Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

Sprawozdanie Robotyka Kosmiczna

Model dynamiki układu o dwóch stopniach swobody

Autor:

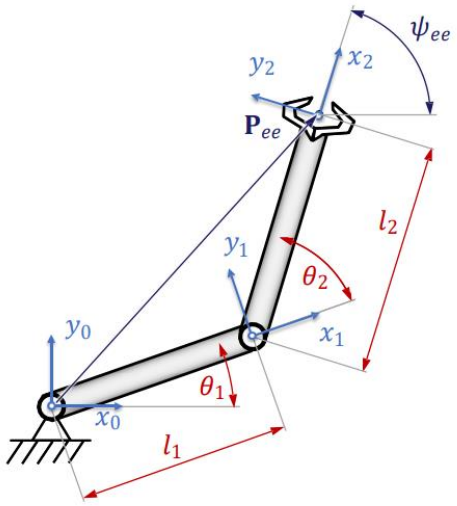
Dawid Lisek

Nr indeksu: 402382
Kierunek studiów: **Automatyka i Robotyka**
Specjalizacja: **Komputerowe Systemy Sterowania**
Grupa: pn 13:15 - 15:45

1. Cel ćwiczenia.

Celem ćwiczenia było zaimplementowanie modelu dynamiki prostej pozwalającego na wyznaczenie momentów napędowych przegubów manipulatora. Następnie należało przeprowadzić symulację dynamiki prostej przy pomocy procedury ode4. Należało również zweryfikować poprawność rozwiązań zawierających momenty napędowe.

2. Model dynamiki manipulatora płaskiego o dwóch stopniach swobody.



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.

Model dynamiki został zaimplementowany przy pomocy poniższych równań:

Równanie 1 Równania opisujące model dynamiki prostej manipulatora 2DoF

$$\begin{bmatrix} \left(\frac{1}{3}m_1 + m_2\right)l_1^2 + \frac{1}{3}m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 & \frac{1}{3}m_2 l_2^2 + \frac{1}{2}m_2 l_1 l_2 \cos\theta_2 \\ \frac{1}{3}m_2 l_2^2 + \frac{1}{2}m_2 l_1 l_2 \cos\theta_2 & \frac{1}{3}m_2 l_2^2 \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix}$$

$$+ \begin{bmatrix} 0 & -m_2 l_1 l_2 \left(\dot{\theta}_1 + \frac{1}{2}\dot{\theta}_2\right)\sin\theta_2 \\ \frac{1}{2}m_2 l_1 l_2 \dot{\theta}_1 \sin\theta_2 & 0 \end{bmatrix} \begin{bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

3. **Skrypt źródłowy do wyznaczenia momentów obciążających pary kinematyczne dla dowolnej trajektorii wyznaczonej za pomocą procedury „Trajectory_Generation". Skrypt ma zostać napisany w programie Matlab.**

   Aby wyznaczyć momenty napędowe stworzona została funkcja dynamics_model, która jako argumenty przyjmuje pozycje kątowe, prędkości kątowe, przyspieszenia kątowe oraz masy i długości członów manipulatora. Funkcja zwraca wyznaczone momenty napędowe dla dwóch przegubów manipulatora.

```
function [u] = dynamics_model(q1, q2, dq1, dq2, ddq1, ddq2, m1, m2, L1, L2)

    n = length(q1);
    for i=1:n
        A = [(((1/3) * m1) + m2) * L1^2 + ((1/3) * m2 * L2^2) + (m2 * L1 * L2 * cos(q2(i))), ...
       ((1/3) * m2 * L2^2) + ((1/2) * m2 * L1 * L2 * cos(q2(i))); ...
       ((1/3) * m2 * L2^2) + ((1/2) * m2 * L1 * L2 * cos(q2(i))), ...
       ((1/3) * m2 * L2^2)];

        B = [ddq1(i); ddq2(i)];

        C = [0, (-m2 * L1 * L2 * (dq1(i) + (1/2) * dq2(i)) * sin(q2(i))); ...
       ((1/2) * m2 * L1 * L2 * dq1(i) * sin(q2(i))), 0];

        D = [dq1(i); dq2(i)];

        u(:, i) = A*B + C*D;

    end
end
```

Listing 1 Kod źródłowy funkcji dynamics_model

4. **Wybrane parametry wejściowe funkcji „Trajectory_Generation" dla dwóch przegubów manipulatora oraz wykres wygenerowanej trajektorii (przebiegi pozycji kątowych przegubów).**

   Do użycia funkcji dynamics_model() potrzebne były przebiegi pozycji, prędkości oraz przyspieszenia kątowego dwóch przegubów manipulatora. W tym celu została wywołana funkcja Trajectory_Generation dla dwóch różnych zestawów danych.

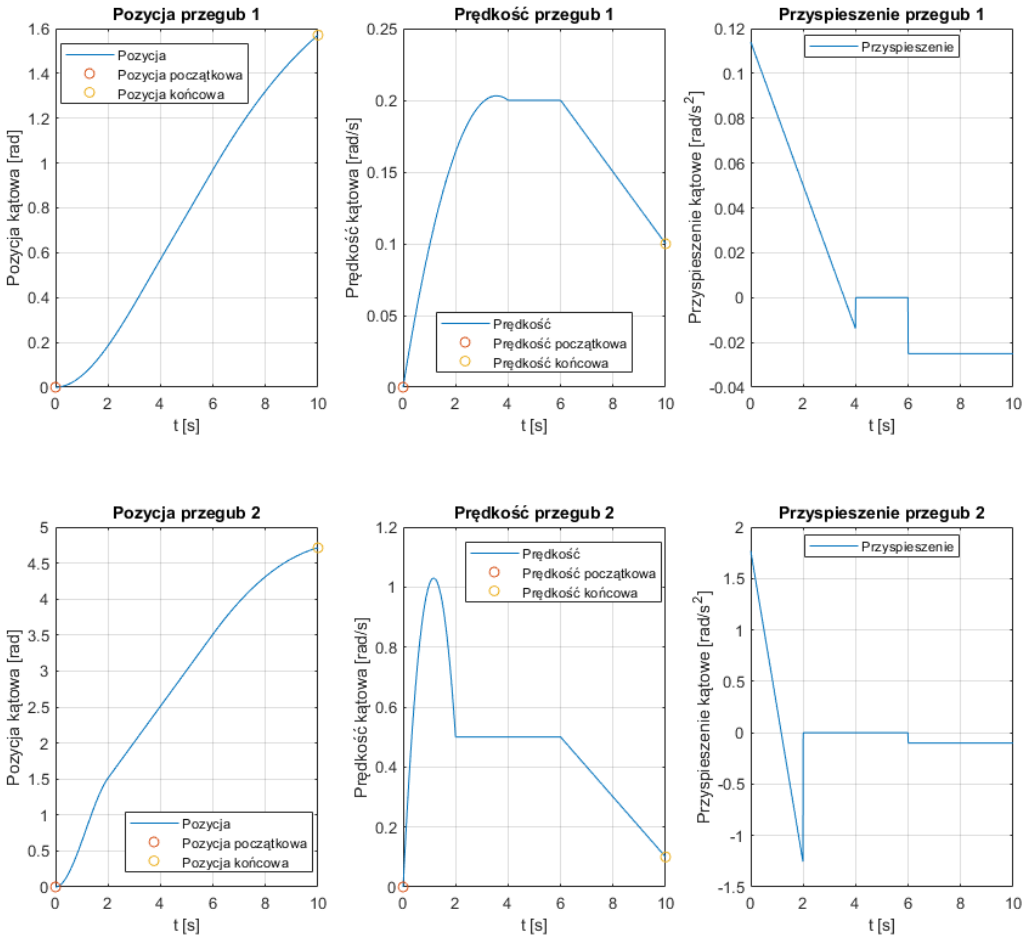
- **Przegub 1**

```
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 = 0.1;       % 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);
```

- **Przegub 2**

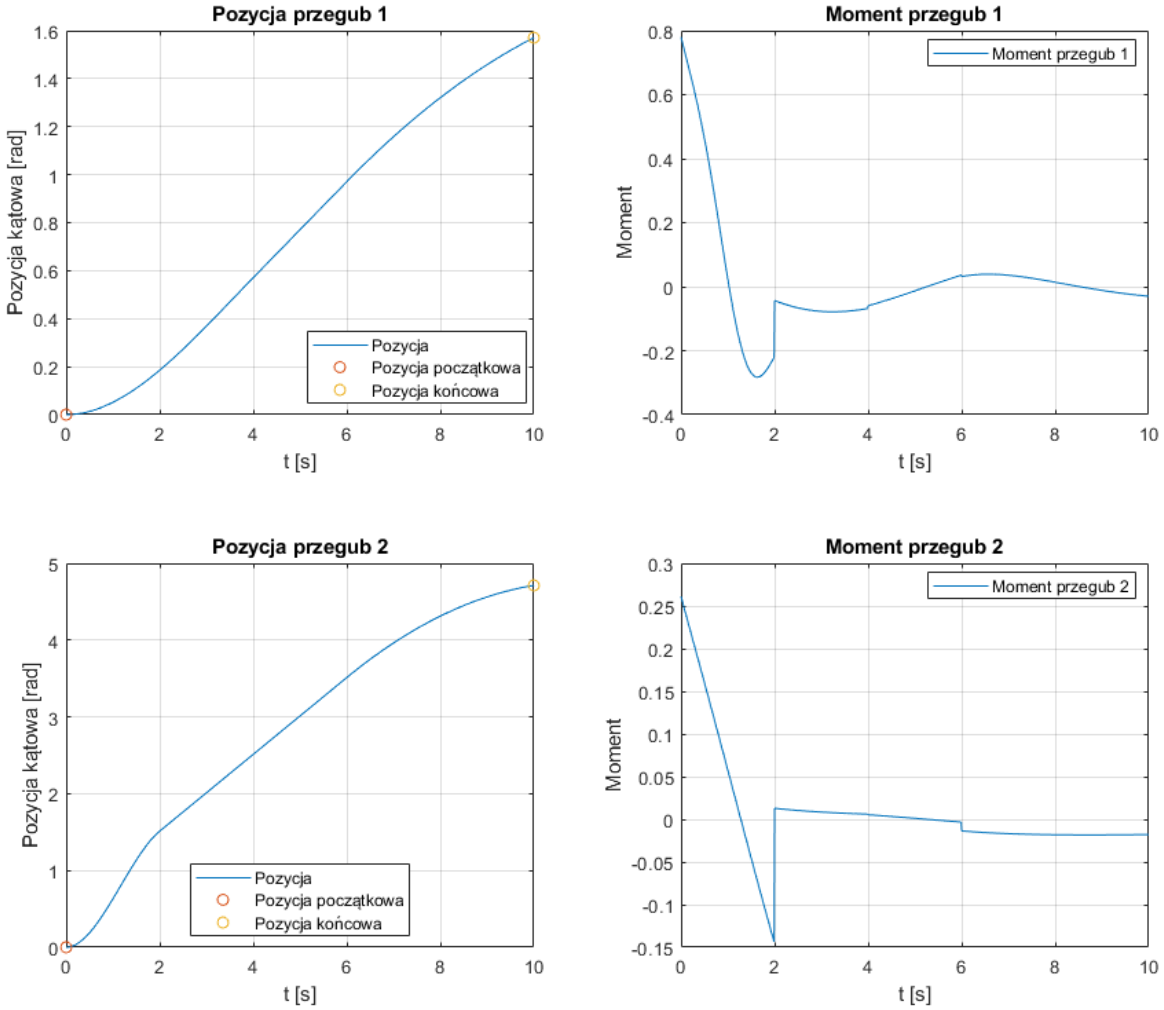
```
q_p_2 = 0;          % pozycja początkowa przegubu nr 2 [rad]
dq_p_2 = 0.0;       % prędkość początkowa przegubu nr 2 [rad]
q_k_2 = 3/2 * pi;   % pozycja końcowa przegubu nr 2 [rad/s]
dq_k_2 = 0.1;       % prędkość końcowa przegubu nr 2 [rad/s]
Tk_2 = 10;          % Czas trwania ruchu przegubu [s]
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 = 0.5;          % 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);
```



Rysunek 2 Wykres wygenerowanych pozycji, prędkości oraz przyspieszenia kątowego dwóch przegubów manipulatora

5. Wykres wyznaczonych pozycji kątowych przegubów oraz wygenerowanych momentów.

Następnie dla wygenerowanych trajektorii przegubów należało wywołać funkcję dynamics_model oraz wygenerować przebiegi momentów napędowych dla poszczególnych przegubów manipulatora.



Rysunek 3 Wygenerowane momenty napędowe dla przegubów oraz ich pozycje kątowe.

6. Skrypt źródłowy procedury ode4. Skrypt ma zostać napisany w programie Matlab.

Celem sprawdzenia poprawności wygenerowanych momentów napędowych została utworzona funkcja numerycznego rozwiązywania równań

różniczkowych zwyczajnych ode4. Równanie nr 1 może zostać przekształcone w poniższy sposób:

Równanie 2 Przekształcenie równania nr 1

$$M(q)\ddot{q} + C(q,\dot{q}) = u$$

$$\ddot{q} = M(q)^{-1}(u - C(q,\dot{q})\dot{q})$$

Mając wyznaczone w ten sposób przyspieszenie kątowe może skorzystać z metody Rungego-Kutty 4 rzędu, która zwróci nam wyznaczoną iteracyjnie prędkość kątową. Jeśli ponownie skorzystamy z tej metody dla wyznaczonej prędkości kątowej to otrzymamy wyznaczoną iteracyjnie pozycję kątową przegubu.

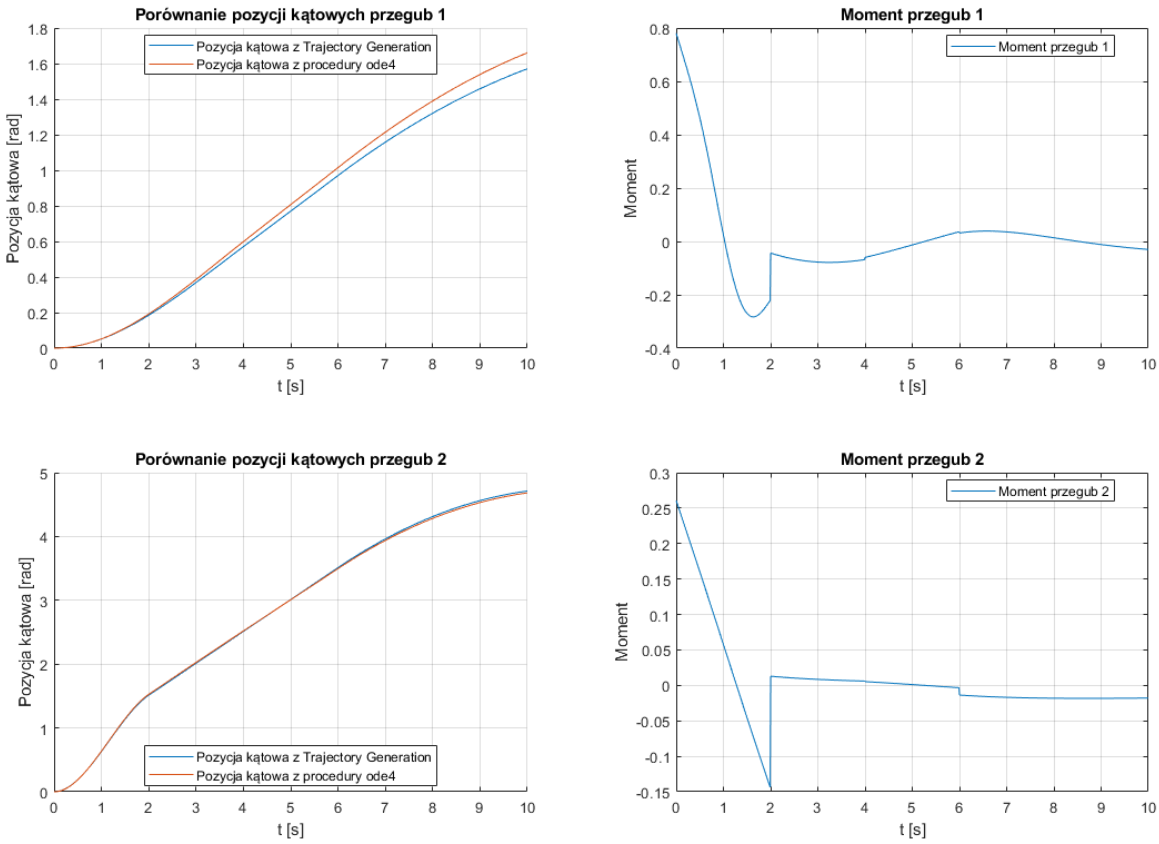
Powyższa procedura została zaimplementowana wewnątrz funkcji ode4 z poniższym kodem źródłowym.

```
function [q, dq] = ode4(u, q_0, dq_0, L1, L2, m1, m2)

    h = 0.01;
    q(:, 1) = q_0;
    dq(:, 1) = dq_0;


    for i = 1:1:1000

        M = [(((1/3) * m1) + m2) * L1^2 + ((1/3) * m2 * L2^2) + (m2 * L1 * L2 * cos(q(2, i))), ...
        ((1/3) * m2 * L2^2) + ((1/2) * m2 * L1 * L2 * cos(q(2, i))); ...
        ((1/3) * m2 * L2^2) + ((1/2) * m2 * L1 * L2 * cos(q(2, i))), ...
        ((1/3) * m2 * L2^2)];

        C = [0, (-m2 * L1 * L2 * (dq(1, i) + (1/2) * dq(2, i)) * sin(q(2, i))); ...
        ((1/2) * m2 * L1 * L2 * dq(1, i) * sin(q(2, i))), 0];

        K_11 = h .* dq(:, i);
        K_12 = h .* inv(M) * (u(:, i) - C*dq(:, i));
        K_21 = h .* (dq(:, i) + K_11/2);
        K_22 = h .* (inv(M) * (u(:, i) - C*dq(:, i)) + K_12/2);
        K_31 = h .* (dq(:, i) + K_21/2);
        K_32 = h .* (inv(M) * (u(:, i) - C*dq(:, i)) + K_22/2);
        K_41 = h .* (dq(:, i)+ K_31);
        K_42 = h .* (inv(M) * (u(:, i) - C*dq(:, i)) + K_32);

        K_1 = (1/6) * (K_11 + 2*K_21 + 2*K_31 + K_41);
        K_2 = (1/6) * (K_12 + 2*K_22 + 2*K_32 + K_42);

        q(:, i + 1) = q(:,i) + K_1;
        dq(:, i + 1) = dq(:,i) + K_2;
    end
end
```

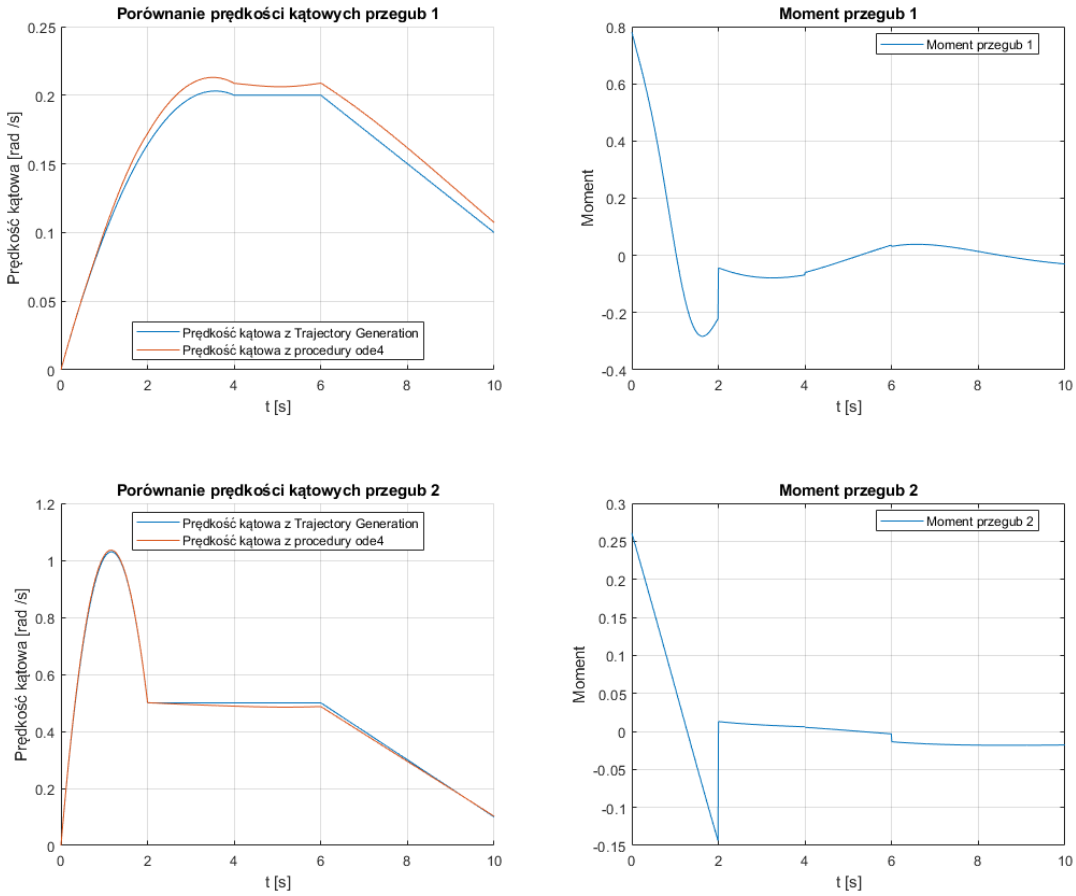
Listing 2 Kod źródłowy funkcji ode4

7. Porównanie przebiegów pozycji kątowych oraz prędkości kątowych.

Następnie porównane zostały przebiegi pozycji oraz prędkości kątowych wygenerowanych z funkcji ode4, z przebiegami wygenerowanymi w funkcji Trajectory_Generation.



Rysunek 4 Porównanie pozycji kątowych



Rysunek 5 Porównanie prędkości kątowych

8. Skrypt źródłowy programu głównego.

```
solved = Z;
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 = 0.1;       % 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);
q_1 = double(q_1);
dq_1 = double(dq_1);
ddq_1 = double(ddq_1);
```

```
q_p_2 = 0;          % pozycja początkowa przegubu nr 1 [rad]
dq_p_2 = 0.0;       % prędkość początkowa przegubu nr 1 [rad]
q_k_2 = 3/2 * pi;   % pozycja końcowa przegubu nr 1 [rad/s]
dq_k_2 = 0.1;       % 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 = 0.5;          % 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);
q_2 = double(q_2);
dq_2 = double(dq_2);
ddq_2 = double(ddq_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'); label('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

m1 = 2.5;
m2 = 1.5;
L1 = 0.6;
L2 = 0.5;
[u] = dynamics_model(q_1, q_2, dq_1, dq_2, ddq_1, ddq_2, m1, m2, L1, L2);
u_1 = u(1, :);
u_2 = u(2, :);
```

```
figure;
subplot(2, 2, 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, 2, 2); plot(t_1, u_1); legend('Moment przegub 1',
"Location","best"); title('Moment przegub 1'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto
subplot(2, 2, 3);
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, 2, 4); plot(t_2, u_2); legend('Moment przegub 2',
"Location","best"); title('Moment przegub 2'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto

q0 = [0; 0];
dq0 = [0; 0];
u = double(u);

[q_check, dq_check] = ode4(u, q0, dq0, L1, L2, m1, m2);
q_1_check = q_check(1, :);
q_2_check = q_check(2, :);
dq_1_check = dq_check(1, :);
dq_2_check = dq_check(2, :);


figure;
subplot(2, 2, 1); hold on; plot(t_1, q_1); plot(t_1, q_1_check); hold off
legend('Pozycja kątowa z Trajectory Generation', 'Pozycja kątowa z procedury
ode4', "Location","best"); title('Porównanie pozycji kątowych przegub 1');
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(2, 2, 2); plot(t_1, u_1); legend('Moment przegub 1',
"Location","best"); title('Moment przegub 1'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto;
subplot(2, 2, 3); hold on; plot(t_2, q_2); plot(t_2, q_2_check); hold off
legend('Pozycja kątowa z Trajectory Generation', 'Pozycja kątowa z procedury
ode4', "Location","best"); title('Porównanie pozycji kątowych przegub 2')
xlabel('t [s]'); ylabel('Pozycja kątowa [rad]'); grid on; axis auto
subplot(2, 2, 4); plot(t_2, u_2); legend('Moment przegub 2',
"Location","best"); title('Moment przegub 2'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto

figure; subplot(2, 2, 1); hold on; plot(t_1, dq_1); plot(t_1, dq_1_check);
hold off; legend('Prędkość kątowa z Trajectory Generation', 'Prędkość kątowa
z procedury ode4', "Location","best"); title('Porównanie prędkości kątowych
```

```
przegub 1'); xlabel('t [s]'); ylabel('Prędkość kątowa [rad /s]'); grid on;
axis auto;
subplot(2, 2, 2); plot(t_1, u_1); legend('Moment przegub 1',
"Location","best"); title('Moment przegub 1'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto;
subplot(2, 2, 3); hold on; plot(t_2, dq_2); plot(t_2, dq_2_check); hold off
legend('Prędkość kątowa z Trajectory Generation', 'Prędkość kątowa z
procedury ode4', "Location","best"); title('Porównanie prędkości kątowych
przegub 2'); xlabel('t [s]'); ylabel('Prędkość kątowa [rad /s]'); grid on;
axis auto
subplot(2, 2, 4); plot(t_2, u_2); legend('Moment przegub 2',
"Location","best"); title('Moment przegub 2'); xlabel('t [s]');
ylabel('Moment'); grid on; axis auto
```

9. Wnioski.

Z wykresów wynika, że symulacja zadania dynamiki prostej działa poprawnie. Największe różnice są zauważalne w przebiegach pozycji kątowej oraz prędkości kątowej przegubu nr 1 jednak różnice są niewielkie. Mogą one wynikać z błędów numerycznych jakie niosą za sobą takie obliczenia lub zbyt dużego kroku *h*, który w naszym wypadku miał wartość 0.01. Dla przegubu nr 2 zauważalne są niewielkie różnice w przebiegu między wartością z funkcji Trajectory_Generation, a wartością uzyskaną z procedury ode4.

Zadanie proste dynamiki pozwala nam na wyznaczenie sił oraz momentów napędowych całego układu. Jest to istotne zadanie z punktu widzenia sterowania naszym manipulatorem, ponieważ pozwala nam na dobór odpowiednich komponentów podczas projektowania takiego manipulatora.