



# **Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie**

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,  
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

## **Sprawozdanie Robotyka Kosmiczna**

Autor:

**Dawid Lisek**

Nr indeksu: 402382

Kierunek studiów: **Automatyka i Robotyka**

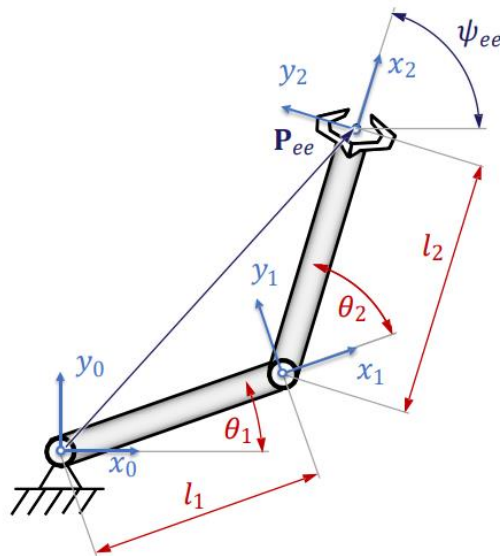
Specjalizacja: **Komputerowe Systemy Sterowania**

Grupa: pn 13:15 - 15:45

## 1. Cel ćwiczenia.

Celem ćwiczenia było rozwiązanie zadania prostego kinematyki na poziomie położeń oraz prędkości. Należało wyprowadzić analityczne zależności na pozycję, orientację, prędkość liniową oraz prędkość kątową. Następnie należało rozwiązać odwrotne zadanie kinematyki. Całość została zrealizowana w programie Matlab.

## 2. Analityczne zależności na pozycję i orientację członu roboczego w funkcji pozycji kątowych przegubów.



Dane:

$$l_1 = 0,6m$$

$$l_2 = 0,5m$$

$$m_1 = 2,5kg$$

$$m_2 = 1,5kg$$

Rysunek 1 Schemat manipulatora 2DoF z danymi o członach

Na podstawie powyższego modelu manipulatora płaskiego należało wyprowadzić wzory na pozycję oraz orientację członu roboczego w funkcji pozycji kątowych przegubów  $\theta_1 + \theta_2$ . Wyprowadzone wzory:

Równanie 1 Wyprowadzone wzory na pozycję i orientację członu roboczego

$$x_{ee} = \cos(\theta_1) * L_1 + \cos(\theta_1 + \theta_2) * L_2$$

$$y_{ee} = \sin(\theta_1) * L_1 + \sin(\theta_1 + \theta_2) * L_2$$

$$\psi_{ee} = \theta_1 + \theta_2$$

Gdzie:

$x_{ee}$  - pozycja członu roboczego na osi x,

$y_{ee}$  - pozycja członu roboczego na osi y,

$\psi_{ee}$  - orientacja członu roboczego,

### 3. Analityczne zależności na prędkość liniową i kątową członu roboczego w funkcji prędkości kątowych i pozycji kątowych przegubów.

Następnie na podstawie powyższych równań należało wyprowadzić wzory na prędkość liniową oraz kątową członu roboczego. W tym celu należało obliczyć poniższe pochodne cząstkowe:

*Równanie 2 Wzory na prędkości liniowe członu roboczego*

$$\begin{aligned}\frac{\partial x_{EE}}{\partial t} &= \frac{\partial x_{EE}}{\partial \theta_1} \frac{\partial \theta_1}{\partial t} + \frac{\partial x_{EE}}{\partial \theta_2} \frac{\partial \theta_2}{\partial t} \\ \frac{\partial y_{EE}}{\partial t} &= \frac{\partial y_{EE}}{\partial \theta_1} \frac{\partial \theta_1}{\partial t} + \frac{\partial y_{EE}}{\partial \theta_2} \frac{\partial \theta_2}{\partial t}\end{aligned}$$

Ostatecznie wzory na prędkość liniową oraz kątową członu roboczego przyjmują poniższą postać:

*Równanie 3 Wyprowadzone wzory na prędkości liniowe oraz prędkość kątową członu roboczego*

$$\begin{aligned}\dot{x}_{ee} &= -L_1 * \sin(\theta_1) * \dot{\theta}_1 - L_2 * \sin(\theta_1 + \theta_2) * (\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{y}_{ee} &= L_1 * \cos(\theta_1) * \dot{\theta}_1 + L_2 * \cos(\theta_1 + \theta_2) * (\dot{\theta}_1 + \dot{\theta}_2) \\ \dot{\psi} &= \dot{\theta}_1 + \dot{\theta}_2\end{aligned}$$

### 4. Skrypt źródłowy funkcji "direct\_2DoF" napisany w programie Matlab

Kolejnym krokiem było napisanie funkcji direct\_2DoF, która realizowała zadanie proste kinematyki. Funkcja na podstawie pozycji kątowych przegubów, prędkości kątowych przegubów oraz długości ramion manipulatora powinna zwracać pozycję, orientację, prędkość liniową oraz prędkość kątową członu roboczego. Funkcja:

```

function [Pee, Psiee, Vee, Omee] = direct_2DoF(q, dq, L1, L2)

% q - dwuelementowy wektor zawierający pozycje katowe przegubów [rad]
% dq - dwuelementowy wektor zawierający prędkości katowe przegubów[rad/s].
% L1, L2 - długości członów manipulatora [m]
% Pee - dwuelementowy wektor zawierający składową X i składową Y pozycji członu roboczego [m].
% Psiee - orientacja członu roboczego [rad].
% Vee - dwuelementowy wektor zawierający prędkość liniową członu roboczego [m/s]
% Omee - prędkość katowa członu roboczego [rad/s].
Pee = [cos(q(:, 1)) .* L1 + cos(q(:,1) + q(:,2)) .* L2, sin(q(:,1)) .* L1 + sin(q(:,1) + q(:,2)) .* L2];
Psiee = q(:,1) + q(:,2);
Vee = [-L1 .* sin(q(:,1)) .* dq(:,1) - L2 .* sin(q(:,1)+q(:,2)).*(dq(:,1)+dq(:,2)), ...
        L1 .* cos(q(:,1)) .* dq(:,1) + L2 .* cos(q(:,1) + q(:,2)) .* (dq(:,1) + dq(:,2))];
Omee = dq(:,1) + dq(:,2);
end

```

*Listing 1 Skrypt źródłowy funkcji "direct\_2DoF"*

Argument  $q$  oraz  $dq$  był przekazywany jako 2 kolumnowy wektor, który zawierał pozycję oraz prędkość kątową poszczególnych przegubów. Dodatkowo do obliczeń zostało wykorzystane mnożenie tablicowe dostępne w pakiecie Matlab.

## 5. Wybrane parametry wejściowe funkcji „Trajectory\_Generation” dla dwóch przegubów manipulatora oraz wykres wygenerowanej trajektorii (przebiegi pozycji katowych i prędkości katowych przegubów).

Następnie należało wywołać funkcję „Trajectory\_Generation” oraz wygenerować trajektorię pozycji oraz prędkości katowych przegubów manipulatora:

Funkcja Trajectory\_Generation została wywołana dla dwóch przegubów z poniższymi argumentami:

- Dane wejściowe dla przegubu nr 1

```

q_p_1 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_1 = 0.0;        % prędkość początkowa przegubu nr 1 [rad]
q_k_1 = 1/2 * pi;     % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_1 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_1 = 10;           % Czas trwania ruchu przegubu [s]
Ta_1 = 4;            % Czas trwania fazy przyspieszania [s]
Tb_1 = 4;            % Czas trwania fazy hamowania [s]
dt_1 = 0.01;         % Krok czasowy [s]
V_1 = 0.2;           % Prędkość przegubu w fazie drugiej [rad/s]
[q_1,dq_1,ddq_1, t_1] = Trajectory_Generation(solved, q_p_1, dq_p_1,...
q_k_1, dq_k_1, Tk_1, Ta_1, Tb_1, dt_1, V_1);

```

*Listing 2 Dane wejściowe funkcji Trajectory\_Generation dla przegubu nr 1*

- Dane wejściowe dla przegubu nr 2

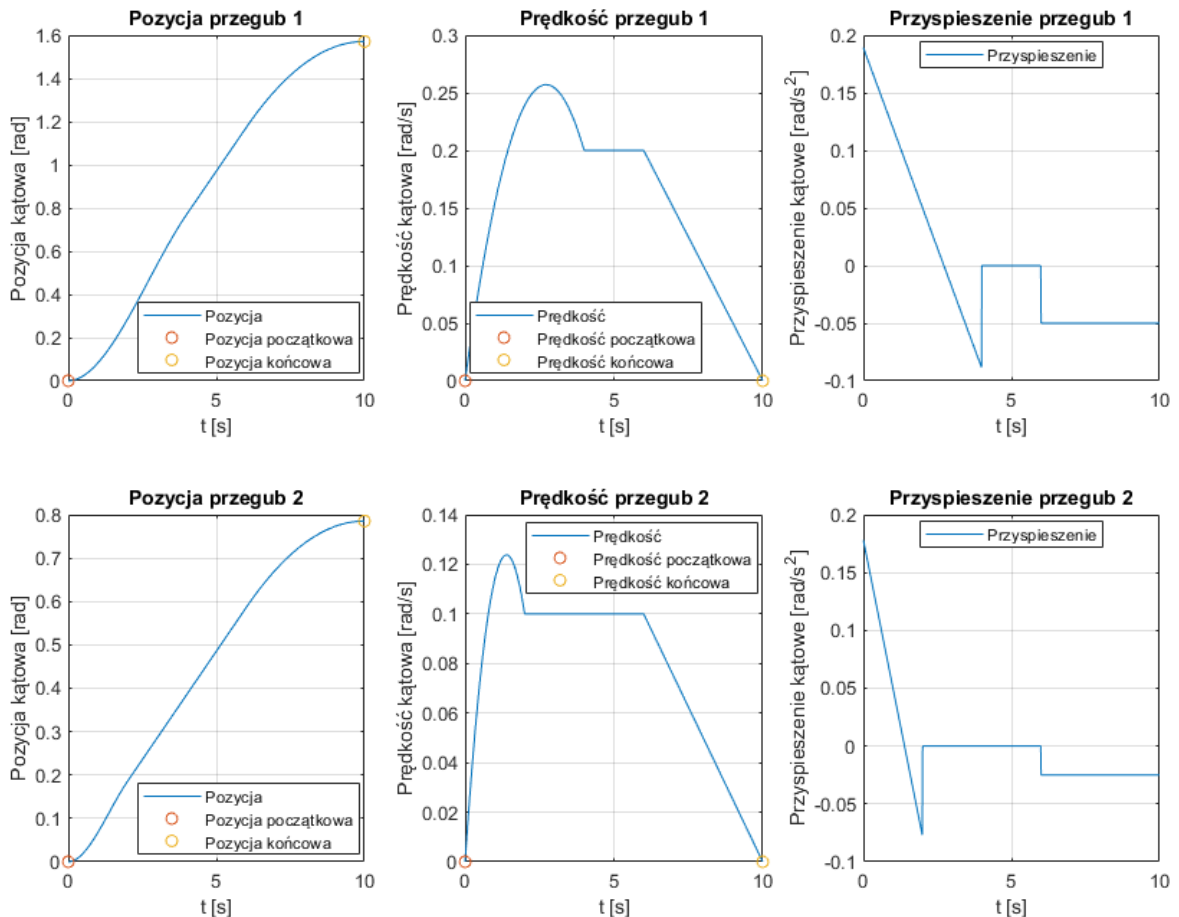
```

q_p_2 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_2 = 0.0;        % prędkość początkowa przegubu nr 1 [rad/s]
q_k_2 = 1/4 * pi;     % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_2 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_2 = 10;           % Czas trwania ruchu przegubu [s]
Ta_2 = 2;            % Czas trwania fazy przyspieszania [s]
Tb_2 = 4;            % Czas trwania fazy hamowania [s]
dt_2 = 0.01;         % Krok czasowy [s]
V_2 = 0.1;           % Prędkość przegubu w fazie drugiej [rad/s]
[q_2,dq_2,ddq_2, t_2] = Trajectory_Generation(solved, q_p_2, dq_p_2, ...
q_k_2, dq_k_2, Tk_2, Ta_2, Tb_2, dt_2, V_2);

```

Listing 3 Dane wejściowe funkcji Trajectory\_Generation dla przegubu nr 2

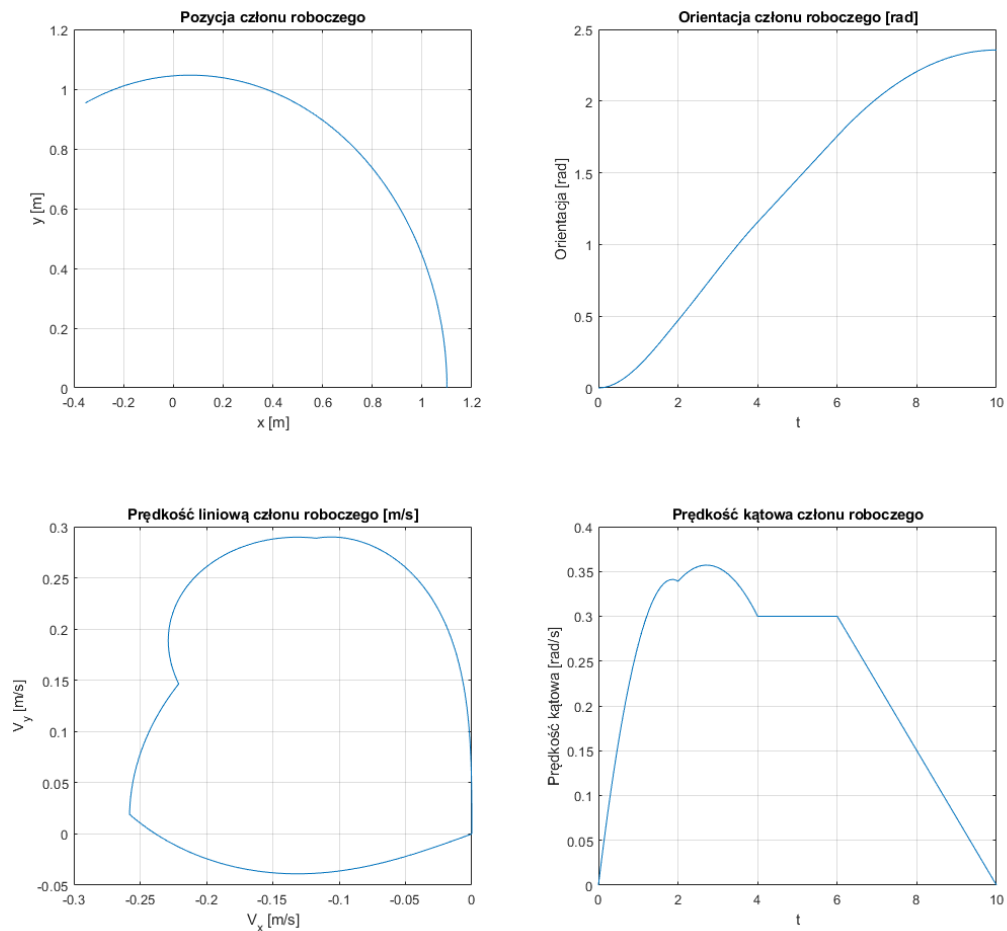
Uzyskane przebiegi pozycji, prędkości oraz przyspieszenia kąowego przegubów manipulatora z funkcji Trajectory\_Generation:



Rysunek 2 Wygenerowane przebiegi pozycji, prędkości i przyspieszenia kąowego przegubów

**6. Przebieg pozycji członu roboczego, przebieg orientacji członu roboczego, prędkość liniowa członu roboczego oraz prędkość kątowna członu roboczego dla wygenerowanej w pkt. 5 trajektorii przegubów.**

Na podstawie pozycji oraz prędkości kątowych przegubów zostały wygenerowane przebiegi pozycji, orientacji, prędkości liniowej oraz prędkości kątownej członu roboczego uzyskane z funkcji Direct\_2DoF.



*Rysunek 3 Wygenerowane przebiegi pozycji, orientacji, prędkości liniowej oraz prędkości kątownej członu roboczego*

**7. Skrypt źródłowy funkcji "inverse\_2DoF" napisany w programie Matlab.**

Kolejnym zadaniem było stworzenie funkcji "inverse\_2DoF" realizującą odwrotne zadanie kinematyki dla naszego modelu manipulatora. Funkcja jako parametry przyjmowała pozycję członu roboczego wygenerowaną przy pomocy funkcji direct\_2DoF oraz długości członów manipulatora.

## Skrypt źródłowy funkcji inverse\_2DoF:

```
function [sol_1, sol_2] = inverse_2DoF(Pee, L1, L2)

%   Pee - dwuelementowy wektor zawierający składową X i składową Y pozycji członu roboczego [m].
%   L1, L2 - długości członów manipulatora [m].
%   sol_1, sol_2 - dwuelementowe wektory zawierające pozycje kątowe przegubów [rad].

sol_1_theta_2 = real(acos((Pee(:, 1).^2 + Pee(:, 2).^2 - L1^2 - L2^2) / (2*L1*L2)));
sol_1_theta_1 = atan2(Pee(:, 2), Pee(:, 1)) - atan2(L2.*sin(sol_1_theta_2), L1 + (L2 .* cos(sol_1_theta_2)));
sol_1 = [sol_1_theta_1, sol_1_theta_2];

sol_2_theta_2 = -real(acos((Pee(:, 1).^2 + Pee(:, 2).^2 - L1^2 - L2^2) / (2*L1*L2)));
sol_2_theta_1 = atan2(Pee(:, 2), Pee(:, 1)) - atan2(L2.*sin(sol_2_theta_2), L1 + (L2 .* cos(sol_2_theta_2)));
sol_2 = [sol_2_theta_1, sol_2_theta_2];

end
```

*Listing 4 Skrypt źródłowy funkcji inverse\_2DoF*

## 8. Parametry wejściowe funkcji „Trajectory\_Generation”.

Funkcja inverse\_2DoF została wywołana w następujący sposób:

- 1) Na początku zostały wygenerowane trajektorie pozycji kątowej oraz prędkości kątowej dla dwóch przegubów manipulatora. W tym celu została wywołana funkcja Trajectory\_Generation z poniższymi parametrami wejściowymi.

Dane wejściowe dla przegubu nr 1

```
q_p_1 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_1 = 0.0;        % prędkość początkowa przegubu nr 1 [rad]
q_k_1 = 1/2 * pi;    % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_1 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_1 = 10;           % Czas trwania ruchu przegubu [s]
Ta_1 = 4;            % Czas trwania fazy przyspieszania [s]
Tb_1 = 4;            % Czas trwania fazy hamowania [s]
dt_1 = 0.01;         % Krok czasowy [s]
V_1 = 0.2;           % Prędkość przegubu w fazie drugiej [rad/s]
[q_1,dq_1,ddq_1, t_1] = Trajectory_Generation(solved, q_p_1, dq_p_1,...
q_k_1, dq_k_1, Tk_1, Ta_1, Tb_1, dt_1, V_1);
```

Dane wejściowe dla przegubu nr 2

```
q_p_2 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_2 = 0.0;        % prędkość początkowa przegubu nr 1 [rad]
q_k_2 = 1/4 * pi;    % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_2 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_2 = 10;           % Czas trwania ruchu przegubu [s]
Ta_2 = 2;            % Czas trwania fazy przyspieszania [s]
Tb_2 = 4;            % Czas trwania fazy hamowania [s]
dt_2 = 0.01;         % Krok czasowy [s]
V_2 = 0.1;           % Prędkość przegubu w fazie drugiej [rad/s]
```

```
[q_2,dq_2,ddq_2, t_2] = Trajectory_Generation(solved, q_p_2, dq_p_2, ...
    q_k_2, dq_k_2, Tk_2, Ta_2, Tb_2, dt_2, V_2);
```

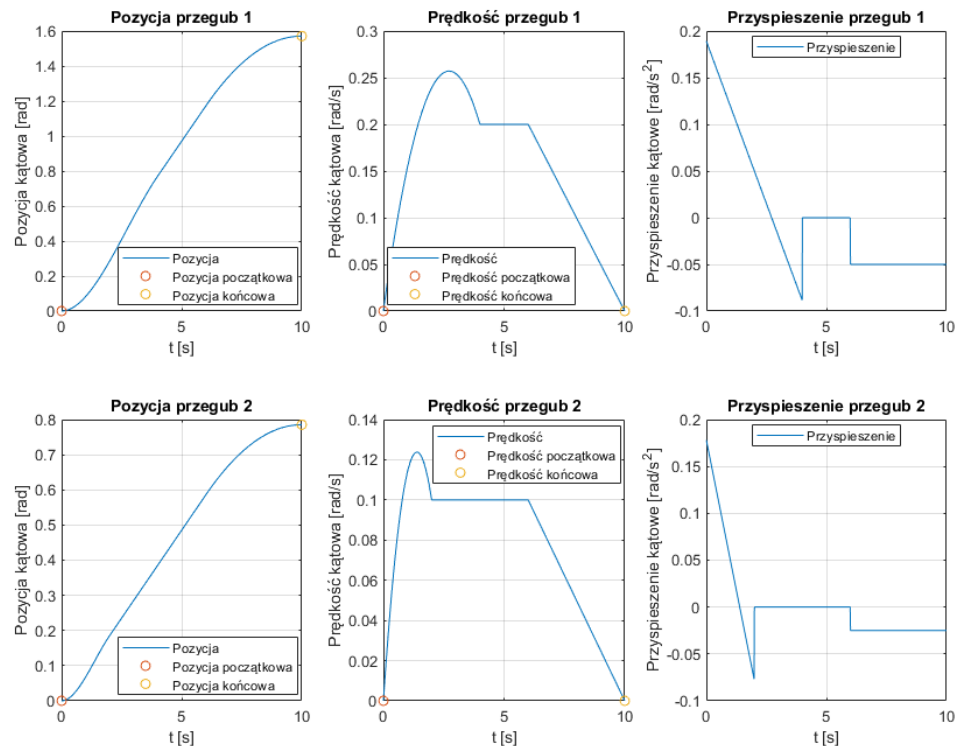
- 2) Następnie w celu uzyskania przebiegu pozycji członu roboczego manipulatora wywołana została funkcja `direct_2DoF` w poniższy sposób:

```
q = double([q_1; q_2]');
dq = double([dq_1; dq_2]');
L1 = 0.6;
L2 = 0.5;
[Pee, Psiee, Vee, Omee] = direct_2DoF(q, dq, L1, L2);
```

- 3) Ostatnim krokiem było wywołanie funkcji `inverse_2DoF` dla przebiegu pozycji członu roboczego wygenerowanego przy pomocy funkcji `direct_2DoF` w poprzednim kroku.

```
[sol1, sol2] = inverse_2DoF(Pee, L1, L2);
```

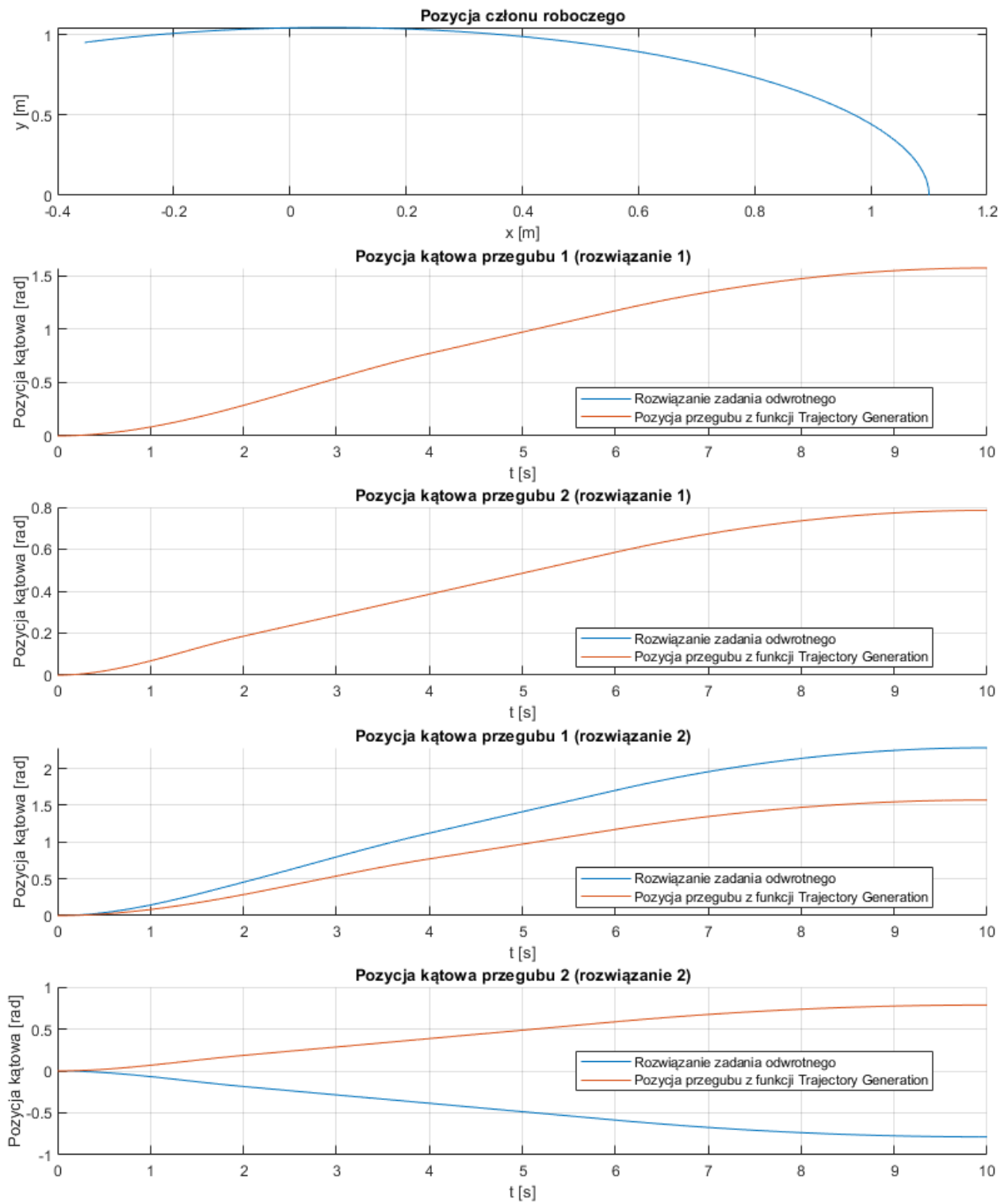
## 9. Wykres wygenerowanej w pkt 8 trajektorii oraz wykresy pozycji kątowych przegubów wyznaczone poprzez rozwiązanie zadania odwrotnego.



Rysunek 4 Wygenerowane przebiegi pozycji, prędkości i przyspieszenia kątowego przegubów manipulatora



Rozwiązanie zadania kinematyki odwrotnej:



Rysunek 5 Uzyskane pozycje kątowe przegubów z rozwiązania zadania kinematyki odwrotnej

Dodatkowo na powyższym wykresie w celach porównawczych pomarańczową linią zostały narysowane przebiegi pozycji kątowych przegubów wygenerowane z funkcji Trajectory\_Generation.

**10. Skrypt źródłowy programu głównego (gdzie są podane parametry i wywoływane są funkcje „Trajectory\_Generation”, „direct\_2DoF” oraz „inverse\_2DoF”, i w którym znajdują się procedury wygenerowania wykresów).**

```
solved = Z;
q_p_1 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_1 = 0.0;        % prędkość początkowa przegubu nr 1 [rad/s]
q_k_1 = 1/2 * pi;     % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_1 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_1 = 10;           % Czas trwania ruchu przegubu [s]
Ta_1 = 4;            % Czas trwania fazy przyspieszania [s]
Tb_1 = 4;            % Czas trwania fazy hamowania [s]
dt_1 = 0.01;         % Krok czasowy [s]
V_1 = 0.2;           % Prędkość przegubu w fazie drugiej [rad/s]
[q_1,dq_1,ddq_1, t_1] = Trajectory_Generation(solved, q_p_1, dq_p_1,...
q_k_1, dq_k_1, Tk_1, Ta_1, Tb_1, dt_1, V_1);

solved = Z;
q_p_2 = 0;           % pozycja początkowa przegubu nr 1 [rad]
dq_p_2 = 0.0;        % prędkość początkowa przegubu nr 1 [rad/s]
q_k_2 = 1/4 * pi;     % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_2 = 0.0;        % prędkość końcowa przegubu nr 1 [rad/s]
Tk_2 = 10;           % Czas trwania ruchu przegubu [s]
Ta_2 = 2;            % Czas trwania fazy przyspieszania [s]
Tb_2 = 4;            % Czas trwania fazy hamowania [s]
dt_2 = 0.01;         % Krok czasowy [s]
V_2 = 0.1;           % Prędkość przegubu w fazie drugiej [rad/s]
[q_2,dq_2,ddq_2, t_2] = Trajectory_Generation(solved, q_p_2, dq_p_2, ...
q_k_2, dq_k_2, Tk_2, Ta_2, Tb_2, dt_2, V_2);

figure;
subplot(2, 3, 1); plot(t_1, q_1, t_1(1), q_p_1, 'o', Tk_1, q_k_1, 'o');
legend('Pozycja', 'Pozycja początkowa', 'Pozycja końcowa',"Location","best")
title('Pozycja przegub 1'); xlabel('t [s]'); ylabel('Pozycja kątowa [rad]')
grid on; axis auto
subplot(2, 3, 2); plot(t_1, dq_1, t_1(1), dq_p_1, 'o', Tk_1, dq_k_1, 'o')
legend('Prędkość', 'Prędkość początkowa', 'Prędkość końcowa',
"Location","best"); title('Prędkość przegub 1'); xlabel('t [s]')
ylabel('Prędkość kątowa [rad/s]'); grid on; axis auto
subplot(2, 3, 3); plot(t_1, ddq_1); legend('Przyspieszenie',
"Location","best"); title('Przyspieszenie przegub 1'); xlabel('t [s]')
ylabel('Przyspieszenie kątowe [rad/s{^2}]'); grid on; axis auto
```

```

subplot(2, 3, 4); plot(t_2, q_2, t_2(1), q_p_2, 'o', Tk_2, q_k_2, 'o')
legend('Pozycja', 'Pozycja początkowa', 'Pozycja końcowa',
"Location","best"); title('Pozycja przegub 2'); xlabel('t [s]');
ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(2, 3, 5); plot(t_2, dq_2, t_2(1), dq_p_2, 'o', Tk_2, dq_k_2, 'o')
legend('Prędkość', 'Prędkość początkowa', 'Prędkość końcowa',
"Location","best"); title('Prędkość przegub 2'); xlabel('t [s]')
ylabel('Prędkość kątowa [rad/s]'); grid on; axis auto
subplot(2, 3, 6); plot(t_2, ddq_2); legend('Przyspieszenie',
"Location","best"); title('Przyspieszenie przegub 2'); xlabel('t [s]')
ylabel('Przyspieszenie kątowe [rad/s{^2}]'); grid on; axis auto

t = t_1;
q = double([q_1; q_2]');
dq = double([dq_1; dq_2]');
L1 = 0.6;
L2 = 0.5;
[Pee, Psiee, Vee, Omee] = direct_2DoF(q, dq, L1, L2);

```

```

figure
subplot(2, 2, 1); plot(Pee(:, 1), Pee(:, 2)); title('Pozycja członu
roboczego'); xlabel('x [m]'); ylabel('y [m]'); grid on; axis auto
subplot(2, 2, 2); plot(t, Psiee); title('Orientacja członu roboczego [rad]')
xlabel('t'); ylabel('Orientacja [rad]'); grid on; axis auto
subplot(2, 2, 3); plot(Vee(:, 1), Vee(:, 2)); title('Prędkość liniową członu
roboczego [m/s]'); xlabel('V{x} [m/s]'); ylabel('V{y} [m/s]'); grid on;
axis auto
subplot(2, 2, 4); plot(t, Omee); title('Prędkość kątowa członu roboczego');
xlabel('t'); ylabel('Prędkość kątowa [rad/s]'); grid on; axis auto

```

```

[sol1, sol2] = inverse_2DoF(Pee, L1, L2);
figure
subplot(5, 1, 1); plot(Pee(:, 1), Pee(:, 2)); title('Pozycja członu
roboczego'); xlabel('x [m]'); ylabel('y [m]'); grid on; axis auto
subplot(5, 1, 2); hold on; plot(t, sol1(:, 1)); plot(t, q_1); hold off
title('Pozycja kątowa przegubu 1 (rozwiązanie 1)')
legend('Rozwiązanie zadania odwrotnego', 'Pozycja przegubu z funkcji
Trajectory Generation', 'Location','best')
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(5, 1, 3); hold on; plot(t, sol1(:, 2)); plot(t, q_2); hold off
title('Pozycja kątowa przegubu 2 (rozwiązanie 1)')
legend('Rozwiązanie zadania odwrotnego', 'Pozycja przegubu z funkcji
Trajectory Generation', 'Location','best')
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(5, 1, 4); hold on; plot(t, sol2(:, 1)); plot(t, q_1); hold off
title('Pozycja kątowa przegubu 1 (rozwiązanie 2)')

```

```

legend('Rozwiązanie zadania odwrotnego', 'Pozycja przegubu z funkcji
Trajectory Generation', 'Location','best')
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(5, 1, 5); hold on; plot(t, sol2(:, 2)); plot(t, q_2); title('Pozycja
kątowa przegubu 2 (rozwiązanie 2)')
legend('Rozwiązanie zadania odwrotnego', 'Pozycja przegubu z funkcji
Trajectory Generation', 'Location','best')
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto

```

## 11. Wnioski.

Z wykresów wynika, że symulacja zadania kinematyki prostej oraz odwrotnej dla manipulatora o dwóch przegubach obrotowych zostało rozwiązane poprawnie. Zadanie proste kinematyki zamienia współrzędne złączowe robota na współrzędne kartezjańskie końcówki członu roboczego, natomiast zadanie odwrotne współrzędne kartezjańskie końcówki członu roboczego na współrzędne złączowe.

W funkcji Trajectory\_Generation końcowa pozycja kątowa przegubu 1 wynosi  $1/2 * \pi$ , zaś przegubu nr 2  $1/4 * \pi$  czyli odpowiednio 90 oraz 45 stopni. W rozwiązaniu nr 1 zadania kinematyki odwrotnej pozycja kątowa przegubów pokrywają się z wynikami uzyskanymi w funkcji „Trajectory\_Generation”. Rozwiązanie nr 2 również zostało wyznaczone poprawnie, ponieważ pozycja kątowa przegubu nr 2 jest odbiciem wykresu z funkcji Trajectory\_Generation od osi X. Kończówka robota osiąga dokładnie ten sam punkt we współrzędnych kartezjańskich lecz robi to na dwa różne sposoby.

Podsumowując, zadanie kinematyki odwrotnej jest dużo trudniejsze od prostego zadania kinematyki, lecz pozwala na sprawdzenie poprawności wygenerowanego rozwiązania kinematyki prostej co jest bardzo ważne w przypadku obliczeń numerycznych. Powyższe ćwiczenie pozwoliło mi na zapoznanie się z symulacją oraz rozwiązywaniem zadania kinematyki prostej i odwrotnej dla manipulatora o dwóch przegubach obrotowych.