowisko blokowe — model buduje się w Simulinku z elementów takich jak: ciała sztywne (*Solid*), przeguby (*Revolute*, *Prismatic*), sprężyny, tłumiki, siły, sensory itd,
- obiekty mają masę, bezwładność, geometrię oraz wizualizację, automatyczne generowanie równań ruchu — użytkownik nie musi ręcznie liczyć układów równań różniczkowych,
- wizualizacja i animacja — wbudowany mechanizm 3D Visualization pozwala obserwować ruch modelu w czasie symulacji,
- integracja z CAD — można importować modele z SolidWorks, Inventora czy STEP/Parasolid, łącznie z właściwościami masy i geometrii, łatwe importowanie plików URDF.

Za pomocą polecenia *import Cobot.urdf* importujemy do środowiska Matlab/Simulink opis robota zawarty w pliku „Cobot.urdf” stanowiący załącznik numer 1 do niniejszego opracowania [3]. Plik „Cobot.urdf” zawiera odniesienia do modeli 3D poszczególnych członów opisywanego robota, które będą niezbędne do wykonania wizualizacji. Modele te stanowią załącznik numer 2 do niniejszego opracowania.

Po zaimportowaniu opisu robota w środowisku Matlab/Simulink otrzymujemy schemat blokowy wszystkich jego komponentów, przedstawiony na poniższym rysunku.

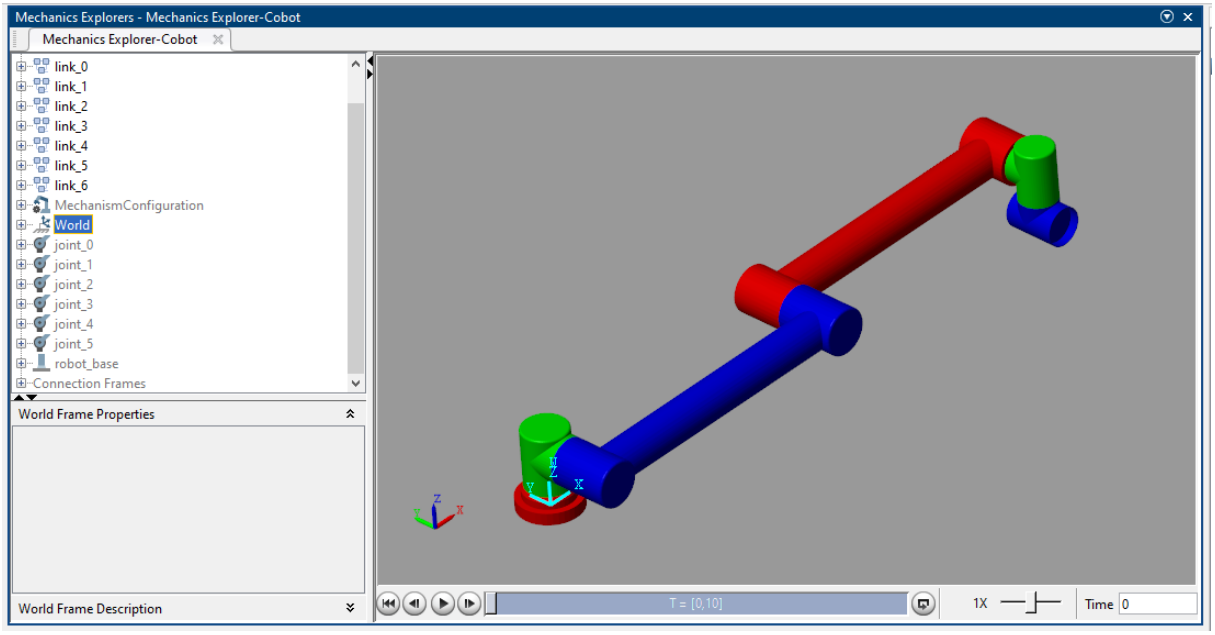
[Tekst alternatywny. Schemat blokowy zaimportowanego modelu robota w Simscape Multibody. Rysunek przedstawia schemat blokowy robota współpracującego w środowisku symulacyjnym. Łańcuch kinematyczny rozpoczyna się od elementu World, przechodzi przez base_link i robot_base, a następnie kolejne segmenty i przeguby oznaczone jako link_0, joint_0, link_1, joint_1, aż do joint_5 i link_6. Każdy joint symbolizuje przegub obrotowy, a każdy link reprezentuje człon robota. Schemat pokazuje pełne połączenie wszystkich siedmiu członów i sześciu przegubów w jedną strukturę kinematyczną. Całość obrazuje strukturę modelu robota gotowego do symulacji lub sterowania.]



Rys 2 Schemat blokowy zaimportowanego modelu robota w Simscape Multibody

Po uruchomieniu symulacji w oknie wizualizacji możemy zobaczyć graficzny podgląd robota. Przedstawiono to na poniższym rysunku.

[Tekst alternatywny. Okno wizualizacji Simscape Multibody. Rysunek pokazuje model 3D robota współpracującego w środowisku Simscape Multibody. Robot jest przedstawiony jako złożenie cylindrycznych członów połączonych w łańcuch kinematyczny z sześcioma przegubami obrotowymi. Człony mają kolor czerwony i niebieski, a przeguby są zaznaczone na zielono. Model jest widziany w lekkim nachyleniu, z widocznymi osiami układu globalnego w lewym dolnym rogu. Po lewej stronie okna znajduje się lista wszystkich elementów: linków, przegubów i bazy robota. Całość przedstawia geometryczny model robota gotowego do symulacji ruchu.]



Rys 3 Okno wizualizacji Simscape Multibody

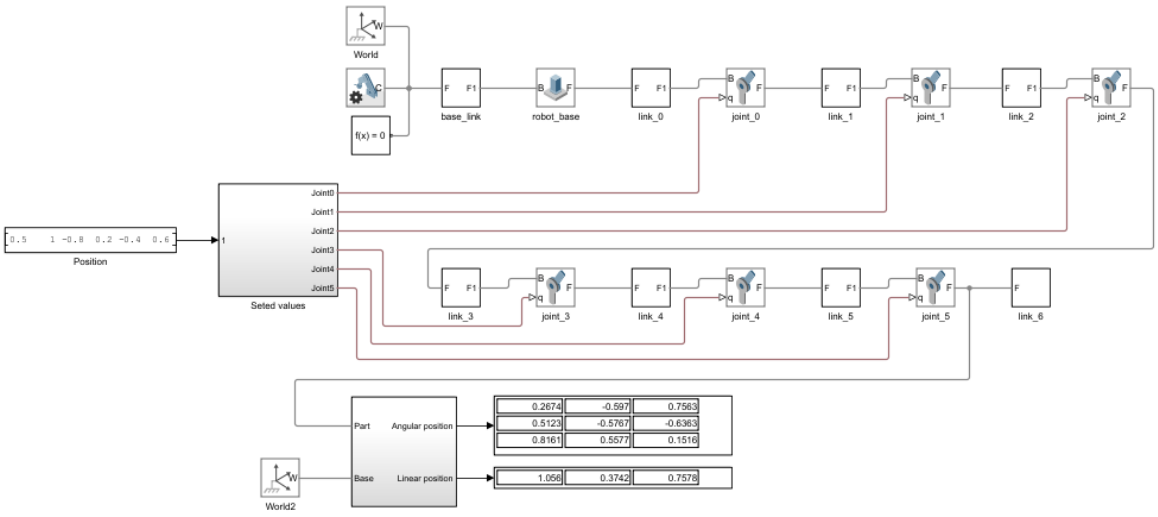
Środowisko Simscape Multibody pozwala na łatwą analizę kinematyczną robota. My jednak wykorzystamy je jedynie do weryfikacji opracowanych przez nas algorytmów rozwiązywania zadania prostego i odwrotnego kinematyki, które zostaną przedstawione w następnych rozdziałach. W tym celu wprowadzimy dane wejściowe w postaci wektora współrzędnych konfiguracyjnych $\boldsymbol{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$ (1). Po wykonaniu symulacji otrzymujemy pozycję i orientację efektora w postaci macierzy 4x4 danej wzorem (2).

$$\mathbf{q} = [0.5, 1, -0.8, 0.2, -0.4, 0.6]^{\mathrm{T}} \tag{1}$$

$$\mathbf{T} = \begin{vmatrix} 0.2674 & -0.5970 & 0.7563 & 1.0560 \\ 0.5123 & -0.5767 & -0.6363 & 0.3742 \\ 0.8161 & 0.5577 & 0.1516 & 0.7578 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{2}$$

Na poniższych rysunkach przedstawiono widok środowiska Matlab/Simulink po wprowadzeniu danych wejściowych.

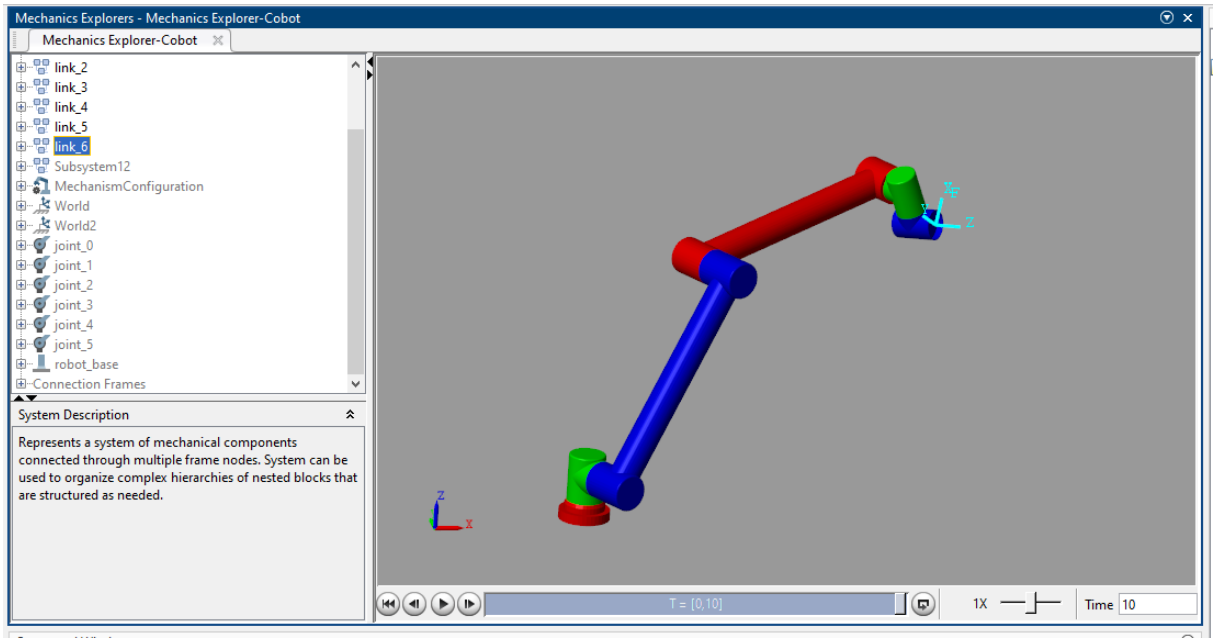
[Tekst alternatywny. Zaimportowany model robota wraz wartościami zadanymi i wyliczonymi. Rysunek przedstawia schemat blokowy robota współpracującego w środowisku symulacyjnym z dodatkowym sterowaniem pozycji przegubów. Po lewej stronie znajduje się blok „Position” z wartościami liczbowymi, które są przesyłane do bloku „Select values” i dalej do poszczególnych przegubów joint_0 do joint_5. Łańcuch kinematyczny robota jest taki sam jak wcześniej – zaczyna się od base_link i robot_base, a następnie prowadzi przez wszystkie linki i przeguby do efektora link_6. Czerwone linie pokazują połączenia sygnałowe między blokiem wejściowym a przegubami robota. Na dole schematu znajdują się dwa bloki wyjściowe pokazujące aktualną pozycję liniową i orientację efektora robota w przestrzeni. Cały układ obrazuje sposób sterowania położeniem robota oraz monitorowania jego pozycji w symulacji.]



Rys 4 Zaimportowany model robota wraz wartościami zadanymi i wyliczonymi

[Tekst alternatywny. Okno wizualizacji zaimportowanego robota dla zadanej pozycji. Rysunek przedstawia widok 3D robota w określonej pozycji roboczej. Robot jest złożony z czerwonych i niebieskich cylindrycznych członów połączonych zielonymi przegubami. Efektor robota jest uniesiony i przesunięty w prawo, co pokazuje zgięcie w kilku przegubach. Na końcu robota widoczny jest lokalny układ współrzędnych oznaczony osiami X, Y i Z. Po lewej stronie widać drzewo elementów modelu z

zaznaczonym link_6, odpowiadającym końcówce robota. Cały widok prezentuje rzeczywiste ustawienie robota w przestrzeni po zadaniu ruchu w symulacji.]



Rys 5 Okno wizualizacji zaimportowanego robota dla zadanej pozycji

Plik „Cobot.slx”, zawierający powyższe operacje, stanowi załącznik numer 3 do niniejszego dokumentu.

3. Zadanie proste kinematyki robota współpracującego

Zadanie proste kinematyki robota o sześciu stopniach swobody (6DoF) polega na wyznaczeniu pozycji i orientacji członu końcowego manipulatora w przestrzeni roboczej na podstawie znanych wartości zmiennych konfiguracyjnych przegubów i wymiarów geometrycznych robota [2]:

- **dane wejściowe** – wektor współrzędnych konfiguracyjnych $\mathbf{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^{\mathrm{T}}$ określający kąty obrotu dla przegubów obrotowych lub przemieszczenia liniowe dla przegubów pryzmatycznych.
- **model matematyczny** – zestaw macierzy transformacji opisujących geometrię manipulatora, najczęściej w postaci parametrów Denavita–Hartenberga (DH),zmodyfikowanych parametrów DH lub pliku URDF.
- **cel obliczeń** – wyznaczenie macierzy przekształcenia jednorodnego postaci ${}^0T_6 \in SE(3)$, opisującej translację i rotację układu współrzędnych członu końcowego względem układu bazowego.
- **odwzorowanie kinematyczne**: $f: \mathbb{R}^6 \rightarrow SE(3), \quad q \mapsto {}^0T_6(q)$

W celu analizy kinematycznej posłużymy się macierzami przekształceń jednorodnych postaci (3):

$$Tx(x,y,z) = \begin{matrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$Rx(a) = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & \cos a & -\sin a & 0 \\ 0 & \sin a & \cos a & 0 \\ 0 & 0 & 0 & 1 \end{matrix} \qquad (3)$$

$$Ry(\beta) = \begin{matrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

$$Rx(\gamma) = \begin{matrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Każdy joint w pliku URDF opisany jest za pomocą sześciu współrzędnych definiujących jego origin w formacie: x,y,z,roll,pitch,yaw. Zatem możemy wyznaczyć położenie układu współrzędnych danego jointa (a tym samym danego linka) jako mnożenie sześciu macierzy przekształceń jednorodnych. Ważna jest tutaj kolejność mnożenia macierzy, gdyż konwencja roll, pitch i yaw odnosi się do osi bazowych a nie bieżących. Oznacza to, że mnożenie macierzy przekształceń musi się odbywać w odwrotnej kolejności: yaw, pitch i roll. Przedstawiono to poniższym równaniem.

$$T_{origin,i} = Tx(x_i)\ Ty(yi)\ Tz(z_i)\ Rz(yaw_i)\ Ry(pitch_i)\ Rx(roll_i) \quad (4)$$

Następnie należy uwzględnić współrzędną konfiguracyjną danego przegubu. W tym celu mnożymy otrzymaną macierz przez macierz obrotu wokół osi Z (w naszym przypadku wszystkie jointy wykonują obrót wokół własnych osi Z). Przedstawiono to równaniem (5)

$$T_i^{i-1}(q_i) = T_{origin,i}\ Rz(q_i) \quad (5)$$

Następnie możemy wyznaczyć końcowe równanie opisujące pozycję i orientację efektora naszego cobota na podstawie danych w liku urdf i współrzędnych konfiguracyjnych.

$$\boldsymbol{T_N^0(q)} = \prod_{i=1}^{N} (T_{origin,i}\ Rz(q_i)) \quad (5)$$

Dysponując powyższymi informacjami możemy przystąpić do opracowania funkcji w środowisku Matlab pozwalającej rozwiązać zadanie proste kinematyki dla robota opisanego w naszym pliku URDF. W tym celu przygotujemy cztery funkcje pomocnicze: „*HT.m*”, „*HRX.m*”, „*HRY.m*”, „*HRZ.m*” służące do tworzenia macierzy przekształceń jednorodnych, funkcję „*fk_urdf.m*” rozwiązująca zadanie proste kinematyki oraz funkcję „*fk_urdf_example.m*” będącą przykładem zastosowania. Funkcje te stanowią załącznik numer 4 do niniejszego opracowania.

```
function T = transl(x,y,z)
    T = eye(4);
    T(1:3,4) = [x;y;z];
end
function T = trotx(a)
    T = [1 0 0 0; 0 cos(a) -sin(a) 0; 0 sin(a)  cos(a) 0; 0 0 0 1];
end
function T = troty(b)
    T = [cos(b) 0 sin(b) 0; 0 1 0 0; -sin(b) 0 cos(b) 0; 0 0 0 1];
```

```
end
function T = trotz(c)
    T = [cos(c) -sin(c) 0 0; sin(c)  cos(c) 0 0; 0 0 1 0; 0 0 0 1];
end
function T_final = fk_urdf(joints, q)
% joints: Nx6 [x, y, z, roll, pitch, yaw] from URDF (meters, radians)
% q: Nx1 vector of joint variables (rad or m)
% Returns: T_final (4x4) pose of end-effector in base frame
    N = size(joints,1);
    T_final = eye(4);
    for i = 1:N
        x = joints(i,1); y = joints(i,2); z = joints(i,3);
        r = joints(i,4); p = joints(i,5); yv = joints(i,6);
        T_origin = transl(x,y,z) * trotz(yv) * troty(p) * trotx(r);
        T_joint = trotz(q(i));       % rotation about local Z
        T_final = T_final * T_origin * T_joint;
    end
end
function T = fk_urdf_example(q)
% Example usage: T = fk_urdf_example(q) --- Actuated joints in kinematic order -
joints = zeros(6,6);
joints(1,:) = [0 0 0.056 0 0 0];
joints(2,:) = [0 0 0.0799 1.57079633 0 0];
joints(3,:) = [0.694 0 0 0 0 0];
joints(4,:) = [0.675 0 0.1155 0 0 0];
joints(5,:) = [0 -0.1155 0 1.57079633 0 0];
joints(6,:) = [0 0.0675 0 -1.57079633 0 0];
```


Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

```
T = fk_urdf(joints, q);
```

Możemy zatem przetestować działanie i porównać otrzymane wyniki z symulacją w Simscape Multibody. W tym celu tworzymy w środowisku Matlab wektor **q** o następujących wartościach:

$$\mathbf{q} = [0.5, 1, -0.8, 0.2, -0.4, 0.6]^{\mathrm{T}} \tag{6}$$

Po uruchomieniu skryptu otrzymujemy w wyniku następującą macierz. Jest ona zgodna z wynikami symulacji, co potwierdza poprawność naszych obliczeń.

$$\mathbf{T} = \begin{vmatrix} 0.2674 & -0.5970 & 0.7563 & 1.0555 \\ 0.5123 & -0.5767 & -0.6363 & 0.3742 \\ 0.8161 & 0.5577 & 0.1516 & 0.7578 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{7}$$

4. Zadanie odwrotne kinematyki

Zadanie odwrotne kinematyki (ang. *Inverse Kinematics, IK*) dla robota o sześciu stopniach swobody (6DoF) polega na wyznaczeniu wartości zmiennych konfiguracyjnych robota – zazwyczaj kątów obrotu przegubów lub przesunięć osi liniowych – na podstawie żądanej pozycji i orientacji efektora końcowego w przestrzeni roboczej oraz wymiarów geometrycznych. W przypadku robota 6DoF oznacza to obliczenie wektora $\boldsymbol{q} = [q_1, q_2, q_3, q_4, q_5, q_6]^T$, który spełnia równanie kinematyki prostej:

$$\mathbf{T}(\mathbf{q}) = \boldsymbol{T_{zad}} \tag{7}$$

gdzie:

- $\mathbf{T}(\mathbf{q})$ – macierz przekształcenia jednorodnego opisująca pozycję i orientację efektora względem układu bazowego,
- $\boldsymbol{T_{zad}}$ – macierz docelowa wynikająca z wymagań zadania.

Zadanie odwrotne jest istotnie trudniejsze niż zadanie proste kinematyki, ponieważ: może mieć wiele rozwiązań (różne konfiguracje przegubów prowadzą do tej samej pozycji efektora), może mieć brak rozwiązań (gdy pozycja lub orientacja leży poza przestrzenią roboczą robota), wymaga rozważenia ograniczeń mechanicznych oraz uniknięcia osobliwości kinematycznych, które powodują utratę stopni swobody w

pewnych konfiguracjach. Metody rozwiązywania zadania odwrotnego kinematyki dla robota 6DoF można podzielić na:

- analityczne – polegające na algebraicznym wyprowadzeniu równań dla wszystkich przegubów, zwykle możliwe tylko dla specyficznych struktur robota (np. manipulatory o nadgarstku sferycznym, roboty współpracujące),
- numeryczne – oparte na iteracyjnych algorytmach optymalizacyjnych lub pseudoodwrotności macierzy Jacobiego, pozwalające na obsługę dowolnych konfiguracji, ale wymagające większej mocy obliczeniowej.

Bazą dla naszych rozważań będzie schemat kinematyczny przedstawiony na rysunku 1. Na początek musimy odczytać z pliku URDF wymiary geometryczne zaznaczone na schemacie: d1, a2, a3, d4, d5, d6. Następnie przyjmujemy, że dane wejściowe będą podawane w postaci macierzy 4x4.

Wyznaczamy centrum nadgarstka, czyli pozycję punktu na osi z_6 według wzoru (8). Punkt ten jest potrzebny do rozwiązania θ_1 oraz później łańcucha 2–3–4.

$$\mathrm{p}_{05}^{0} = \mathrm{p}_{06}^{0} - \mathrm{d}_6 \mathrm{z}_6^{0} = [\mathrm{px}, \mathrm{py}, \mathrm{pz}]^{\mathrm{T}} - \mathrm{d}_6 \mathrm{a} \tag{8}$$

Następnie z rzutu na płaszczyznę XY wyznaczamy kąt θ_1 za pomocą równania (9). Otrzymujemy dwa rozwiązania: „prawe” i „lewe”.

$$\begin{aligned} \mathrm{R} &= \mathrm{sqrt}((\mathrm{p}_{05}^{0})_{\mathrm{x}}{}^{2} + (\mathrm{p}_{05}^{0})_{\mathrm{y}}{}^{2}) \\ \theta_1 &= \mathrm{atan2}\left((\mathrm{p}_{05}^{0})_{\mathrm{y}}, (\mathrm{p}_{05}^{0})_{\mathrm{x}}\right) \pm \mathrm{acos}\left(\frac{\mathrm{d}^4}{R}\right) + \frac{\pi}{2} \end{aligned} \tag{9}$$

Dalej wyznaczamy zgięcie nadgarstka. Dla każdego otrzymanego θ_1 wyznaczamy θ_5 według zależności (10).

$$\theta_5 = \pm \cos^{-1}\left(\frac{p_x \sin\theta_1 - p_y \cos\theta_1 - d_4}{d_6}\right) \tag{10}$$

Następnie wyznaczamy obrót efektora. W tym celu korzystamy z następującego wzoru:

Politechnika Świętokrzyska Kielce University of Technology

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

$$\theta_6 = \text{atan2}\left(\frac{-y_x \sin\theta_1 - y_y \cos\theta_1}{\sin\theta_5}, \frac{-(-x_x \sin\theta_1 - x_y \cos\theta_1)}{\sin\theta_5}\right) \tag{11}$$

gdzie x i y to składowe kolumn x i y macierzy obrotu opisującej orientacje efektora względem układu bazy, czyli pierwszej i drugiej kolumny macierzy zadanej.
W osobliwości $\sin\theta_5 = 0$ („wrist flip") układ jest niedookreślony. Pozostałe trzy przeguby zachowują się jak klasyczny robot płaszczyznowy 3R. Najpierw rzutujemy punkt p_{05}^0 na układ powstały po obrocie bazy o kąt θ_1 wokół osi z i odejmujemy odległość d_1w osi z.

$$\begin{gathered} x_1 = \cos\theta_1 \, (\mathrm{p}_{05}^0)_\mathrm{x} + \sin\theta_1 (\mathrm{p}_{05}^0)_\mathrm{Y} \\ z_1 = (\mathrm{p}_{05}^0)_\mathrm{z} - \mathrm{d}_1 \\ r = \sqrt{(x_1^2 + z_1^2)} \\ D = \frac{r^2 - a_2^2 - a_3^2}{2a_2 a_3} \\ \theta_3 = atan2(\pm\sqrt{1 - D^2}, D) \\ \theta_2 = atan2(z_1, x_1) - atan2(a_3 \sin\theta_3, a_2 + a_3 \cos\theta_3) \end{gathered} \tag{12}$$

W ten sposób otrzymujemy konfiguracje „elbow up/elbow down" (zależnie od znaku przy pierwiastku). Teraz możemy wyznaczyć orientację łokcia według poniższych zależności.

$$\begin{gathered} \sin\theta_{234} = \frac{-a_z}{\sin\theta_5} \\ cos\,\theta_{234} = \frac{-\cos\theta_1 \, a_x + \sin\theta_1 \, a_y}{\sin\theta_5} \\ \theta_{234} = \mathrm{a}atan2 \sin\theta_{234}, \cos\theta_{234} \\ \theta_4 = \theta_{234} - \theta_2 - \theta_3 \end{gathered} \tag{13}$$

Finalnie otrzymujemy 8 rozwiązań zadania odwrotnego kinematyki. Należy poddać je weryfikacji pod kątem ograniczeń mechanicznych, osobliwości robota i zadanej konfiguracji.

W ramach niniejszych materiałów opracowano funkcję „*ik_urdf.m*”, „*ik_urdf_example.m*” , wraz z funkcjami pomocniczymi, dla środowiska Matlab pozwalające rozwiązać zadanie odwrotne kinematyki dla robota współpracującego, według powyższego schematu. Funkcje te stanowią załącznik numer 5 do niniejszego dokumentu. Poniżej przedstawiono sposób wykorzystania opracowanego algorytmu. Zaczynamy od przygotowania danych wejściowych w postaci macierzy 4x4. Najlepiej w tym celu przygotować wektor **q** i na jego podstawie za pomocą kinematyki prostej wyznaczyć macierz położenia. Dalej przygotowujemy wymiary geometryczne. Na podstawie pliku URDF określamy wymiary d1, a2, a3, d4, d5, d6. Są one na stałe zapisane w funkcji „*ik_urdf_example.m*”. Następnie uruchamiamy tę funkcję przekazując do niej jako parametr macierz zadaną.

```
q = [0.5 1 -0.8 0.2 -0.4 0.6];
T06=fk_urdf_example(q)
Q=ik_urdf_example(T06);
```

W wyniku działania otrzymujemy zestaw do 8 rozwiązań rzeczywistych zadania odwrotnego kinematyki (14). Otrzymane rozwiązania należy poddać analizie od kątem ograniczeń mechanicznych. Poniżej przedstawiono wynik działania tej funkcji. Można zauważyć w czwartym wierszu rozwiązanie zgodne z wektorem wejściowym. Potwierdza to poprawność naszych obliczeń. Pozostałe rozwiązania stanowią inne konfiguracje robota.

$$\mathbf{Q} = \begin{vmatrix} 0.5000 & -0.0185 & 0.8937 & 2.6664 & 0.4000 & -2.5416 \\ 0.5000 & 0.8619 & -0.8937 & -2.7098 & 0.4000 & -2.5416 \\ 0.5000 & 0.2117 & 0.8000 & -0.6117 & -0.4000 & 0.6000 \\ 0.5000 & 1.0000 & -0.8000 & 0.2000 & -0.4000 & 0.6000 \\ -2.8544 & 2.1720 & 0.7444 & 3.0954 & 2.5402 & 0.7459 \\ -2.8544 & 2.9056 & -0.7444 & -2.4325 & 2.5402 & 0.7459 \\ -2.8544 & 2.2547 & 0.9425 & -0.3269 & -2.5402 & -2.3957 \\ -2.8544 & -3.1002 & -0.9425 & 0.6297 & -2.5402 & -2.3957 \end{vmatrix} \quad (14)$$

5. Literatura

1. Hawkins K. P., Analytic Inverse Kinematics for the Universal Robots UR-5/UR-10 Arms, (2013)
2. Spong M. W., Vidyasagar M., Dynamika i sterowanie robotów (1997)
3. https://www.mathworks.com/help/sm/ug/urdf-model-import.html
4. https://wiki.ros.org/urdf

6. Spis załączników

1. Plik konfiguracyjny Cobot.urdf
2. Modele 3D elementów opracowanego robota w postaci plików *.stl
3. Plik środowiska Matlab/Simulink Cobot.slx
4. Funkcje dla środowiska Matlab do rozwiązywania zadań prostego i odwrotnego kinematyki

Wszystkie powyższe załącznki można pobrać ze strony internetowej autora https://staff.tu.kielce.pl/dpietrala