PROJEKT SPECJALNOŚCIOWY
Algorytmy sterowania manipulatorem typu EDDA

Autor: Emilia Szymańska 248975
czw 16:10

27.10.2021r.

1 Wstęp

EDDA to manipulator posiadający dwa przeguby obrotowe (2R). Obiekt ten może zostać opisany następującym równaniem dynamiki:

$$M(q) \cdot \ddot{q} + C(q, \dot{q}) \cdot \dot{q} + D(q) = u, \tag{1}$$

gdzie:

- q - wektor uogólnionych współrzędnych,
- $\dot{q}$ - wektor uogólnionych prędkości,
- $\ddot{q}$ - wektor uogólnionych przyspieszeń,
- $M(q)$ - macierz bezwładności (symetryczna i dodatnio określona),
- $C(q, \dot{q})$ - macierz sił Coriolisa i odśrodkowych bezwładności,
- $D(q)$ - wektor sił grawitacji,
- u - wektor sterowań.

Macierze są w następującej postaci:

$$M(q) = \begin{bmatrix} \theta_1 + a \cdot \theta_4 \cos q_2 & \theta_3 + b \cdot \theta_4 \cos q_2 \\ \theta_1 + b \cdot \theta_4 \cos q_2 & \theta_3 \end{bmatrix}, \tag{2}$$

$$C(q, \dot{q}) = \begin{bmatrix} \dot{q_2} & -(\dot{q_1} + \dot{q_2}) \\ -\dot{q_1} & d \end{bmatrix} \cdot c \cdot \theta_4 \sin q_2, \tag{3}$$

$$D(q) = \begin{bmatrix} \theta_2 \cos q_1 + \theta_4 \cos q_1 + q_2 \\ \theta_4 \cos q_1 + q_2 \end{bmatrix} \cdot g, \tag{4}$$

gdzie:

- $\theta_1 = 3,1kg \cdot m^2$,
- $\theta_2 = 9,5kg \cdot m$,
- $\theta_3 = 0,24kg \cdot m^2$,
- $\theta_4 = 0,77kg \cdot m$,
- $g = 9,81\frac{m}{s^2}$,
- $a = 0,6$,
- $b = 0,3$,
- $c = 0,3$,
- $d = 0$.

Dokonano następujących przekształceń, by ułatwić implementację obiektu w Simulinku:

$$M\ddot{q} + C\dot{q} + D = u \tag{5}$$

$$M\ddot{q} = u - C\dot{q} - D \tag{6}$$

$$\ddot{q} = M^{-1}(u - C\dot{q} - D) \tag{7}$$

Przyjmując oznaczenia:

$$M = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix}, \tag{8}$$

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}, \tag{9}$$

$$D = \begin{bmatrix} D_1 \\ D_2 \end{bmatrix}, \tag{10}$$

wówczas:

$$M^{-1} = \frac{1}{M_{11} \cdot M_{22} - M_{12} \cdot M_{21}} \begin{bmatrix} M_{22} & -M_{12} \\ -M_{21} & M_{11} \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \tag{11}$$

W takim przypadku możemy zapisać:

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \left(\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} - \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} - \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \right). \tag{12}$$

Podstawiając:

$$h_1 = u_1 - C_{11}\dot{q_1} - C_{12}\dot{q_2} - D_1, \tag{13}$$

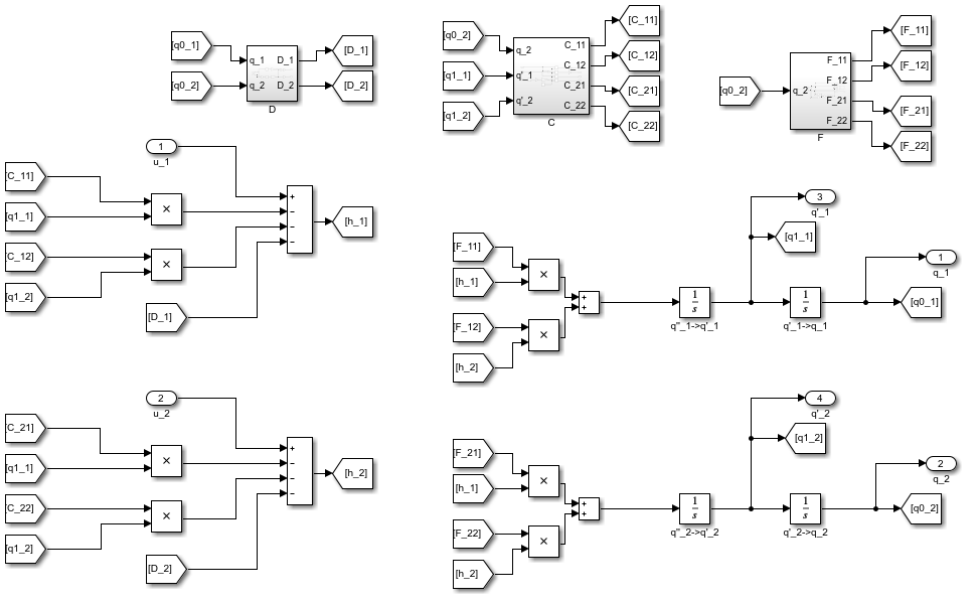
$$h_2 = u_2 - C_{21}\dot{q_1} - C_{22}\dot{q_2} - D_2, \tag{14}$$

otrzymujemy:

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \end{bmatrix}, \tag{15}$$

$$\begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \end{bmatrix} = \begin{bmatrix} F_{11}h_1 + F_{12}h_2 \\ F_{21}h_1 + F_{22}h_2 \end{bmatrix}. \tag{16}$$

Realizacja manipulatora EDDA w Simulinku jest widoczna na schemacie 1. Warunki początkowe wpisane w bloczki całkujące zależą od tego, jakie mamy położenie początkowe manipulatora - dla uproszczenia wpisałam wszędzie wartości 0.



Rysunek 1: Schemat w Simulinku realizujący implementację manipulatora EDDA.

2 Algorytmy sterowania

Jednym z zadań, jakie mogą być realizowane przez manipulatory, jest jest śledzenie trajektorii - określanie położeń i prędkości manipulatora w czasie. Do wykonywania takiego zadania implementowaliśmy dwa algorytmy - algorytm dokładnej linearyzacji oraz algorytm Qu i Dorsey'a.

2.1 Algorytm Qu i Dorsey'a (liniowy regulator PD)

Algorytm ten oblicza sterowanie jako różnicę pomiędzy sprzężeniem wyprzedzającym a korekcją narzucaną przez regulator PD o stałym wzmocnieniu. Możemy zapisać:

$$u = -K_p e - K_d \dot{e}, \tag{17}$$

gdzie:

- K_d, K_p - macierze wzmocnień (diagonalne),
- $e(t) = q(t) - q_d(t)$ - wektor błędów położeń,
- $\dot{e}(t) = \dot{q}(t) - \dot{q_d}(t)$ - wektor błędów prędkości.

Indeks d wskazuje na powiązanie z wartościami zadanymi.

Dokonując dalszych przekształceń:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = - \begin{bmatrix} K_p & 0 \\ 0 & K_p \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} - \begin{bmatrix} K_d & 0 \\ 0 & K_d \end{bmatrix} \begin{bmatrix} \dot{e_1} \\ \dot{e_2} \end{bmatrix}, \tag{18}$$

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = - \begin{bmatrix} K_p & 0 \\ 0 & K_p \end{bmatrix} \begin{bmatrix} q_1 - q_{1d} \\ q_2 - q_{2d} \end{bmatrix} - \begin{bmatrix} K_d & 0 \\ 0 & K_d \end{bmatrix} \begin{bmatrix} \dot{q_1} - \dot{q_{1d}} \\ \dot{q_2} - \dot{q_{2d}} \end{bmatrix}. \tag{19}$$

Z właściwości algorytmu można wymienić, że:

- dla błędu ustalonego dążącego do zera wartości wzmocnień regulatora PD dążą do nieskończoności,
- błąd dąży do błędu ustalonego różnego od zera dla wszystkich skończonych nastaw.

Trajektoria zadana w tym zadaniu opisana jest następującymi wzorami:

$$q_{1d}(t) = \frac{1}{2} \sin \frac{t}{/} 2, \tag{20}$$

$$q_{2d}(t) = 2 \cos t. \tag{21}$$

Do implementacji trajektorii w Symulinku potrzebujemy jeszcze obliczenia pochodnych:

$$\dot{q_{1d}}(t) = \frac{1}{4} \cos \frac{t}{2}, \tag{22}$$

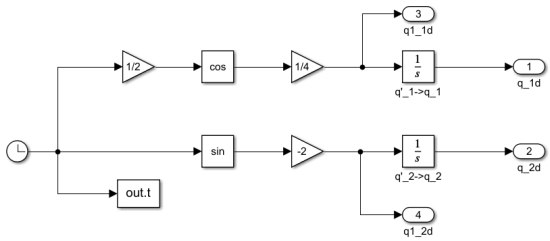
$$\dot{q_{2d}}(t) = -2 \sin t. \tag{23}$$

Wartości początkowe wynoszą:

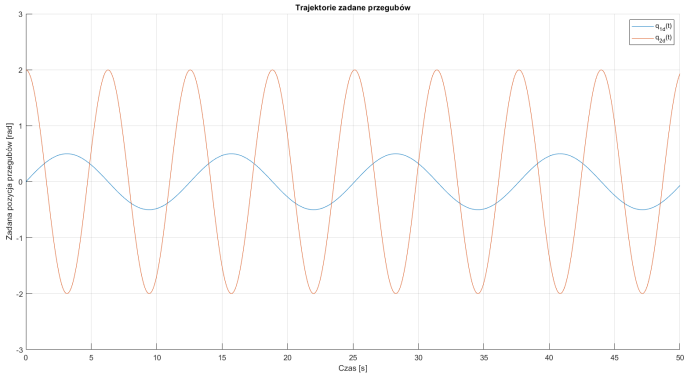
$$q_{1d}(0) = 0, \tag{24}$$

$$q_{2d}(0) = 2. \tag{25}$$

Schemat realizujący trajektorię zadaną wygląda jak na rys. 2, a ich przebieg zaprezentowano na rys. 3.



Rysunek 2: Schemat w Simulinku realizujący implementację trajektorii użytej przy algorytmie Qu i Dorsey'a.



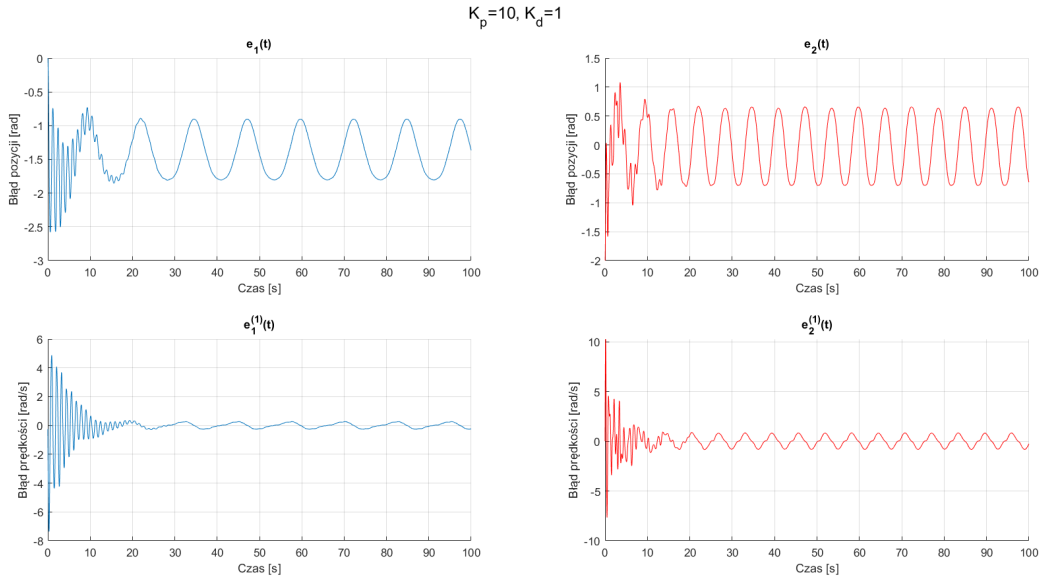
Rysunek 3: Zadane trajektorie w algorytmie Qu i Dorsey'a.

Zostały zbadane różne przypadki związane z wyborem wartości wzmocnień członów PD. Przyjęliśmy, że stosunek $\frac{K_p}{K_d}$ powinien wynosić 10. Wybrałam wartości K_p z wartości $\{10, 100, 1000, 10000, 100000\}$ oraz odpowiadające im wartości K_d. Rzędy otrzymanych błędów położeń i prędkości przegubów zamieszczono w tabeli 1. Na podstawie tych danych da się potwierdzić prawdziwość własności algorytmu dotyczącą zbiegania błędu ustalonego do zera przy coraz większych wzmocnieniach regulatora.

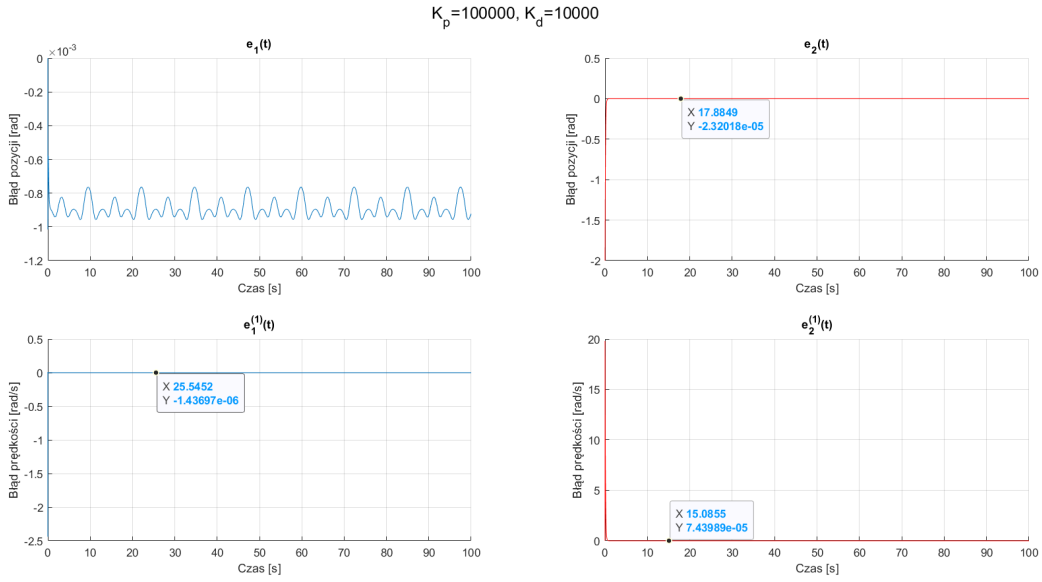
K_p	K_d	rząd błędu e_1	rząd błędu e_2	rząd błędu $\dot{e_1}$	rząd błędu $\dot{e_2}$
10	1	10^0	10^{-1}	10^0	10^0
100	10	10^0	10^{-2}	10^{-1}	10^{-1}
1 000	100	10^{-1}	10^{-3}	10^{-3}	10^{-2}
10 000	1 000	10^{-2}	10^{-4}	10^{-3}	10^{-4}
100 000	10 000	10^{-4}	10^{-5}	10^{-3}	10^{-5}

Tabela 1: Zależność między nastawami a błędami położeń.

Wyniki przedstawione na rys. 4 ukazują zbieganie błędu do błędu ustalonego, jednak dopiero na rys. 5 widać, że te wartości błędów ustalonych są bliższe zeru ze względu na większe wartości wzmocnień.



Rysunek 4: Wyniki dla $K_p = 10$ i $K_d = 1$.



Rysunek 5: a $K_p = 100000$ i $K_d = 10000$.

2.2 Algorytm dokładnej linearyzacji

Drugim algorytmem, który można zastosować w przypadku sterowania manipulatorem EDDA, jest algorytm dokładnej linearyzacji. Sposób postępowania przy takim sterowaniu można podzielić na dwa etapy:

- sprzężenie zwrotne,
- zmiana współrzędnych.

W pierwszym etapie zakładamy sterowanie o następującej postaci:

$$u = M(q) \cdot v + C(q, \dot{q}) \cdot \dot{q} + D, \tag{26}$$

gdzie dochodzi nam zmienna v - nowe wejście. Otrzymujemy równanie zamkniętej pętli sprzężenia zwrotnego:

$$M(q) \cdot \ddot{q} + C(q, \dot{q}) \cdot \dot{q} + D(q) = M(q) \cdot v + C(q, \dot{q}) \cdot \dot{q} + D. \tag{27}$$

Po przekształceniach mamy:

$$M(q) \cdot \ddot{q} = M(q) \cdot v, \tag{28}$$

$$\ddot{q} = v. \tag{29}$$

Równanie 29 ukazuje, układ ma cechy podwójnego integratora.

W drugim etapie skupiamy się na sterowaniu tym podwójnym integratorem. Dążymy do tego, by błędy dążyły do 0. Do równania $\ddot{q} = v$ dodajemy elementy związane z błędami położeń i błędami prędkości przegubów. Dokonując odpowiednich przekształceń i podstawień:

$$v = \ddot{q_d} - K_1(\dot{q} - \dot{q_d}) - K_0(q - q_d), \tag{30}$$

$$\ddot{q} = \ddot{q_d} - K_1(\dot{q} - \dot{q_d}) - K_0(q - q_d), \tag{31}$$

$$\ddot{q} - \ddot{q_d} + K_1(\dot{q} - \dot{q_d}) + K_0(q - q_d) = 0, \tag{32}$$

$$\ddot{e} + K_1 \cdot \dot{e} + K_0 \cdot e = 0, \tag{33}$$

gdzie:

- $\ddot{e}(t) = \ddot{q}(t) - \ddot{q_d}(t)$ - wektor błędów przyspieszeń,
- K_0, K_1 - macierze diagonalne parametrów implementowanego regulatora.

Po nałożeniu transformaty Laplace'a i skorzystania z kryterium stabilności Hurwitza można dojść do wniosku, że układ będzie stabilny dla $K_1, K_0 > 0$ (gdzie te brane są tu pod uwagę jako wartości skalarne, a nie macierze).

Ostatecznie otrzymane zostały następujące równania, które były pomocne przy implementacji w Simulinku:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} \dot{q_1} \\ \dot{q_2} \end{bmatrix} + \begin{bmatrix} D_1 \\ D_2 \end{bmatrix} \tag{34}$$

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \ddot{q}_{1d} \\ \ddot{q}_{2d} \end{bmatrix} - \begin{bmatrix} K_1 & 0 \\ 0 & K_1 \end{bmatrix} \begin{bmatrix} \dot{e_1} \\ \dot{e_2} \end{bmatrix} - \begin{bmatrix} K_0 & 0 \\ 0 & K_0 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \end{bmatrix}. \tag{35}$$

Pomocniczo zdefiniowałam następujące sygnały:

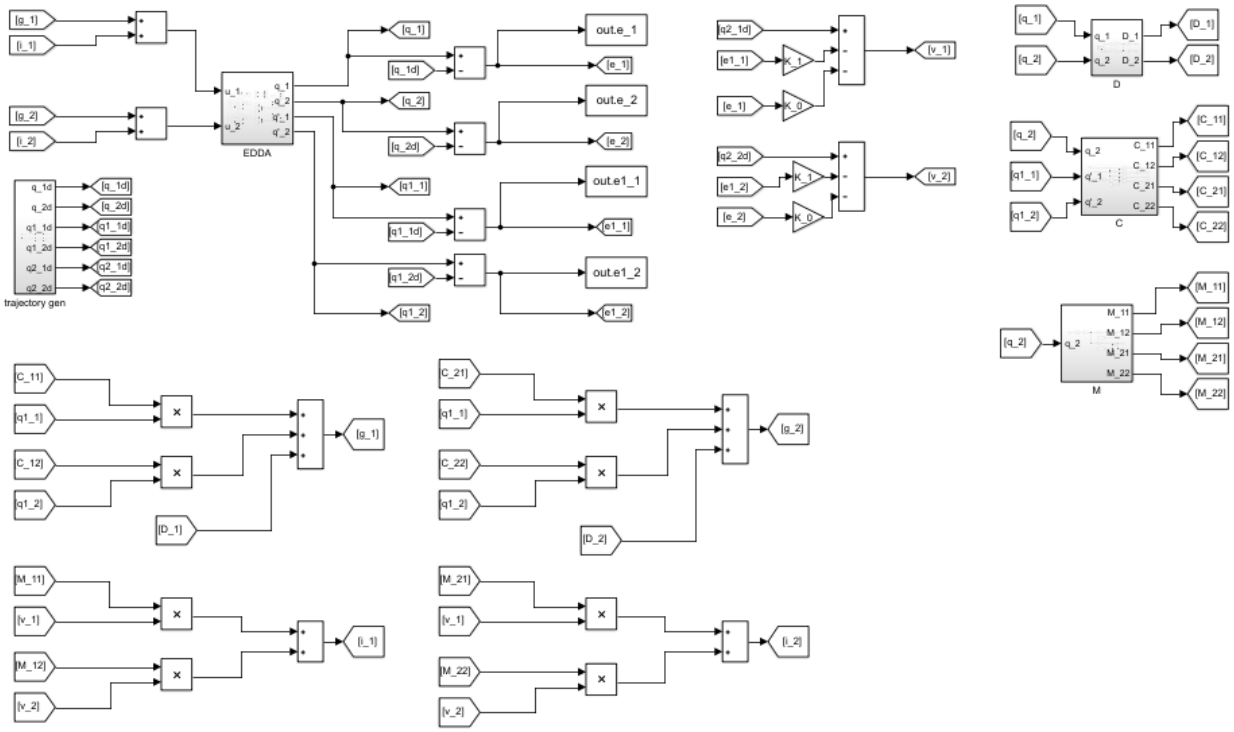
$$g_1 = C_{11}\dot{q}_1 + C_{12}\dot{q}_2 + D_1, \tag{36}$$

$$g_2 = C_{21}\dot{q}_1 + C_{22}\dot{q}_2 + D_2, \tag{37}$$

$$i_1 = M_{11}v_1 + M_{12}v_2, \tag{38}$$

$$i_1 = M_{21}v_1 + M_{22}v_2. \tag{39}$$

Zaimplementowany system wygląda jak na rys. 6.



Rysunek 6: Zaimplementowany algorytm dokładnej linearyzacji dla manipulatora EDDA.

Trajektorie zadane były wzorami:

$$q_{1d} = \frac{1}{2}\sin\frac{t}{2}, \tag{40}$$

$$q_{2d} = 2\cos t. \tag{41}$$

Zostały obliczone pochodne trajektorii oraz warunki początkowe do całkowania:

$$\dot{q}_{1d} = \frac{1}{4}\cos\frac{t}{2}, \tag{42}$$

$$\dot{q}_{2d} = -2\sin t, \tag{43}$$

$$\ddot{q}_{1d} = -\frac{1}{8}\sin\frac{t}{2}, \tag{44}$$

$$\ddot{q}_{2d} = 2\cos t, \tag{45}$$

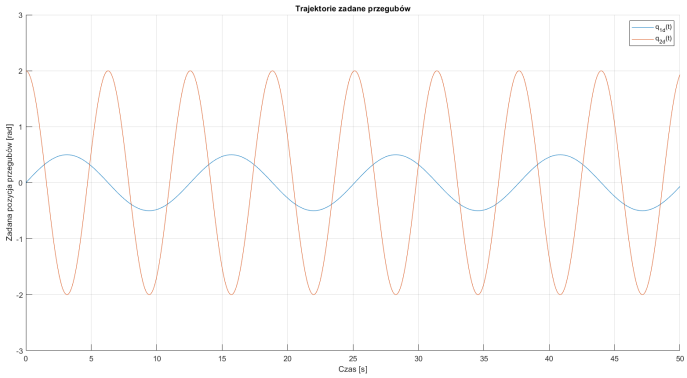
$$\dot{q}_{1d}(0) = \frac{1}{4}, \tag{46}$$

$$\dot{q}_{2d}(0) = 0, \tag{47}$$

$$q_{1d}(0) = 0, \tag{48}$$

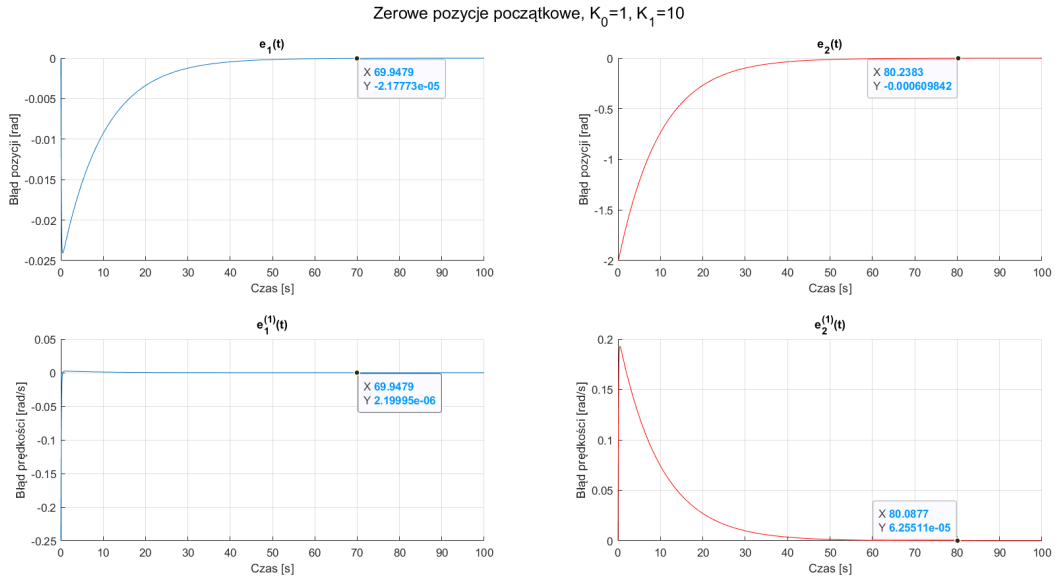
$$q_{2d}(0) = 2. \tag{49}$$

Zaimplementowane przebiegi trajektorii ukazano na wykresie 7.

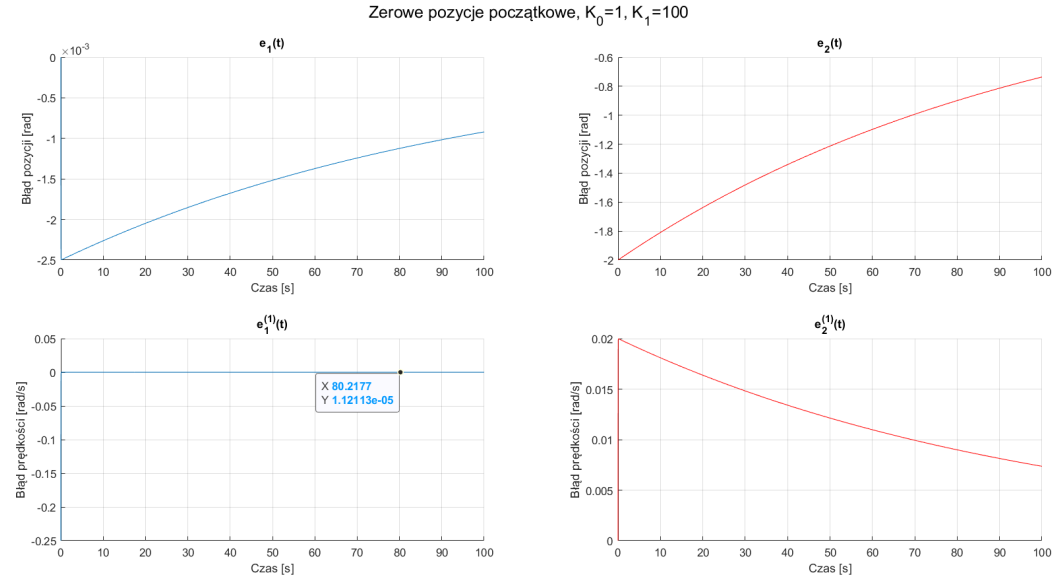


Rysunek 7: Przebieg trajektorii zadanych dla algorytmu dokładnej linearyzacji.

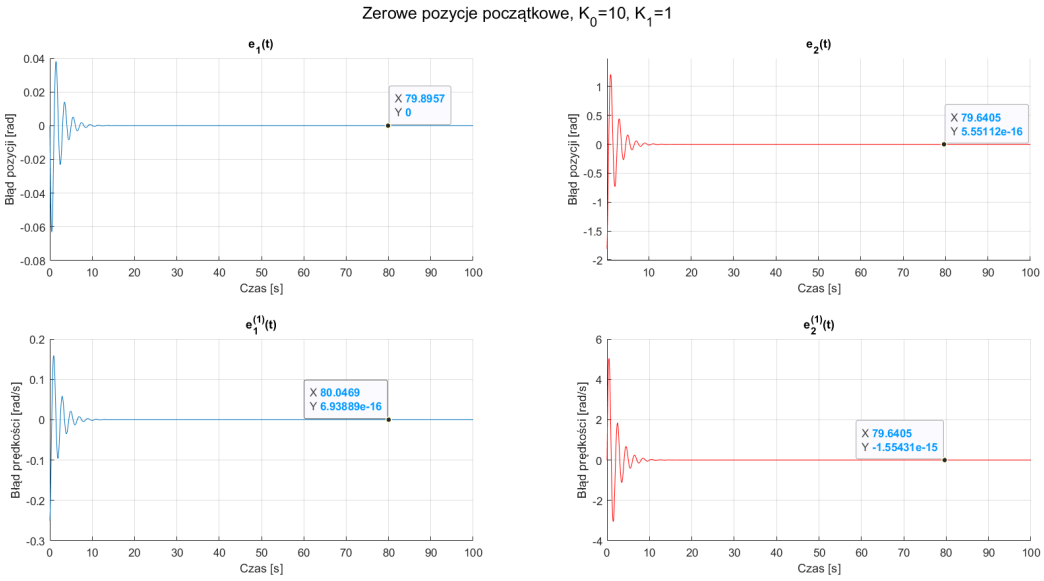
Przeprowadzono badania symulacyjne dla różnych wartości parametrów K_0 oraz K_1. W przypadku, gdy wartość $K_0 < K_1$ (rys. 8) widać, że wartości błędów konsekwentnie, „po delikatnym łuku" zbiegają do zera. Im wartość parametru K_1 była wyższa (rys. 9), tym więcej czasu było potrzebnego do ustabilizowania się. W przypadku, gdy $K_0 > K_1$ (rys. 8) początkowo wartości błędów oscylują wokół zera, by następnie do niego zbiec. Im większa wartość parametru K_0 w tej zależności, tym więcej oscylacji następuje na początku (z większą częstotliwością), co widać na rys. 11. Gdy wartości parametrów są sobie równe, to błędy bez oscylacji stabilizują się na zerze (rys. 12), jednak następuje przeregulowanie (im większe wartości parametrów, tym to przeregulowanie jest mniejsze - rys. 13).



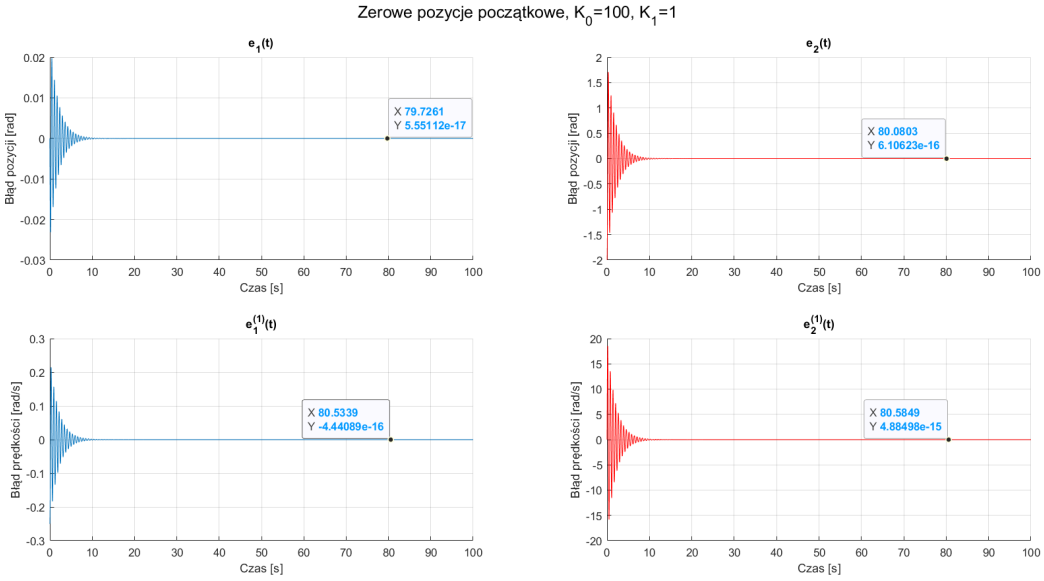
Rysunek 8: Błędy położeń i prędkości dla $K_0 = 1$ i $K_1 = 10$.



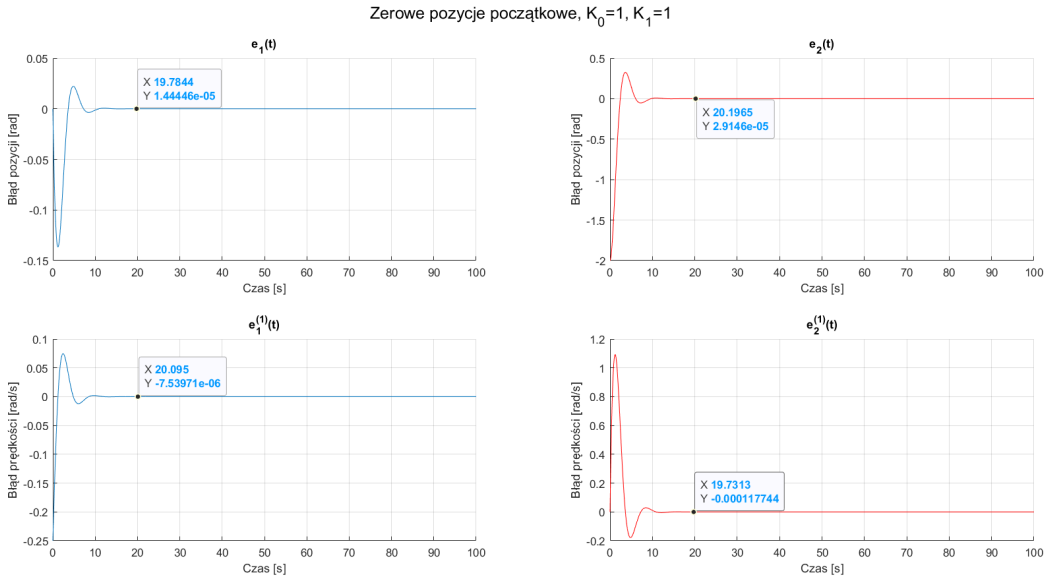
Rysunek 9: Błędy położeń i prędkości dla $K_0 = 1$ i $K_1 = 100$.



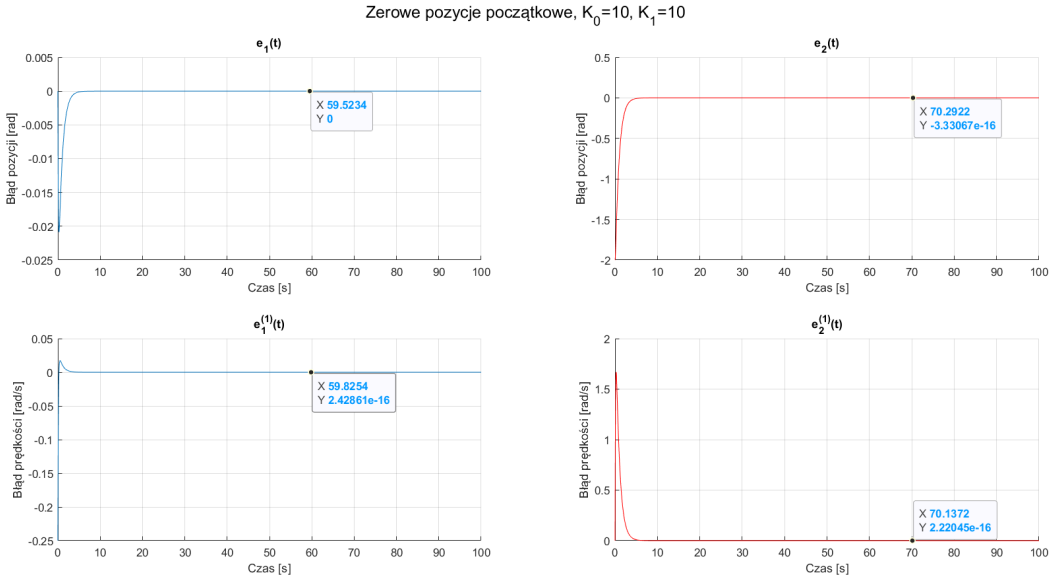
Rysunek 10: Błędy położeń i prędkości dla $K_0 = 10$ i $K_1 = 1$.



Rysunek 11: Błędy położeń i prędkości dla $K_0 = 100$ i $K_1 = 1$.

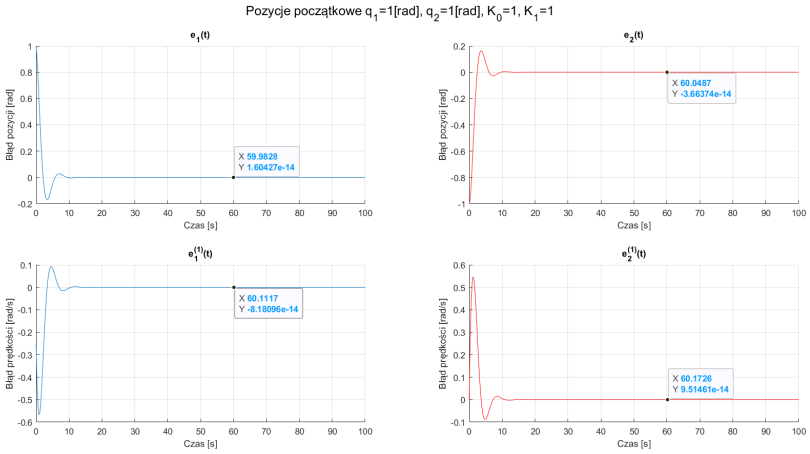


Rysunek 12: Błędy położeń i prędkości dla $K_0 = 1$ i $K_1 = 1$.

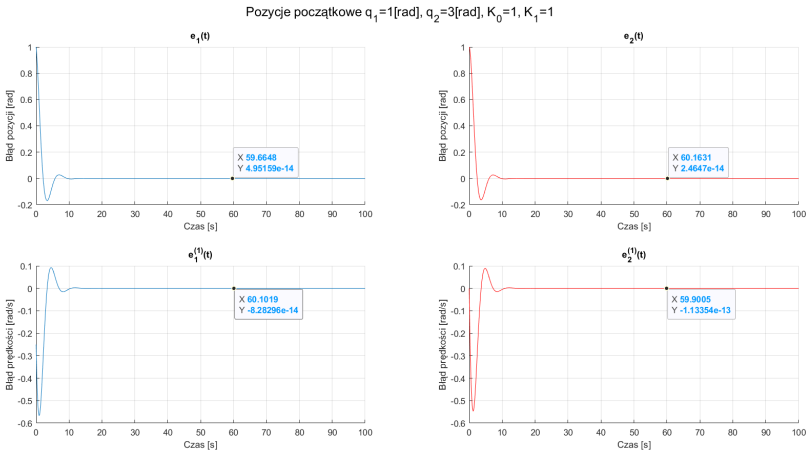


Rysunek 13: Błędy położeń i prędkości dla $K_0 = 10$ i $K_1 = 10$.

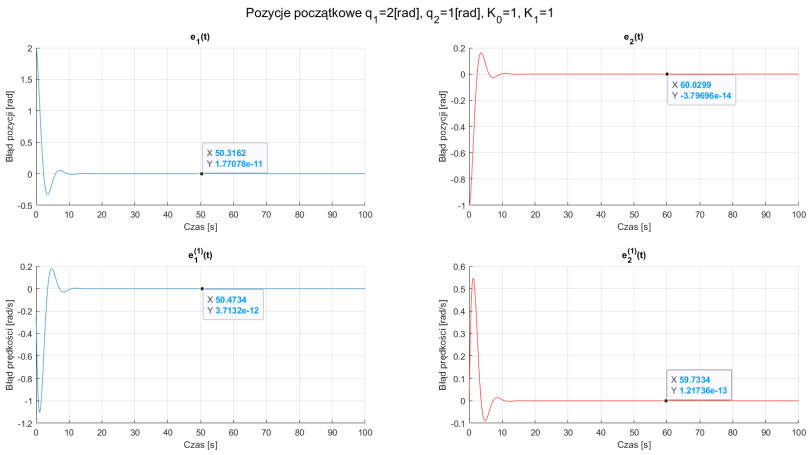
Symulacje zostały także przeprowadzone dla różnych położeń początkowych. Dla parametrów $K_1 = 1$, $K_0 = 1$ zasymulowano trzy sytuacje, dla których pary (q_1, q_2) wybrano ze zbioru $\{\{1, 1\}, \{1, 3\}, \{2, 1\}\}$ - rys. 14, 15, 16. Można zauważyć, że równie dobrze sobie algorytm radzi dla innych pozycji (nawet lepiej niż w przypadku zerowych pozycji początkowych rozważając rząd błędu, tu poniżej 10^{-10}). Różnią się wartości, z których robot zaczyna ruch, zatem widać, że błędy pozycji pierwszego przegubu na rys. 15 i 16 zaczynają zbiegać do zera z innych wartości (nawet o innym znaku).



Rysunek 14: Błędy położeń i prędkości dla położeń początkowych $q_1 = 1$[rad] i $q_2 = 1$[rad].



Rysunek 15: Błędy położeń i prędkości dla położeń początkowych $q_1 = 1$[rad] i $q_2 = 3$[rad].



Rysunek 16: Błędy położeń i prędkości dla położeń początkowych $q_1 = 2$[rad] i $q_2 = 1$[rad].

3 Wnioski

- Oba algorytmy (Qu i Dorsey'a oraz dokładnej linearyzacji) umożliwiają sterowaniem manipulatorem EDDA w taki sposób, by błędy prędkości i położeń przegubów zbiegały do zera z dokładnością zależną od parametrów regulatorów.
- Dla algorytmu Qu i Dorsey'a, im większe wartości wzmocnień regulatora PD, tym mniejszy rząd błędów. Ponadto, w ogólności można zauważyć trend, że przy zwiększaniu wzmocnień dziesięciokrotnie, rząd błędów zmniejsza się dziesięciokrotnie.
- Algorytm dokładnej linearyzacji lepiej radzi sobie z minimalizacją błędów - już praktycznie dla parametrów $K_0 = 1$, $K_1 = 1$ otrzymujemy błędy na poziomie 10^{-5}. Udało się zejść do błędów poniżej 10^{-10}, gdzie przy algorytmie Qu i Dorsey'a otrzymanie takiej dokładności wiązało się z długim oczekiwaniem na wynik w symulacji.
- Zależności pomiędzy parametrami w algorytmie dokładnej linearyzacji wpływają na to, w jaki sposób błąd dąży do zera - czy na początku są oscylacje (i o jakiej częstotliwości), jak duże jest przeregulowanie i jak szybko stabilizacja następuje.
- Algorytm dokładnej linearyzacji radzi sobie z śledzeniem trajektorii równie dobrze z pozycjami początkowymi różnymi od zera.