POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AIR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA MAGISTERSKA

Zadanie sterowania robota mobilnego napędzanego dwiema półsferami

Control problem for two HOG wheel mobile robot

AUTOR:
Paweł Joniak

PROWADZĄCY PRACĘ:
dr inż. Robert Muszyński

OCENA PRACY:

WROCŁAW 2017

Spis treści

Rozdział 1

Wprowadzenie

Od zarania dziejów ludzie wyrażali pragnienie przemieszczania się. Odkrycie koła niewątpliwie ułatwiło realizację tego celu. Od tego czasu wiele się zmieniło, jednak podstawą transportu naziemnego wciąż stanowi proste koło. Czas to zmienić, czas na kolejną rewolucję, czas na wirującą półsferę*! Pierwsze prace na napędem z użyciem wirującej półsfery rozpoczęto prawie 100 lat temu [7], jednak przez długie lata napęd ten nie zyskał popularności.

Niniejsza praca stanowi kontynuację badań nad robotem mobilnym napędzanym dwiema wirującymi półsferami rozpoczętych w [3]. Poprzednio zostały przedstawione niewątpliwe zalety napędu typu HOG w postaci ogromnej zwrotności i natychmiastowych przyśpieszeń. Wyprowadzono wtedy jedynie modele kinematyki robota oraz zbadano jego podstawowe zachowanie, jednak żeby w pełni wykorzystać potencjał robota z takim napędem należy nauczyć się nad nim panować. W wyobraźni autora narodziła się wizja świata w którym wszystkie kołowe pojazdy zostały wyparte przez autonomiczne roboty mobilne napędzane półsferami, które same zaplanują i wyznaczą trasę a potem zawiozą nas do wybranej destynacji. Wprawdzie ta wizja jest dalece odległa, ale pierwszym krokiem w jej kierunku będzie zapewne dokładne zrozumienie możliwości sterowania takim robotem. Ten właśnie problem stara się rozwiązać niniejsza praca.

Celem pracy jest omówienie zagadnienia sterowania robotem mobilnym w kontekście robota poruszającego się na dwóch wirujących półsferach. Przedstawiane zostaną zadania sterowania oraz algorytmy pozwalające je rozwiązać. Pokazane zostanie jak zaadaptować istniejące i dobrze znane algorytmy w przypadku naszego robota a także zaprezentowane zostaną nowe. Następnie przeprowadzone zostaną badania symulacyjne w celu sprawdzenia działania, skuteczności oraz możliwości implementacji poszczególnych algorytmów.

Układ pracy jest następujący. W rozdziale 2 zdefiniowaliśmy zadanie sterownia oraz scharakteryzowaliśmy ogólne algorytmy sterowania. Następnie w rozdziale 3 przytoczyliśmy wyprowadzone w [3] modele i pokazaliśmy implementacje algorytmów dla nich oraz zaproponowaliśmy autorskie rozwiązania problemu sterowania. W dalszej części (rozdział 4) przedstawiliśmy wyniki badań symulacyjnych pokazujące działanie poszczególnych algorytmów, omówiliśmy również ich wady, zalety oraz możliwość wykorzystania/implementacji. Praca została podsumowana w rozdziale 5.

*a nawet dwie

Rozdział 2

Algorytmy sterowania

Po uwzględnieniu ograniczeń, którym podlega robot mobilny, jego model przedstawia się za pomocą bezdryfowego układu sterowania w postaci [6]

$$\dot{q} = G(q)\eta, \tag{2.1}$$

gdzie $q \in \mathbb{R}^n$, $\dot{q} \in \mathbb{R}^n$ oznaczają odpowiednio wektor zmiennych konfiguracyjnych oraz ich prędkości, $G(q) \in \mathbb{R}^{n\times m}$ jest macierzą sterowań, zaś $\eta \in \mathbb{R}^m$ wektorem prędkości pomocniczych (sterowań). Dla tak opisanego modelu robota proponuje się różnorakie algorytmy sterowania, pozwalające na rozwiązania zadania sterowania. Dla robotów mobilnych najczęściej definiuje się trzy rodzaje zadań sterowania [5]:

- sterowanie do punktu – dąży się do znalezienia takich sterowań η, aby robot dojechał i zatrzymał się w ustalonej konfiguracji q_d,
- podążanie po zadanej ścieżce – celem tego jest wyznaczenie sterowań η, które pozwolą prowadzić robota po zadanej ścieżce $q_d(s)$ parametryzowanej odległością krzywoliniową s,
- śledzenie zadanej trajektorii – w tym przypadku należy tak dobrać sterowania η, aby robot podążał po zadanej trajektorii $q_d(t)$, tym samym błąd $e = q - q_d$ dążył do zera.

Warto tutaj dodać, że w postawionych powyżej zadaniach sterowania nie zawsze jesteśmy zainteresowani określeniem wartości pożądanych dla całego wektora q*. Częstokroć równania (2.1) są uzupełniane dodatkowym odwzorowaniem $h(q)$, stanowiącym funkcję wyjścia układu.

Poniżej zostaną pokrótce scharakteryzowane dwa algorytmy pozwalające śledzić zadaną trajektorię: algorytm linearyzacji statycznej i dynamicznej oraz jeden algorytm śledzenia ścieżki, które to w dalszej części pracy zostaną zastosowane do sterowania robotem napędzanym półsferami.

2.1 Linearyzacja statyczna

Linearyzacja statyczna polega na wyborze takich wyjść linearyzujących, by następnie poprzez sprzężenie zwrotne uzyskać sterowania umożliwiające ich śledzenie [2]. W tym

*w wielu przypadkach rozwiązanie takiego zadania jest wręcz niemożliwe

celu wybierzmy współrzędne linearyzujące w postaci odwzorowania[†]

$$h(q) : \mathbb{R}^n \to \mathbb{R}^m, \tag{2.2}$$

nazywanego wyjściem linearyzującym. Po policzeniu jego pochodnej

$$\dot{h}(q) = \frac{\partial h}{\partial q}\dot{q}, \tag{2.3}$$

i podstawieniu $\dot{q}$ z układu bezdryfowego (2.1) otrzymujemy

$$\dot{h}(q) = \frac{\partial h}{\partial q}\dot{q} = \frac{\partial h}{\partial q}G\eta. \tag{2.4}$$

Przyjmijmy następnie η równe

$$\eta = \left(\frac{\partial h}{\partial q}G\right)^{-1} u, \tag{2.5}$$

gdzie u to nowy wektor sterowania, a $R = \frac{\partial h}{\partial q}G$ to macierz odsprzęgania. W ten sposób, zakładając, że wyznacznik macierzy R jest niezerowy, otrzymujemy model układu z nowym prostym równaniem różniczkowym

$$\dot{h} = u, \tag{2.6}$$

dla którego sprzężenie zwrotne

$$u = \dot{h}_d - K(h - h_d), \tag{2.7}$$

gdzie h_d to zadane h, $K \in \mathbb{R}^{m\times m}$ – dodatnio określona macierz wzmocnień, pozwala na rozwiązanie zadania sterowania. Ostatecznie otrzymujemy algorytm sterowania opisany wzorem w postaci

$$\eta = \left(\frac{\partial h}{\partial q}G\right)^{-1} (\dot{h}_d - K(h - h_d)), \tag{2.8}$$

nazywamy algorytmem sterowania poprzez statyczne sprzężenie zwrotne.

2.2 Linearyzacja dynamiczna

Linearyzacja dynamiczna [2] jest zazwyczaj stosowana tam, gdzie zawodzi linearyzacja statyczna ($\det\left(\frac{\partial h}{\partial q}G\right) = 0$). W celu znalezienia sterowań ponownie wybierzmy współrzędne linearyzujące w postaci odwzorowania

$$h(q) : \mathbb{R}^n \to \mathbb{R}^m. \tag{2.9}$$

Tym razem należy policzyć drugą pochodną $\ddot{h}$ każdorazowo podstawiając $\dot{q}$ z układu bezdryfowego (2.1). Następnie rozszerzamy przestrzeń stanu układu poprzez wybór nowych zmiennych sterujących. Wybór uzależniony jest od przypadku, jednak najogólniejsza metoda polega na wybraniu pochodnych zmiennych pomocniczych $\dot{\eta}$. Kolejno przekształcamy $\ddot{h}$ do postaci

$$\ddot{h} = K_{dd}u + P, \tag{2.10}$$

[†]Przykładowe odwzorowania dla standardowych robotów mobilnych zostały przedstawione w [2]

gdzie u oznacza nowe sterowanie (w ogólnym przypadku $\dot{\eta}$). Zakładając, że wyznacznik macierzy $\det K_{dd}$ jest niezerowy, możemy przyjąć za u

$$u = K_{dd}^{-1}(-P + v). \tag{2.11}$$

Takie podstawienie sprowadza równanie (2.10) do postaci podwójnego integratora

$$\ddot{h} = v. \tag{2.12}$$

Przyjmując teraz sterowanie v jako

$$v = \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d), \tag{2.13}$$

gdzie $K_1, K_2 \in \mathbb{R}^{m\times m}$ – dodatnio określone macierze wzmocnień, h_d – zadane h, można zagwarantować śledzenie zadanej trajektorii h_d. Ostatecznie otrzymamy równania na pochodne sterowań

$$\dot{\eta} = u = K_{dd}^{-1}(-P + \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d)). \tag{2.14}$$

W ten sposób układ (2.1) został rozszerzony o dodatkowe zmienne

$$\begin{pmatrix} \dot{q} \\ \dot{\eta} \end{pmatrix} = \begin{pmatrix} G\eta \\ K_{dd}^{-1}(-P + \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d)) \end{pmatrix}. \tag{2.15}$$

Jeśli w $\ddot{h}$ nie występuje któraś z składowych $\dot{\eta}$ w jej miejsce można wybrać po prostu η. Wówczas z równania (2.15) otrzymamy bezpośrednio wzór na sterowanie a nie jego pochodną.

2.3 Algorytm Samsona

Algorytm Samsona [1], w odróżnieniu od przedstawionych dwóch poprzednich, pozwala na rozwiązanie zadania podążania po ścieżce. W tym, miejscu został on wspomniany jedynie dla kompletności rozdziału, gdyż jego postać zależy od modelu układu, dla którego został zaproponowany i stąd jego konstrukcję, jak i sposób wykorzystania, opisano w podrozdziale 3.2.3.

Rozdział 3

Implementacja algorytmów sterowania

Głównym zadaniem postawionym w pracy jest próba sterowania robotem poruszającym się na dwóch półsferach. W pracy [3] wyprowadzono jego różne modele. Poniżej dla kompletności pracy je przytoczymy, a następnie przedstawimy implementacje algorytmów sterowania z rozdziału 2 (oraz kilku nowych) dla tychże. Każdorazowo pokazane zostanie jakie wybrać odwzorowania linearyzujące i jakie ma to następstwa. Omówione zostaną również wady i zalety poszczególnych rozwiązań*.

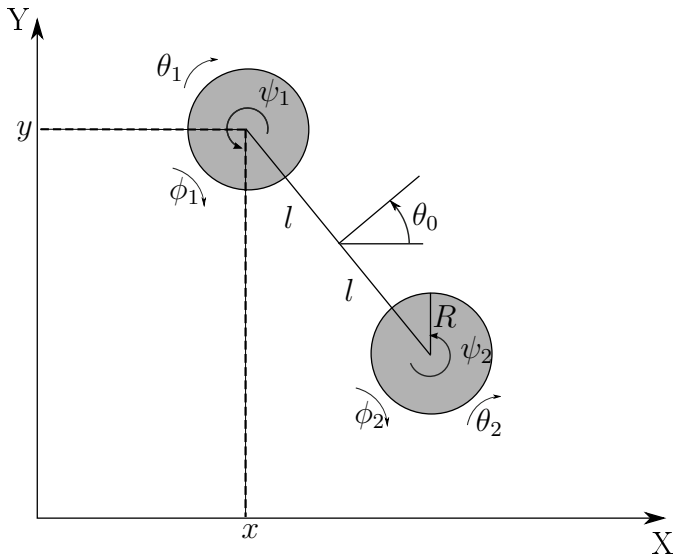
3.1 Modele pełne

W pracy [3] został przedstawiony model robota poruszającego się na dwóch półsferach, opisanego wektorem konfiguracyjnym $q = (x, y, \theta_0, \phi_1, \theta_1, \psi_1, \phi_2, \theta_2, \psi_2)^T$ (zobacz rysunek 3.1), gdzie x, y – położenie środka pierwszej półsfery, θ_0 – orientacja robota, ϕ_i, θ_i wychylenia i-tej półsfery a ψ_i jej wirowanie, $i = 1, 2$. Następnie wyprowadzone zostały dwa równoważne układy bezdryfowe (2.1), dane jako:

- model robota z bezpośrednim sterowaniem wychyleniami i jednym wirowaniem

$$\begin{cases} \dot{x} = Rs_{\theta_0}\eta_1 + Rc_{\theta_0}c_{\phi_1}\eta_2 + R(s_{\theta_0}s_{\theta_1} - c_{\theta_0}c_{\theta_1}s_{\phi_1})\eta_3 \\ \dot{y} = -Rc_{\theta_0}\eta_1 + Rs_{\theta_0}c_{\phi_1}\eta_2 - R(c_{\theta_0}s_{\theta_1} + s_{\theta_0}c_{\theta_1}s_{\phi_1})\eta_3 \\ \dot{\theta}_0 = \frac{R}{2l}(-\cot\theta_2 s_{\phi_2}\eta_1 - c_{\phi_1}\eta_2 + (c_{\theta_1}s_{\phi_1} - \cot\theta_2 s_{\theta_1}s_{\phi_2})\eta_3 + \\ \qquad\qquad + \cot\theta_2 s_{\phi_2}\eta_4 + c_{\phi_2}\eta_5) \\ \dot{\phi}_1 = \eta_1 \\ \dot{\theta}_1 = \eta_2 \\ \dot{\psi}_1 = \eta_3 \\ \dot{\phi}_2 = \eta_4 \\ \dot{\theta}_2 = \eta_5 \\ \dot{\psi}_2 = \csc\theta_2(\eta_1 + \sin\theta_1\eta_3 - \eta_4) \end{cases}, \tag{3.1}$$

*Mniej wprawnemu czytelnikowi zaleca się po zapoznaniu z opisaną poniżej strukturą robota rozpoczęcie tego rozdziału od podrozdziału 3.3.



Rysunek 3.1: Model robota poruszającego się na dwóch półsferach

- model robota z bezpośrednim sterowaniem wirowaniami i trzema wychyleniami

$$\begin{cases} \dot{x} = Rs_{\theta_0}\eta_1 + Rc_{\theta_0}c_{\phi_1}\eta_2 + R(s_{\theta_0}s_{\theta_1} - c_{\theta_0}c_{\theta_1}s_{\phi_1})\eta_3 \\ \dot{y} = -Rc_{\theta_0}\eta_1 + Rs_{\theta_0}c_{\phi_1}\eta_2 - R(c_{\theta_0}s_{\theta_1} + s_{\theta_0}c_{\theta_1}s_{\phi_1})\eta_3 \\ \dot{\theta}_0 = \frac{R}{2l}(-c_{\phi_1}\eta_2 + c_{\theta_1}s_{\phi_1}\eta_3 + c_{\phi_2}\eta_4 - c_{\theta_2}s_{\phi_2}\eta_5) \\ \dot{\phi}_1 = \eta_1 \\ \dot{\theta}_1 = \eta_2 \\ \dot{\psi}_1 = \eta_3 \\ \dot{\phi}_2 = \eta_1 + s_{\theta_1}\eta_3 - s_{\theta_2}\eta_5 \\ \dot{\theta}_2 = \eta_4 \\ \dot{\psi}_2 = \eta_5 \end{cases}, \tag{3.2}$$

gdzie R oznacza promień półsfery, l to połowa odległości między środkami półsfer zaś $c_\square$ i $s_\square$ to standardowo $\cos\square$ i $\sin\square$. Układy te nazywać będziemy modelami pełnymi robota.

W tym miejscu warto wskazać kilka właściwości powyższych modeli i budowy fizycznego robota. Po pierwsze, w modelu (3.1) występuje dzielenie przez $\sin\theta_2$, należy więc unikać sterowań, które prowadzą do zerowania się kąta θ_2, gdyż może to prowadzić do pojawienia się ogromnych wartości sterowania $\dot{\psi}_2$. Kolejno zauważmy iż w modelu (3.2) nie możemy bezpośrednio sterować kątem ϕ_2, co może prowadzić do pojawienia się wartości tego kąta przekraczających fizyczne ograniczenia wynikające z konstrukcji rzeczywistego robota[†] ($\phi_i, \theta_i \in [-\frac{\pi}{2}, \frac{\pi}{2}]$) nie uwzględnione w ograniczeniach nieholonomicznych. Poniżej przedstawimy kilka algorytmów sterowania dla obu wyprowadzonych modeli.

[†]W pracy [3] pokazana została przykładowa konstrukcja robota oraz omówiono ograniczenia fizyczne wynikające z takiej budowy.

3.1.1 Linearyzacja statyczna

W celu zastosowania algorytmu linearyzacji statycznej do opisanego powyżej robota, jako wektor wyjść wybierzmy trywialne odwzorowanie w postaci

$$h(q) = \begin{pmatrix} x \\ y \\ \theta_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}. \tag{3.3}$$

Wybór trzech pierwszych wyjść jest oczywisty, gdyż chcemy mieć możliwość sterowania pozycją i orientacją robota. Kąty ψ_i zostały wybrane, ponieważ w rzeczywistym robocie wygodnie jest, aby prędkość wirowania półsfer była stała, dodatkowo zwiększając ją możemy zmniejszyć amplitudy wychyleń. Dla tak wybranego wektora stosujemy algorytm (2.8). Należy jednakże zwrócić uwagę na wartość wyznacznika macierzy $\frac{\partial h}{\partial q}G$. Dla modelu (3.1) wynosi ona

$$\det\left(\frac{\partial h}{\partial q}G\right) = -\frac{R^3 \csc\theta_2 \cos\phi_1 \cos\phi_2}{2l}, \tag{3.4}$$

a dla modelu (3.2)

$$\det\left(\frac{\partial h}{\partial q}G\right) = -\frac{R^3 \cos\phi_1 \cos\phi_2}{2l}. \tag{3.5}$$

Jak wspomniano wcześniej, fizyczne ograniczenia robota sprawiają, że kąty ϕ_i nie mogą osiągnąć wartości $\frac{\pi}{2}$, a co za tym idzie sterowanie działa w dopuszczalnych zakresach wychyleń. Dodatkowo na wyznacznik (3.4) jest postawione takie samo ograniczenie jak na sam model (3.1), to znaczy $\theta_2 \neq 0$.

3.1.2 Linearyzacja dynamiczna

W celu zastosowania algorytmu linearyzacji dynamicznej, jako wektor wyjść wybierzmy ponownie trywialne odwzorowanie w postaci

$$h(q) = \begin{pmatrix} x \\ y \\ \theta_0 \\ \psi_1 \\ \psi_2 \end{pmatrix}. \tag{3.6}$$

Kolejno postępujemy zgodnie z zasadą opisaną w podrozdziale 2.2. W tym przypadku należy zwrócić uwagę na wartość wyznacznika macierzy K_{dd}. Dla modelu (3.1) wynosi ona

$$\det K_{dd} = -\frac{R^3 \csc\theta_2 \cos\phi_1 \cos\phi_2}{2l}, \tag{3.7}$$

a dla modelu (3.2)

$$\det K_{dd} = -\frac{R^3 \cos\phi_1 \cos\phi_2}{2l}. \tag{3.8}$$

Jak widać oba wyznaczniki są takie same jak w przypadku linearyzacji statycznej (podrozdział 3.1.1).

3.1.3 Algorytm JPTD

Jak zostało to opisane w podrozdziale 3.3.5, w trakcie aplikacji algorytmów sterowania do modelu symultanicznego robota (podrozdział 3.3) pojawił się pomysł ich uproszczenia, prowadzący do opracowania algorytmu nazwango przez autora JPTD. Poniżej przedstawimy implementację algorytmu JPTD dla modelu (3.2)[‡].

W pierwszej kolejności zauważmy, że sterowanie wychyleniami η_1, η_2 i η_4 ma znacząco mniejszy wpływ (nawet o rząd) na prędkość postępową robota oraz zmianę jego orientacji niż prędkości wirowania półsfer η_3 i η_5. Przy takim założeniu można uprościć macierz sterowań G modelu (3.2) z postaci

$$G(q) = \begin{bmatrix} Rs_{\theta_0} & Rc_{\theta_0}c_{\phi_1} & Rcc_1 & 0 & 0 \\ -Rc_{\theta_0} & Rs_{\theta_0}c_{\phi_1} & -Rsc_1 & 0 & 0 \\ 0 & -\bar{R}c_{\phi_1} & -\bar{R}c_{\theta_1}s_{\phi_1} & -\bar{R}c_{\phi_2} & -\bar{R}c_{\theta_2}s_{\phi_2} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & s_{\theta_1} & 0 & -s_{\theta_1} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.9}$$

gdzie $cc_1 = s_{\theta_0}s_{\theta_1} - c_{\theta_0}c_{\theta_1}s_{\phi_1}$, $sc_1 = (c_{\theta_0}s_{\theta_1} - s_{\theta_0}c_{\theta_1}s_{\phi_1})$, $\bar{R} = \frac{R}{2l}$, do postaci

$$G(q) = \begin{bmatrix} 0 & 0 & Rcc_1 & 0 & 0 \\ 0 & 0 & -Rsc_1 & 0 & 0 \\ 0 & 0 & -\bar{R}c_{\theta_1}s_{\phi_1} & 0 & -\bar{R}c_{\theta_2}s_{\phi_2} \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & s_{\theta_1} & 0 & -s_{\theta_1} \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.10}$$

Następnie używamy tak uproszczoną macierz w algorytmie linearyzacji dynamicznej (podrozdział 2.2)[§]. Odwzorowanie $h(q)$ wybieramy takie samo jak w podrozdziale 3.1.2. Skutkuje to wyznacznikiem macierzy K_{dd} w postaci

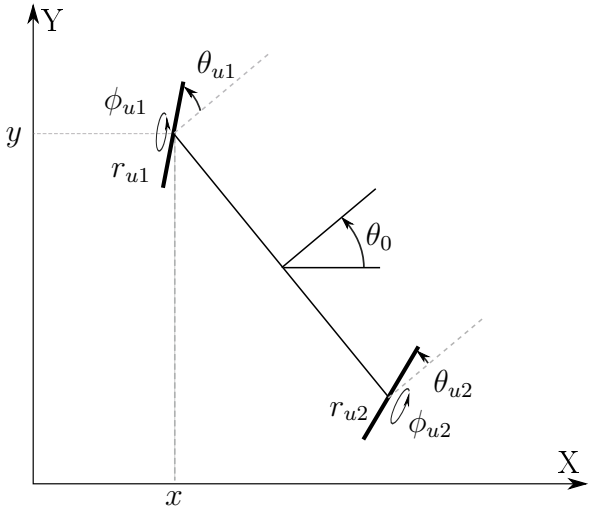
$$\det\left(K_{dd}\right) = -\frac{R^3{\eta_3}^2\eta_5 \sin\theta_2 \cos^2\theta_1 \sin\phi_2 \cos\phi_1}{2l}. \tag{3.11}$$

Oznacza to, że dla poprawnego działania algorytmu obie półsfery muszą wirować, a wychylenia ϕ_2 i θ_2 nie mogą być zerowe.

3.2 Model uproszczony

W pracy [3] zostało pokazane podobieństwo pomiędzy modelem robota poruszającego się na 2 półsferach a robotem klasy (1, 2) z dwoma skrętnymi kołami o zmiennych promieniach (zobacz rysunek 3.2) , opisanego wektorem zmiennych konfiguracyjnych

[‡]Analogiczną metodę uproszczania macierzy G próbowano zastosować dla modelu (3.1) jednak nie udało się znaleźć odwzorowań przy których wyznaczniki $\det\left(\frac{\partial h}{\partial q}G\right)$ (dla linearyzacji statycznej) lub



Rysunek 3.2: Model robota z dwoma skrętnymi kołami

$q = (x, y, \theta_0, \theta_{u1}, \phi_{u1}, \theta_{u2}, \phi_{u2}, r_{u1}, r_{u2})^T$, gdzie x, y – położenie środka pierwszego koła, θ_0 – orientacja robota, θ_{ui} – skręcenie i-tego koła, ϕ_{ui} – jego wirowanie a r_{ui} – jego promień, $i = 1, 2$. Wyprowadzony także został układ bezdryfowy opisujący takiego robota w postaci

$$\begin{cases} \dot{x} = \cos(\theta_0 + \theta_{u1}) r_{u1} \eta_2 \\ \dot{y} = \sin(\theta_0 + \theta_{u1}) r_{u1} \eta_2 \\ \dot{\theta}_0 = \csc\theta_{u2} \sin(\theta_{u1} - \theta_{u2}) \frac{r_{u1}}{2l} \eta_2 \\ \dot{\theta}_{u1} = \eta_1 \\ \dot{\phi}_{u1} = \eta_2 \\ \dot{\theta}_{u2} = \eta_3 \\ \dot{\phi}_{u2} = \csc\theta_{u2} \sin\theta_{u1} \frac{r_{u1}}{r_{u2}} \eta_2 \\ \dot{r}_{u1} = \eta_4 \\ \dot{r}_{u2} = \eta_5 \end{cases}, \tag{3.12}$$

gdzie l to połowa odległości pomiędzy kołami. W powyższym modelu występuje dzielenie przez $\sin\theta_{u2}$, a więc należy tak dobierać sterowanie, aby uniknąć sytuacji, gdy kąt θ_{u2} zeruje się. Należy jednak zwrócić uwagę no to, że gdy oba kąty θ_{u1} i θ_{u2} równocześnie osiągną zero problem nie występuję. Kolejno zauważmy, iż w modelu (3.12) prędkość postępową kół możemy zwiększyć zarówno poprzez bezpośrednie sterowanie prędkością ich wirowania jak i zmianą wielkości promienia. Bardziej intuicyjne wydaje się być sterowanie prędkością wirowania, gdyż przekłada się ona bezpośrednio na prędkość wirowania półsfer w modelu pełnym (zobacz równania 3.14).

W pracy [3] wyprowadzono również formuły określające przekształcenie pomiędzy zmiennymi stanu modelu pełnego i uproszczonego, przytoczone poniżej dla kompletności:

$$\begin{aligned} \phi_u &= \psi, \\ r_u &= R\sqrt{\cos^2\phi(\sin^2\theta - 1) + 1}, \\ \theta_u &= \arctan\frac{\sin\theta}{\cos\theta\sin\phi}, \end{aligned} \tag{3.13}$$

$\det K_{dd}$ (dla linearyzacji dynamicznej) byłyby niezerowe.

[§]Postępując analogicznie dla linearyzacji statycznej nie udało się znaleźć odwzorowania przy którym macierz odsprzęgania miałaby niezerowy wyznacznik.

oraz przekształcenia odwrotne:

$$\begin{aligned} \psi &= \phi_u, \\ \phi &= \pm \arccos\left(\frac{\sqrt{r_u^2 - R^2}\sqrt{\tan^2(\theta_u)+1}}{\sqrt{-R^2 - R^2\tan^2(\theta_u) + r_u^2\tan^2(\theta_u)}}\right), \\ \theta &= \pm \arcsin\left(\frac{r_u \tan(\theta_u)}{R\sqrt{\tan^2(\theta_u)+1}}\right). \end{aligned} \tag{3.14}$$

Poniżej zostaną pokazane algorytmy sterowania dla modelu (3.12).

3.2.1 Linearyzacja statyczna

Na początku ponownie zastosujemy algorytm linearyzacji statycznej. W tym przypadku odwzorowanie $h(q)$ nie będzie już trywialne. Aby uniknąć zerowego wyznacznika macierzy $\frac{\partial h}{\partial q}G$ jako punkt prowadzenia robota, stanowiący dwie pierwsze zmienne linearyzujące, nie wybierzemy jak poprzednio pozycji środka pierwszego koła robota x, y, a punkt przesunięty względem niego o odległość d, co prowadzi do wyjścia linearyzującego w postaci[¶]

$$h(q) = \begin{pmatrix} x + d\cos(\theta_{u1} + \theta_0) \\ y + d\sin(\theta_{u1} + \theta_0) \\ \theta_{u2} \\ r_{u1} \\ r_{u2} \end{pmatrix}. \tag{3.15}$$

Przy tak wybranym odwzorowaniu wspomniany wyznacznik przyjmuje wartość $-dr_{u1}$. Wybierając jako trzecie wyjście θ_{u2} możemy kontrolować jego wartość i nie dopuścić do tego, aby była równa zero[‖]. Jako dwa ostatnie wyjścia wybierzemy promienie kół. Jak już wspominaliśmy optymalnie byłoby tutaj sterować wirowaniem kół, jednak w przypadku linearyzacji statycznej jest to niemożliwe. Dodatkowo, zakładając że promienie r_{u1} oraz r_{u2} są stałe, a przekształcenie odwrotne (3.14) ma postać $\theta_2 = f(r_{u2}, \theta_{u2})$, taką że $\theta_2 = 0 \Leftrightarrow \theta_{u2} = 0$, nie dopuszczając aby $\theta_{u2} = 0$, nie dopuszczamy $\theta_2 = 0$, co jest kluczowe, gdy używamy modelu (3.1). Tak wybrane odwzorowanie zostaje następnie użyte zgodnie z zasadą opisaną w podrozdziale 2.1. Prowadzi to do otrzymania „przepisu" na sterowanie modelem uproszczonym.

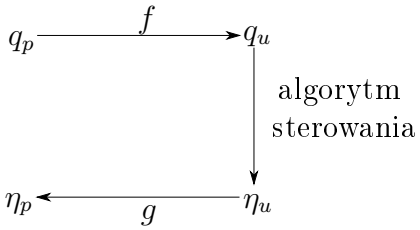
W tym przypadku pozostaje jeszcze kwestia przeniesienia sterowań wyliczanych dla modelu uproszczonego na model pełny. Możemy tego dokonać w dwójnasób: wyliczając sterowania dla układu pełnego po wyliczeniu sterowań dla modelu uproszczonego (tryb „offline"), albo też wyliczając je na bieżąco w trakcie stosowania algorytmu rozwiązującego zadanie dla modelu uproszczonego (tryb „online").

W przypadku „offline" w pierwszej kolejności wyznaczamy trajektorię ruchu układu (3.12), (3.15), (2.1). Kolejno wyliczone trajektorie przekształcamy na trajektorię modelu pełnego z użyciem przekształceń (3.14). Ostatecznie ich pochodne zadaje się jako sterowania modelu (3.1) lub (3.2). Takie postępowanie ma niewątpliwe zalety w postaci prostoty, jednak nie pozwala korygować sterowań już w trakcie jazdy modelem pełnym.

Drugą metodą jest przeniesienie „online". Tutaj implementujemy algorytm który na bieżąco przelicza układy i steruje modelem pełnym na podstawie sterowań wyliczonych dla modelu uproszczonego według schematu pokazanego na rysunku 3.3. W celu wyjaśnienia

[¶] takie postępowanie oznacza wprowadzenie w rzeczywistym robocie dyszla związanego z pierwszym kołem i określenie trajektorii pożądanej dla jego końca

[‖] jak wspomniano, gdy $\theta_{u2} = 0$ w modelu (3.12) pojawia się dzielenie przez zero



Rysunek 3.3: Sterownie w trybie „online”

tego sposobu postępowania przyjmijmy następujące oznaczenia:

- $q = (x, y, \theta_0)^T$ – współrzędne konfiguracyjne wspólne dla obu modeli,
- $q_p = (\theta_1, \phi_1, \theta_2, \phi_2, \psi_1, \psi_2)^T$ – współrzędne konfiguracyjne charakterystyczne dla modelu pełnego,
- $q_u = (\theta_{u1}, \phi_{u1}, \theta_{u2}, \phi_{u2}, r_{u1}, r_{u2})^T$ – współrzędne konfiguracyjne charakterystyczne dla modelu uproszczonego,
- $\eta_p = (\eta_1, \eta_2, \eta_3, \eta_4, \eta_5)^T$ – wektor sterowania dla modelu pełnego,
- $\eta_u = (\eta_{u1}, \eta_{u2}, \eta_{u3}, \eta_{u4}, \eta_{u5})^T$** – wektor sterowania dla modelu uproszczonego,
- f – transformacja zmiennych modelu pełnego do uproszczonego, $q_u = f(q_p)$ (na podstawie (3.13)),
- g – transformacja odwrotna do f, $q_p = g(q_u)$ (na podstawie (3.14)).

Modele (3.1), (3.2) zostały tak wyprowadzone, aby sterowania η_p odpowiadały bezpośrednio konkretnym pochodnym wektora q_p. Możemy zatem zamiast szukać sterowań η_p szukać prędkości $\dot{q}_p$. Zapiszmy więc $\dot{q}_p$ za pośrednictwem funkcji g

$$\dot{q}_p = \frac{d}{dt} g(q_u) = \frac{\partial g}{\partial q_u}(q_u)\dot{q}_u. \tag{3.16}$$

Za q_u przyjmujemy $f(q_p)$, natomiast pochodna $\dot{q}_u$ zgodnie z modelem (3.12) ma postać

$$\dot{q}_u = \begin{pmatrix} \dot{\theta}_{u1} \\ \dot{\phi}_{u1} \\ \dot{\theta}_{u2} \\ \dot{\phi}_{u2} \\ \dot{r}_{u1} \\ \dot{r}_{u2} \end{pmatrix} = \begin{pmatrix} \eta_{u1} \\ \eta_{u2} \\ \eta_{u3} \\ \csc\theta_{u2} \sin\theta_{u1} \frac{r_{u1}}{r_{u2}} \eta_{u2} \\ \eta_{u4} \\ \eta_{u5} \end{pmatrix}. \tag{3.17}$$

Sterowania η_u wyliczamy używając algorytmu linearyzacji statycznej jak pokazano powyżej. Otrzymujemy wówczas wzór na pochodne $\dot{q}_u$ w funkcji od q_u i q

$$\dot{q}_u = M(q_u, q). \tag{3.18}$$

Przyjmując $q_u = f(q_p)$, dostajemy sterowania dla modelu uproszczonego jako funkcję zmiennych stanu modelu pełnego. Ostatecznie możemy zapisać wyrażenie (3.16) jako

$$\dot{q}_p = \frac{d}{dt} g(q_u) = \frac{\partial g}{\partial q_u}(q_u)\dot{q}_u = \frac{\partial g}{\partial q_u}(f(q_p))M(f(q_p), q), \tag{3.19}$$

co określa sposób postępowania przy przenoszeniu sterowań „online”.

**Takie oznaczenie stosujemy tylko na potrzeby wytłumaczenia przeniesienia sterowań w trybie „online”. Dalej w pracy mówiąc o sterowaniach modelu uproszonego będziemy używać po prostu symbolu η.

3.2.2 Linearyzacja dynamiczna

W przypadku linearyzacji dynamicznej nie pojawia się problem z zerującym się wyznacznikiem macierzy odsprzęgania $\frac{\partial h}{\partial q}G$, stąd odwzorowanie $h(q)$ przyjmujemy ponownie w trywialnej postaci

$$h(q) = \begin{pmatrix} x \\ y \\ \theta_{u2} \\ r_{u1} \\ r_{u2} \end{pmatrix}. \tag{3.20}$$

Mając wybrane odwzorowanie wyliczmy sterowania zgodnie z zasadą opisaną w podrozdziale 2.2. Inaczej niż w poprzednich przypadkach linearyzacji dynamicznej układ nie rozszerzy się o pochodne wszystkich sterowań a tylko o $\dot{\eta}_2$, $\dot{\eta}_3$, $\dot{\eta}_4$ i $\dot{\eta}_5$. Wyznacznik macierzy K_{dd} wynosi $\det(K_{dd}) = -r_{u1}^2\eta_2$, a więc, aby sterowanie działało, układ musi pozostawać w ruchu ($\eta_2 \neq 0$).

Jak wspomniano w poprzednim podrozdziale, docelowo zależy nam na sterowaniu prędkością wirowania kół, a nie zmianą promieni. W przypadku linearyzacji dynamicznej można tego dokonać. W tym celu wybieramy alternatywne przekształcenie $h(q)$ jako

$$h(q) = \begin{pmatrix} x \\ y \\ \theta_{u2} \\ \phi_{u1} \\ \phi_{u2} \end{pmatrix}, \tag{3.21}$$

i stosujemy algorytm opisany w podrozdziale 2.2. Takie postępowanie komplikuje jednakże znacząco postać wyznacznika macierzy K_{dd}, który teraz jest opisany wzorem

$$\det(K_{dd}) = -\frac{r_{u1}^2{\eta_2}^3 \sin\theta_{\mathrm{u1}} \csc\theta_{\mathrm{u2}}}{r_{u2}^2}. \tag{3.22}$$

Trzeba także dodać, że rezygnacja ze sterowania promieniami sprawia, iż przekształcenie zmiennych stanu modelu uproszczonego do modelu pełnego zgodnie z formułami (3.14) może być niemożliwe.

W przypadku linearyzacji dynamicznej możliwe jest jedynie przeniesienie sterowań w trybie „offline" zarówno dla odwzorowania (3.20) jak i (3.21).

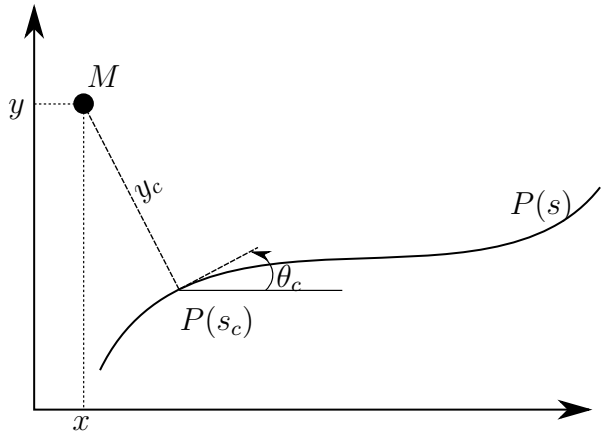
3.2.3 Algorytm Samsona

Przybliżymy teraz pokrótce algorytm śledzenia ścieżki dla robota mobilnego z dwoma skrętnymi kołami zaproponowany w [1]. W pierwszej kolejności wyprowadzimy opis układu względem danej krzywej. Wówczas, w miejsce standardowego opisu położenia punktu za pomocą współrzędnych kartezjańskich, określamy je przez podanie odległości pomiędzy nim a jego projekcją na zadaną krzywą i położeniem krzywoliniowym tej projekcji na tej krzywej (zobacz rysunek 3.4 – położenie punktu M określone jest współrzędnymi $(y_c, s_c)^T$). Oznaczmy przez θ_c kąt stycznej do ścieżki w punkcie $P(s_c)$ i przez $c_c(s)$ krzywiznę ścieżki, tak że

$$\dot{\theta}_c(s_c(t)) = c_c(s_c(t))\dot{s}_c(t). \tag{3.23}$$

Pochodna krzywizny względem s dana jest jako $g_c(s)$, tak że

$$\dot{c}_c(s(t)) = g_c(s(t))\dot{s}(t). \tag{3.24}$$



Rysunek 3.4: Położenie względem krzywej.

Teraz zadnie sterowania polega na takim dobraniu sterowań, aby odległość y_c spadła do zera, a kąt $\theta = \theta_0 - \theta_c$, będący różnicą pomiędzy orientacją robota a kątem stycznej do krzywej dążył do zadanej wartości θ_d.

W celu realizacji powyższego zadania w pracy [1] przedstawiono model robota z dwoma skrętnymi kołami podobny do tego wyprowadzonego w [3] i przytoczonego w podrozdziale 3.2[††]. Model ten został rozszerzony o odwrotność odległości od punktu obrotu robota σ, którą można zapisać jako

$$\sigma = \frac{1}{2l}(\sin\theta_{u1} - \tan\theta_{u2}\cos\theta_{u1}). \tag{3.25}$$

Ostatecznie, w [1] dla takiego układu zaproponowano algorytm sterowania oparty na linearyzującym sprzężeniu zwrotnym w postaci

$$\begin{cases} \dot{\theta}_{u2d} = \frac{\dot{\theta}_{u1}(1-2l\sigma\sin\theta_{u1})-2l\dot{\sigma}\cos\theta_{u1}}{(2l\sigma-\sin\theta u1)^2+\cos^2\theta_{u1}} \\ \dot{\theta}_{u2} = \frac{\dot{\theta}_{u1}(1-2l\sigma\sin\theta_{u1})-2l\dot{\sigma}\cos\theta_{u1}}{(2l\sigma-\sin\theta u1)^2+\cos^2\theta_{u1}} - k_{\theta_{u2}}(\dot{\theta}_{u2} - \dot{\theta}_{u2d}) \\ \dot{s}_c = v\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c} \\ \dot{y}_c = v s_{\theta+\theta_{u1}} \\ \dot{\theta} = v(\sigma - c_c\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}) \\ \dot{\theta}_{u1} = v\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}[y_c\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}(g_c s_{\theta+\theta_{u1}} - k_{p_y})+ \\ \qquad + s_{\theta+\theta_{u1}}(c_c s_{\theta+\theta_{u1}} - k_{v_y}c_{\theta+\theta_{u1}}sign(\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c})) + c_c] - v\sigma \\ \dot{\sigma} = v\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}\left\{\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}\left[-k_{p_\theta}\tilde{\theta} + g_c\right] + \sigma\left\{\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}y_c\left(g_c c_{\theta+\theta_{u1}} + k_{p_y}s_{\theta+\theta_{u1}}\right) + \right.\right. \\ \qquad \left.\left. + s_{\theta+\theta_{u1}}\left[c_c c_{\theta+\theta_{u1}} + k_{v_y}s_{\theta+\theta_{u1}}sign\left(\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}\right)\right]\right\} - k_{v_\theta}\left[\sigma - c_c\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}\right]sign\left(\frac{c_{\theta+\theta_{u1}}}{1-c_c y_c}\right)\right\} \\ \dot{\phi}_{u1} = v \\ \dot{\phi}_{u2} = v\sqrt{(2l\sigma - \sin\theta_{u1})^2 + \cos^2\theta_{u1}} \end{cases}, \tag{3.26}$$

gdzie $c_{\theta+\theta_{u1}} = \cos(\theta+\theta_{u1})$, $s_{\theta+\theta_{u1}} = \sin(\theta+\theta_{u1})$, $\tilde{\theta} = \theta - \theta_d$ – błąd orientacji robota względem ścieżki, $k_{\theta_{u2}}, k_{p_y}, k_{v_y}, k_{p_\theta}, k_{v_\theta}$ – dodatnie wzmocnienia a v to stała ustalona prędkość wirowania pierwszego koła.

[††]Modele różnią się orientacją początkową robota oraz kół – dodatkowo model [1] nie ma zmiennych promieni.

3.3 Model symultaniczny

W celu lepszego zrozumienia zachowania robota napędzanego dwiema półsferami, jego sterowania i metod ich implementacji, algorytmy zostały przetestowane na modelu, w którym obie półsfery poruszają się synchronicznie, to znaczy $\phi_1 = \phi_2 = \phi$, $\theta_1 = \theta_2 = \theta$ oraz $\psi_1 = \psi_2 = \psi$. Opisujący go układ bezdryfowy ma postać

$$\begin{cases} \dot{x} = R\left(\eta_2 \cos\theta_0 \cos\phi + \eta_3 \left(\sin\theta \sin\theta_0 - \cos\theta \cos\theta_0 \sin\phi\right) + \eta_1 \sin\theta_0\right) \\ \dot{y} = R\left(\eta_2 \sin\theta_0 \cos\phi - \eta_3 \left(\sin\theta_0 \cos\theta \sin\phi + \sin\theta \cos\theta_0\right) - \eta_1 \cos\theta_0\right) \\ \dot{\theta}_0 = 0 \\ \dot{\phi} = \eta_1 \\ \dot{\theta} = \eta_2 \\ \dot{\psi} = \eta_3 \end{cases} \tag{3.27}$$

z wektorem konfiguracyjnym $q = (x, y, \theta_0, \phi, \theta, \psi)^T$. Jak widać w powyższym modelu nie jesteśmy w stanie sterować orientacją robota θ_0, dlatego w celu dalszego uproszczenia przyjęto $\theta_0 = 0$ i je pominięto, co sprowadza układ do postaci

$$\begin{cases} \dot{x} = R\left(\cos\phi\eta_2 - \cos\theta \sin\phi\eta_3\right) \\ \dot{y} = -R\left(\eta_1 + \sin\theta\eta_3\right) \\ \dot{\phi} = \eta_1 \\ \dot{\theta} = \eta_2 \\ \dot{\psi} = \eta_3 \end{cases} . \tag{3.28}$$

Taki model będziemy nazywać symultanicznym. Co ważne, w modelu (3.28) nie występuje dzielenie przez 0 dla żadnych wartości zmiennych stanu.

Poniżej dla modelu symultanicznego opiszemy szczegółowo algorytmy sterowania zaprezentowane w rozdziale 2. W celu lepszego ich zrozumienia zostaną przedstawione kolejne kroki prowadzące do otrzymania za ich pomocą wyrażeń opisujących sterowania. Dodatkowo zaprezentujemy prosty algorytm sterowania, który eliminuje niektóre wady linearyzacji statycznej i dynamicznej. Omówione zostaną też próby poprawienia algorytmów linearyzacji.

3.3.1 Linearyzacja statyczna

Jako pierwsze pokażemy, jak zastosować algorytm linearyzacji statycznej do modelu (3.28). W tym celu, w roli wektora współrzędnych linearyzujących wybierzmy trywialne odwzorowanie w postaci

$$h(q) = \begin{pmatrix} x \\ y \\ \psi \end{pmatrix}. \tag{3.29}$$

Następnie, zgodnie z opisaną w podrozdziale 2.1 metodą, zapisujemy sterowanie η w postaci

$$\eta = \left(\frac{\partial h}{\partial q} G\right)^{-1} (\dot{h}_d - K(h - h_d)). \tag{3.30}$$

Wyznacznik macierzy $\det \frac{\partial h}{\partial q} G = R^2 \cos\phi$, a więc sterowanie jest możliwe gdy $\phi \neq \pm\frac{\pi}{2}$.

3.3.2 Linearyzacja dynamiczna

Kolejno do modelu (3.28) zastosujemy przedstawiony w podrozdziale 2.2 algorytm linearyzacji dynamicznej. Wybierzmy najpierw trywialne odwzorowanie w postaci

$$h(q) = \begin{pmatrix} x \\ y \\ \psi \end{pmatrix}. \tag{3.31}$$

Następnie policzmy drugą pochodą h i podstawmy zmienne z modelu (3.28) otrzymując

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{pmatrix} = \begin{pmatrix} -R(-\sin\theta\sin\phi\eta_2\eta_3 + \sin\phi\eta_1\eta_2 + \cos\theta\cos\phi\eta_3\eta_1 + \\ \qquad - \cos\phi\dot{\eta}_2 + \cos\theta\sin\phi\dot{\eta}_3) \\ -R(\cos\theta\eta 2\eta_3 + \dot{\eta}_1 + \sin\theta\dot{\eta}_3) \\ \dot{\eta}_3 \end{pmatrix}. \tag{3.32}$$

Wybierzmy nowe sterowanie $(\dot{\eta}_1, \dot{\eta}_2, \dot{\eta}_3)^T$ i zapiszmy ponownie powyższe równanie

$$\ddot{h} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\psi} \end{pmatrix} = K_{dd} \begin{pmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \\ \dot{\eta}_3 \end{pmatrix} + P = K_{dd}u + P \tag{3.33}$$

gdzie

$$K_{dd} = \begin{bmatrix} 0 & R\cos(\phi) & -R\cos\theta\sin\phi \\ -R & 0 & -R\sin\theta \\ 0 & 0 & 1 \end{bmatrix},$$

$$P = \begin{pmatrix} -R\left(\eta_1\left(\eta_3\cos\theta\cos\phi + \eta_2\sin\phi\right) - \eta_2\eta_3\sin\theta\sin\phi\right) \\ -R\eta_2\eta_3\cos\theta \\ 0 \end{pmatrix}.$$

Jeśli teraz za u weźmiemy

$$u = K_{dd}^{-1}(-P + v) \tag{3.34}$$

a za v

$$v = \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d), \tag{3.35}$$

otrzymamy równania na pochodne sterowań

$$u = \begin{pmatrix} \dot{\eta}_1 \\ \dot{\eta}_2 \\ \dot{\eta}_3 \end{pmatrix} = K_{dd}^{-1}(-P + \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d)). \tag{3.36}$$

W ten sposób rozszerzyliśmy wektor stanu układu o 3 zmienne, co sumarycznie możemy zapisać jako

$$\begin{pmatrix} \dot{q} \\ \dot{\eta} \end{pmatrix} = \begin{pmatrix} G\eta \\ K_{dd}^{-1}(-P + \ddot{h}_d - K_1(\dot{h} - \dot{h}_d) - K_2(h - h_d)) \end{pmatrix}. \tag{3.37}$$

Wyznacznik macierzy K_{dd} wynosi $R^2\cos(\phi(t))$. Podobnie jak w przypadku linearyzacji statycznej model jest sterowalny, gdy $\phi \neq \pm\frac{\pi}{2}$.

3.3.3 Sterowanie naiwne

Ponieważ efekty zastosowania linearyzacji statycznej oraz dynamicznej nie były zadowalające (zobacz podrozdziały 4.1.1 i 4.1.2), zaproponowany został dodatkowy model sterowania nazwany przez autora sterowaniem naiwnym a opisanym poniżej.

Załóżmy najpierw tymczasowo, że sterowanie kątami θ i ϕ, czyli η_1, η_2, nie wpływa na prędkość postępową robota $(\dot{x}, \dot{y})$ a prędkość wirowania η_3 jest stała. Wówczas model (3.28) przyjmuje postać

$$\begin{cases} \dot{x} = R\cos\theta\sin\phi\eta_3 \\ \dot{y} = -R\sin\theta\eta_3 \\ \dot{\phi} = \eta_1 \\ \dot{\theta} = \eta_2 \\ \dot{\psi} = \eta_3 \end{cases} . \tag{3.38}$$

Wyrażenia na $\dot{x}$, $\dot{y}$ można potraktować jako układ równań z dwiema niewiadomymi ϕ i θ, z następującym rozwiązaniem

$$\begin{cases} \theta = \arcsin\left(-\frac{\dot{y}}{R\eta_3}\right) = \theta(\dot{x},\dot{y}) \\ \phi = \arcsin\left(\frac{\dot{x}}{R\eta_3\cos\theta}\right) = \phi(\dot{x},\dot{y}) \end{cases} . \tag{3.39}$$

Równanie (3.39) pozwala na wyliczenie wartości kątów θ, ϕ przy zadanych prędkościach ruchu punku prowadzenia robota $(x, y)^T$. Teraz możemy zaproponować sterownik w postaci

$$\begin{cases} \eta_1 = k_2(\phi_d(k_1e_x, k_1e_y) - \phi) \\ \eta_2 = k_2(\theta_d(k_1e_x, k_1e_y) - \theta) \\ \eta_3 = const \end{cases} . \tag{3.40}$$

gdzie $e_x = x_d - x$, $e_y = y_d - y$, x_d, y_d zadana trajektoria a k_1, k_2 dodatnie wzmocnienia.

3.3.4 Filtracja

Jak pokazują badania symulacyjne (zobacz podrozdział 4.1.1), wyniki otrzymywane z użyciem linearyzacji statycznej nie są możliwe do zrealizowania na rzeczywistym robocie z powodu pojawiających się w nich dużych oscylacji. Prostą metodą rozwiązania tego problemu może być filtracja. Zadanie to można zrealizować na dwa sposoby.

Pierwszy z nich, filtracja „offline", wymaga wcześniejszego wyznaczenia oczekiwanych trajektorii ruchu układu, a następnie ich przefiltrowania (np. filtrem dolnoprzepustowym) i zadanie pochodnych w ten sposób uzyskanych sygnałów jako nowe sterowanie. Wydaje się jednak, że lepszym rozwiązaniem będzie rozszerzenie bezdryfowego układu sterowania opisującego zachowanie robota o dodatkowe zmienne pomocnicze, pozwalające na implementacje wbudowanego filtra dolnoprzepustowego według schematu

$$\dot{\phi} = \eta_1 \rightarrow \begin{array}{l} \dot{\phi}_{tmp} = \eta_1 \\ \dot{\phi} = \frac{K\phi_{tmp} - \phi}{T} \end{array} , \tag{3.41}$$

gdzie ϕ_{tmp} to zmienna tymczasowa, której sterowanie jest wyliczane zgodnie z algorytmem linearyzacji statycznej, natomiast rzeczywiste ϕ jest wynikiem działania filtra dolnoprzepustowego nałożonego na ϕ_{tmp}. Parametry K, T są parametrami filtra, odpowiednio: wzmocnieniem w pasmie przepustowym filtra i jego stałą czasową.

3.3.5 Algorytm JPTD

Obserwacja zachowania algorytmu linearyzacji statycznej dla modelu symultanicznego pozwoliła na zaproponowanie jeszcze jednego algorytmu, który w praktyce okazuje się być niebywale skuteczny. Co więcej, zaproponowany algorytm można łatwo zaimplementować dla modelu pełnego, gdzie także się sprawdził. Poniżej zostanie opisany sposób postępowania jaki zaprowadził do otrzymania wspomnianego sposobu sterowania.

W pierwszej kolejności należy zauważyć, że sterowanie kątami η_1, η_2 jest znacząco (nawet o rząd) mniejsze od prędkości obrotowej półsfery η_3. Co za tym idzie wpływ tych sterowań na prędkość robota $(\dot{x}, \dot{y})^T$ można w większości przypadków w praktyce pominąć. Przy takim założeniu uprościmy macierz G modelu (3.28) z postaci

$$\begin{bmatrix} 0 & R\cos\phi & -\cos\theta\sin\phi \\ -R & 0 & \sin\theta \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.42}$$

do postaci

$$\begin{bmatrix} 0 & 0 & -\cos\theta\sin\phi \\ 0 & 0 & \sin\theta \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.43}$$

Kolejno, podejmujemy próbę użycia tak uproszczonej macierzy G w algorytmach linearyzacji statycznej (podrozdział 2.1) i dynamicznej (podrozdział 2.2). Niestety dla pierwszego z nich nie da się wybrać żadnego sensownego wyjścia linearyzującego $h(q)$, aby wyznacznik macierzy odsprzęgania $\frac{\partial h}{\partial q}G$ nie był zerowy. Dla linearyzacji dynamicznej natomiast trywialne odwzorowanie $h(q)$, w takiej samej postaci jak użyte w podrozdziale 3.3.2, daje wyznacznik $\det K_{dd} = R^2{\eta_3}^2\cos^2(\theta)\cos(\phi)$. Wynika z tego, że aby sterowanie mogło działać półsfery muszą wirować ($\cos\theta, \cos\phi$ nie mogą być zerowe z powodu omawianych wcześniej fizycznych ograniczeń robota). Zaproponowany powyżej algorytm pozwolił zniwelować oscylacje, a przyjęte uproszczenia nie pogorszyły znacząco wyników śledzenia trajektorii, co pokazano w podrozdziale 4.1.5.

Rozdział 4

Badania symulacyjne

W rozdziale zostały zebrane reprezentacyjne wyniki symulacji przeprowadzonych w ramach pracy. Kolejno przedstawione zostaną rezultaty dla modelu symultanicznego, pełnego i uproszczonego będące efektem zastosowania algorytmów opisanych w rozdziale 2. Omówiony zostanie wpływ parametrów na otrzymywane wyniki i przedstawione wnioski dotyczące skuteczności i użyteczności poszczególnych algorytmów.

Wszystkie badania przedstawione poniżej przeprowadzono z użyciem środowiska Mathematica [4]. We wszystkich symulacjach przyjęto $l = 0.1$ oraz $R = 0.03$. Jako reprezentacyjną trajektorię zadaną $x_d(t)$, $y_d(t)$ przyjęto

$$\begin{cases} x_d(t) = \sin \frac{t}{2} \\ y_d(t) = \cos\left(t + \frac{\pi}{4}\right) \end{cases}, \tag{4.1}$$

która formuje krzywą Lissajous w postaci „ósemki”*. W przedstawionych poniżej wykresach jako błąd traktujemy różnicę między odwzorowaniem linearyzującym a jego wartością zadaną $e = h - h_d$, natomiast na wykresach śladu ruchu robota (umieszczonych zawsze po prawej stronie na dole rysunków z wykresami) czarną linią zaznaczono zadaną trajektorię a na czerwono rzeczywistą pozycję robota $(x, y)^T$.

4.1 Model symultaniczny

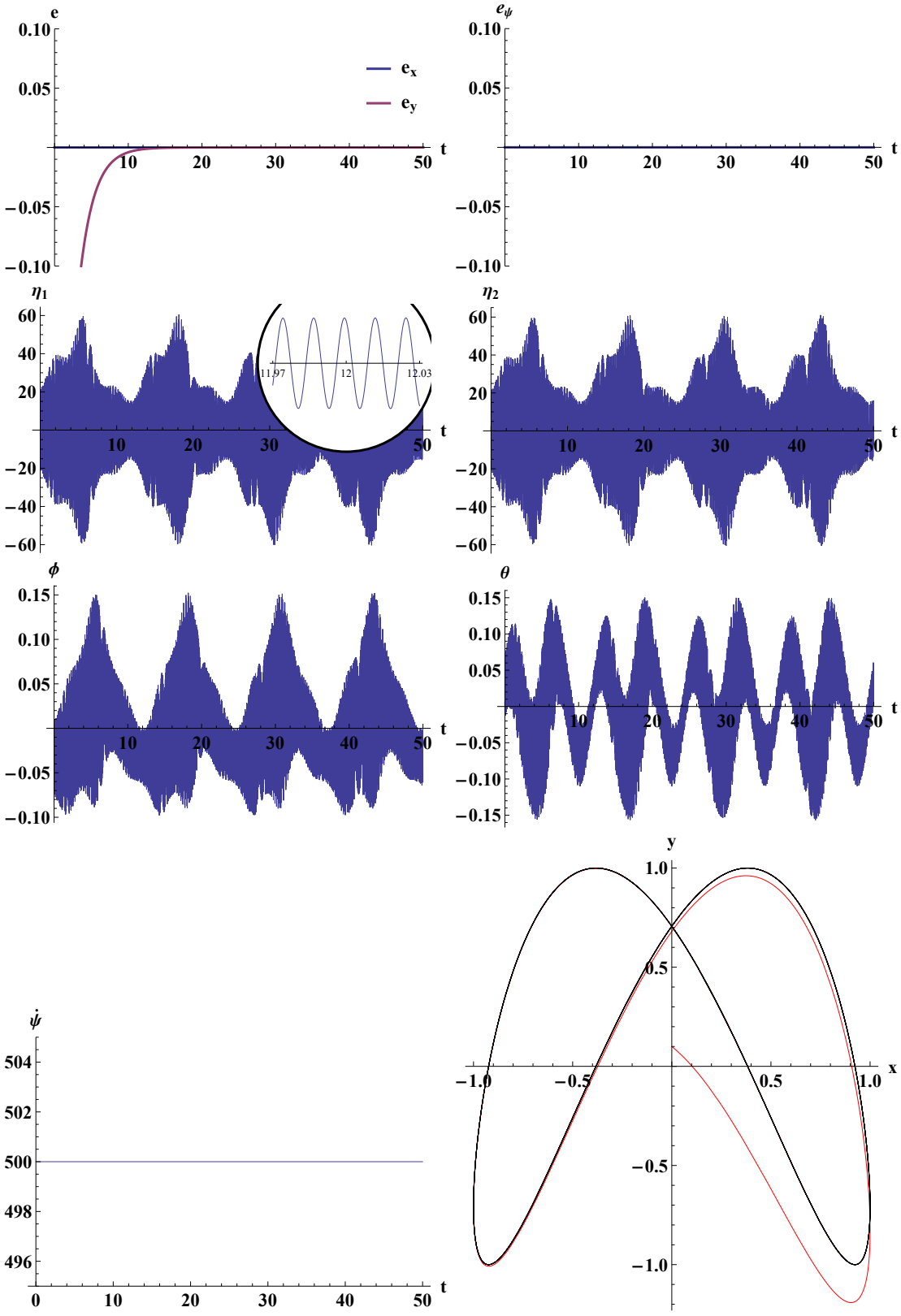
Na początku algorytmy przetestowano na stosunkowo prostym modelu symultanicznym (3.28). Zaznaczmy, że jeśli na przedstawionych dalej wykresach nie pokazano całego przebiegu błędu e_y to jego maksimum zawsze występowało w stanie początkowym i wynosiło $\frac{\sqrt{2}}{2} - 0.1$.

4.1.1 Linearyzacja statyczna

W pierwszej kolejności zbadany został algorytm linearyzacji statycznej zaprezentowaną w podrozdziale 3.3.1. Przykładowe rezultaty działania algorytmu pokazano na rysunku 4.1. Uzyskano je przy macierzy wzmocnień $K = 0.5I_3$ zadanej wartości wirowania $\psi_d = 500t$ i warunkach początkowych $q(0) = (0, 0.1, 0, 0, 0, 0)^T$†. Możemy zaobserwować, że błędy spadają do zera i robot śledzi zadaną trajektorię. Niestety, jak widać, w układzie pojawiają się niepożądane oscylacje o dużej amplitudzie i częstotliwości. Badania

*Badania przeprowadzono również z użyciem innych trajektorii takich jak: odcinek, okrąg i kwadrat.

†jako wartość początkową współrzędnej y przyjęto 0.1, aby środek robota znajdował się w punkcie $(0, 0)^T$



Rysunek 4.1: Model symultaniczny – linearyzacja statyczna

wykazały, że częstotliwość tych oscylacji jest ściśle związana z zadaną prędkością wirowania, a mianowicie $f = \frac{\dot{\psi}_d}{2\pi}$. Zwiększanie wartości w macierzy wzmocnień przyśpiesza spadanie błędów, ale jednocześnie zwiększa amplitudy sterowań i kątów. Zwiększenie zadanej wartości wirowania zmniejsza wychylenia kątów, ale zgodnie z tym co wspomniano wcześniej, zwiększa częstotliwość oscylacji. Dla innych trajektorii zadanych otrzymywano analogiczne wyniki. Mimo iż główny cel zadania sterowania został osiągnięty (śledzenie trajektorii), to nie sposób uznać tego za sukces. Z punktu widzenia możliwości sterowania rzeczywistym robotem niewykonalne jest, by zmieniał on kąty wychyleń z taką dużą częstotliwością, eliminuje to całkowicie możliwość zaimplementowania linearyzacji statycznej w praktyce.

4.1.2 Linearyzacja dynamiczna

Kolejno przebadano algorytm linearyzacji dynamicznej opisany w podrozdziale 3.3.2. Przyjmując macierze wzmocnień $K_1 = I_3$ oraz $K_2 = 2I_3$, zadaną wartość $\psi_d = 200$ i warunki początkowe $q(0) = (0, 0.1, 0, 0, 0, 0, 0, 0, 200)^T$ otrzymano wyniki przedstawione na rysunku 4.2. Jak można zaobserwować błędy oscylując spadają do zera. Znaczący jest fakt, że oscylacje na sterowaniu znacząco zmniejszyły swoją amplitudę (związek ich częstotliwości z zadaną prędkością wirowania jest taki sam jak w przypadku linearyzacji statycznej), natomiast oscylacje kątów spadły prawie do zera. Z perspektywy sterowania rzeczywistym robotem jest to bardzo dobra wiadomość. Zwiększanie wartości w macierzy wzmocnień oczywiście prowadzi do szybszego zbiegania błędów do zera, jednak powoduje uwidocznienie się oscylacji w kątach wychyleń. Zwiększanie zadanej prędkości wirowania ψ_d oprócz opisanego powyżej wpływu na częstotliwość niepożądanych oscylacji może prowadzić do pojawiania się ich na kątach wychyleń, zmniejsza także amplitudę sterowań i wychyleń. Podobne zachowania jak dla „ósemki" powtarzają się dla innych trajektorii zadanych.

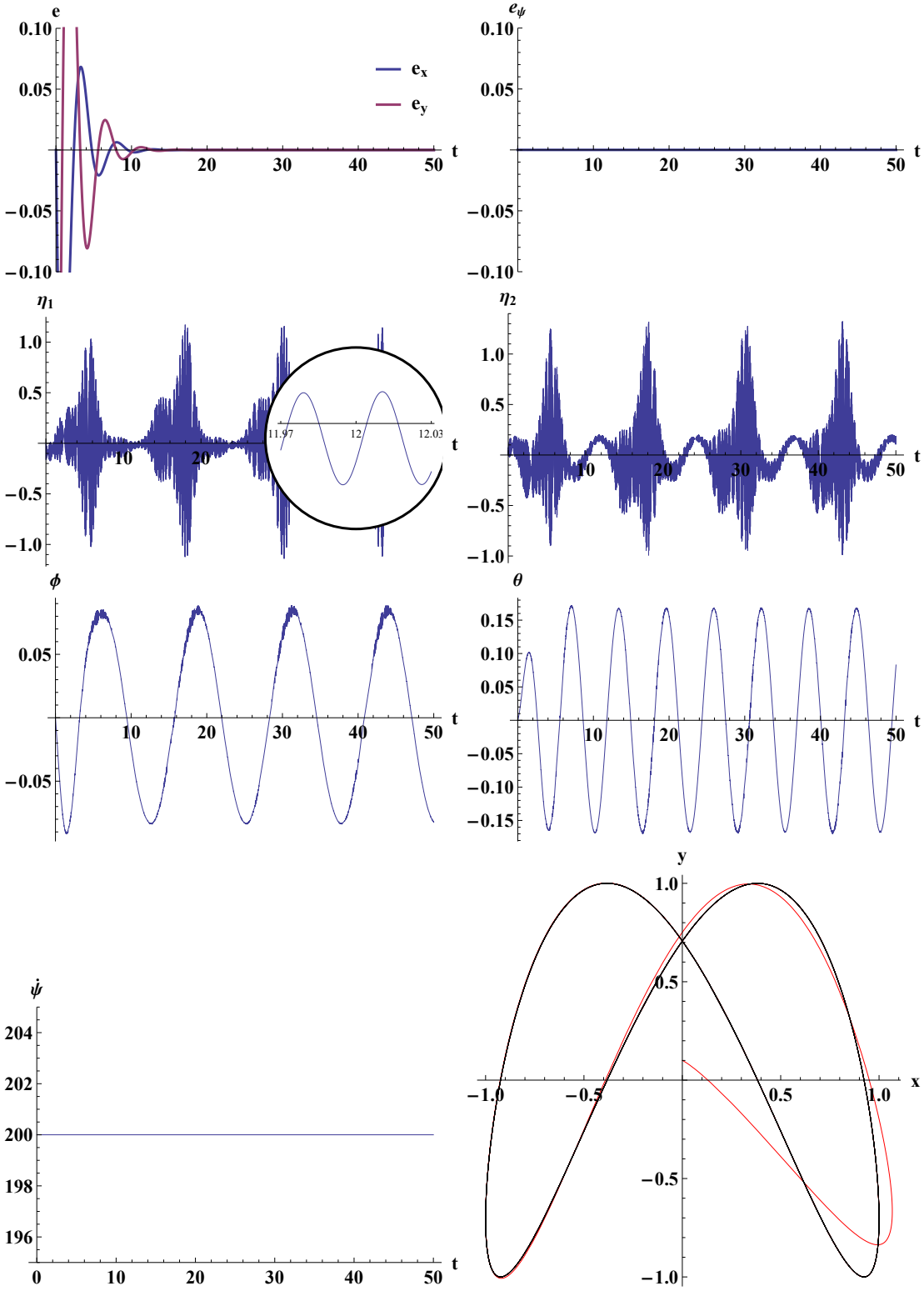
Algorytm linearyzacji dynamicznej przyniósł znaczącą poprawę względem linearyzacji statycznej i daje nadzieje na implementację na rzeczywistym robocie. Należy jednak uważać przy doborze wzmocnień i zadanej prędkości wirowania, niewłaściwe ich wartości mogą spowodować nasilone oscylacje na kątach ϕ i θ oraz ostatecznie uniemożliwić sterowanie robotem.

4.1.3 Sterowanie naiwne

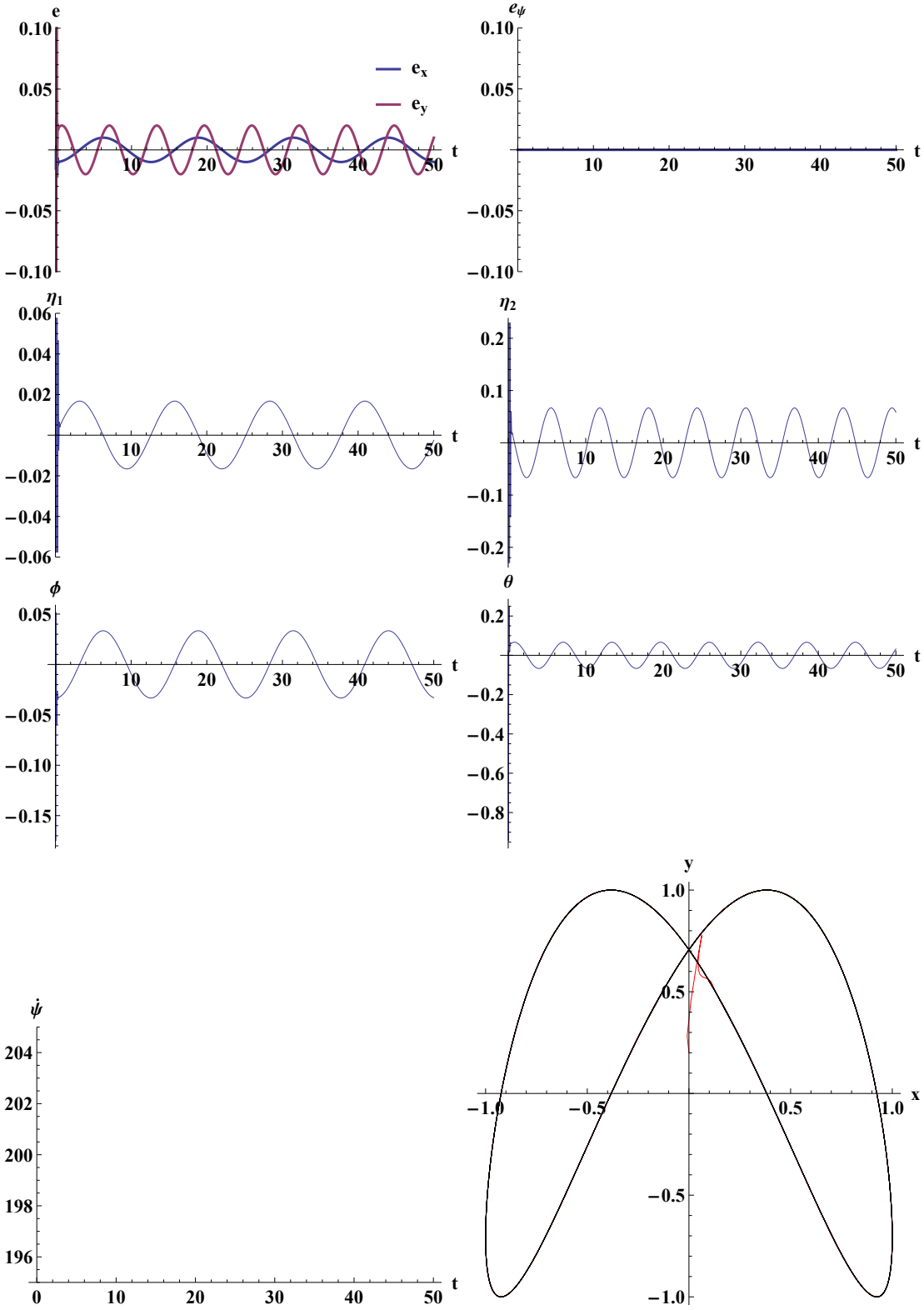
Następnym zbadanym algorytmem było sterowanie naiwne opisane w podrozdziale 3.3.3. Przykładowe wyniki dla wzmocnień $k_1 = 50$, $k_2 = 40$, stałego wirowania $\eta_3 = 200$ i warunków początkowych $q(0) = (0, 0.2, 0, 0, 0, 0)^T$ pokazano na rysunku 4.3. Obserwujemy, że błędy e_x, e_y nie zanikają a oscylują wokół zera. Dobrą wiadomością jest natomiast to, że w tym przypadku nie ma oscylacji ani na sterowaniach, ani na samych wychyleniach. Zwiększając wzmocnienia zmniejszamy maksymalny błąd w stanie ustalonym. Błędy, mimo iż nie spadają do zera są akceptowalne, ich wartość jest stosunkowo mała w porównaniu z przebytą przez robota drogą.

4.1.4 Filtracja

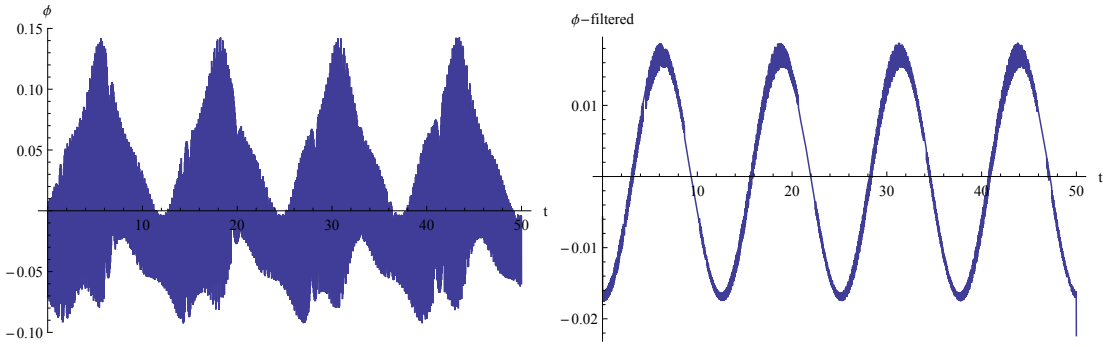
W trajektoriach otrzymywanych przy sterowaniu z użyciem linearyzacji statycznej pojawiały się niepożądane oscylacje. W ramach badań podjęto próbę usunięcia ich poprzez



Rysunek 4.2: Model symultaniczny – linearyzacja dynamiczna



Rysunek 4.3: Model symultaniczny – sterowanie naiwne



Rysunek 4.4: Efekt zastosowania filtra dolnoprzepustowego do wyników linearyzacji statycznej

filtrację zgodnie z zasadą opisaną w podrozdziale 3.3.4. Przykładowe wyniki filtracji „offline" na otrzymanej trajektorii pokazano na rysunku 4.4. Uzyskany w ten sposób przefiltrowany sygnał zdaje się odpowiadać intuicji, mimo to zastosowanie jego pochodnej jako sterowania nie sprawdza się – robot szybko zjeżdża z zadanej trajektorii. Pomysł z rozszerzeniem układu o filtr dolnoprzepustowy i filtracją na bieżąco również nie przyniósł oczekiwanych rezultatów.

4.1.5 Algorytm JPTD

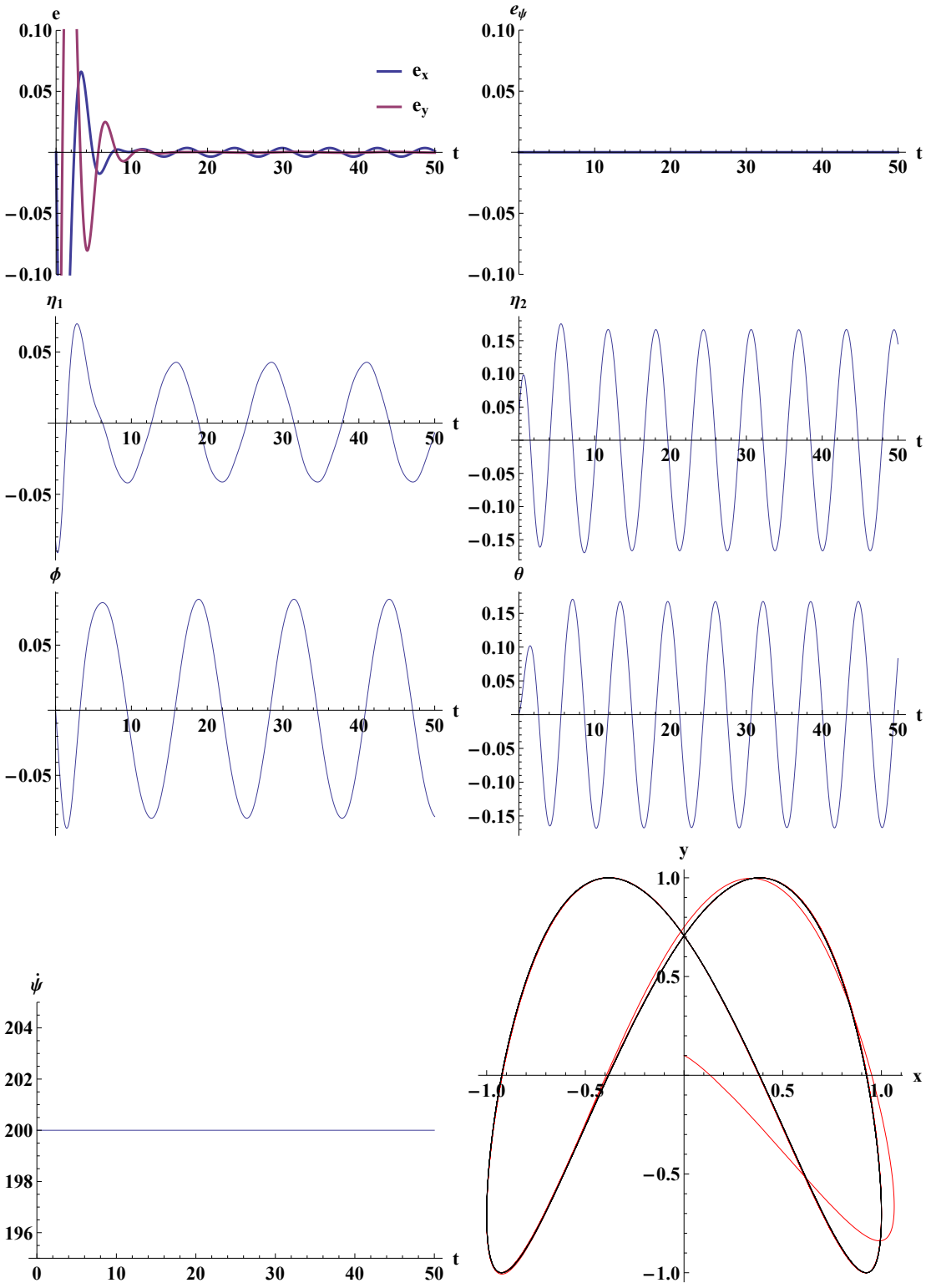
Tutaj zilustrujemy działanie algorytmu JPTD, opisanego w podrozdziale 3.3.5. Przyjmując macierze wzmocnień $K_1 = I_3$, $K_2 = 2I_3$, zadane wirowanie $\psi_d = 200t$ i stan początkowy $q(0) = (0, 0.2, 0, 0, 0, 0, 200)^T$ otrzymujemy wyniki przedstawione na rysunku 4.5. Algorytm JPTD łączy w sobie cechy linearyzacji dynamicznej oraz sterowania naiwnego. Prędkość spadania błędów jest taka samo jak w przypadku linearyzacji, jednak podobnie jak w przypadku sterowania naiwnego błędy nie spadają do zera, nie pojawiają się również żadne niepożądane oscylacje. Zwiększenie wartości w macierzach wzmocnień oczywiście przyśpiesza zbieżność i zmniejsza końcowy błąd. Otrzymane wyniki pozwalają wierzyć, że algorytm JPTD można zaimplementować na rzeczywistym robocie.

4.2 Model pełny

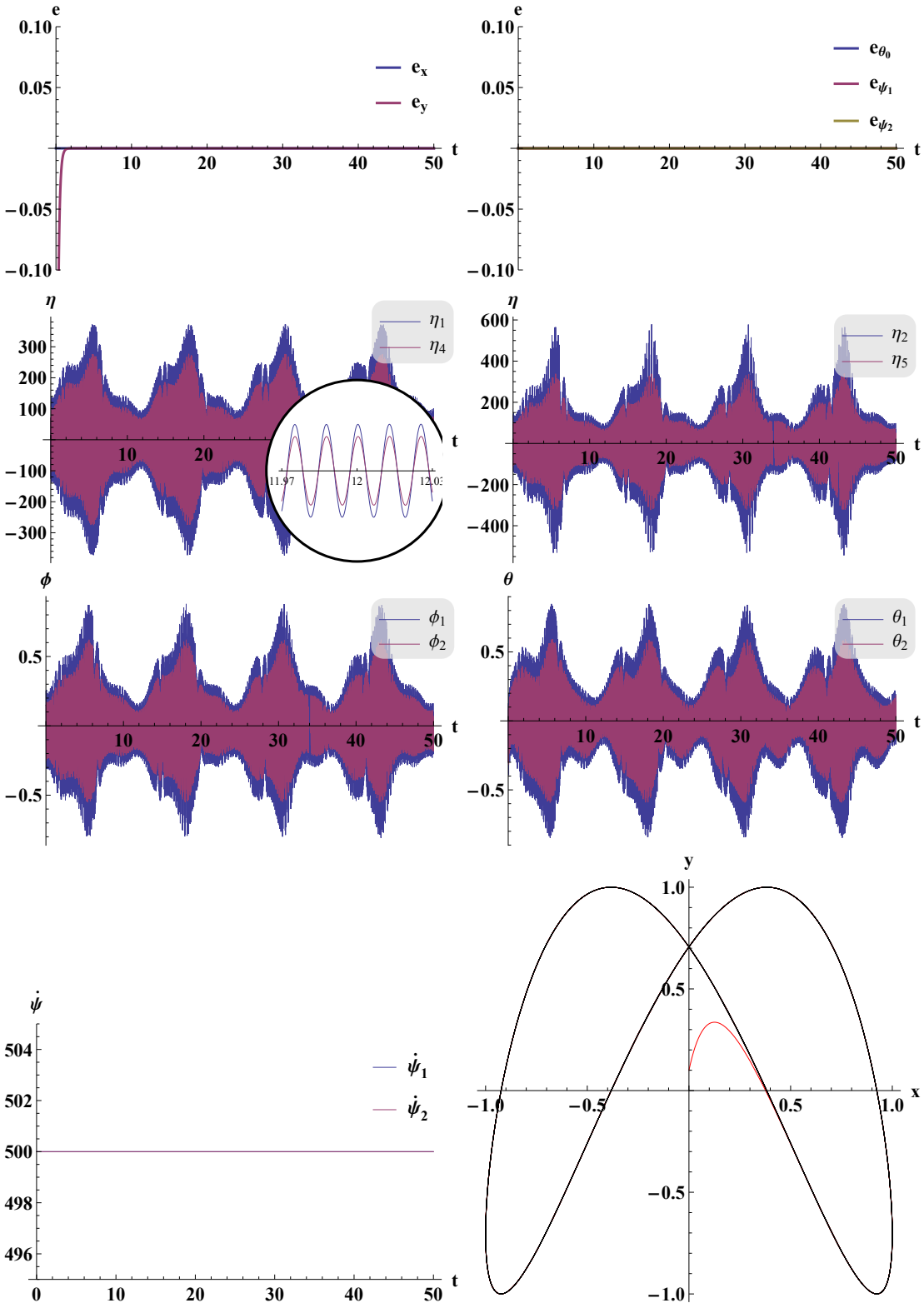
Zaprezentujemy teraz wyniki badań algorytmów przedstawionych w podrozdziale 3.1. Podobnie jak w poprzednim podrozdziale, jeśli na przedstawionych dalej wykresach nie przedstawiono całego przebiegu błędu e_y to maksimum zawsze pojawiało się w stanie początkowym i wynosiło $\frac{\sqrt{2}}{2} - 0.1$.

4.2.1 Linearyzacja statyczna

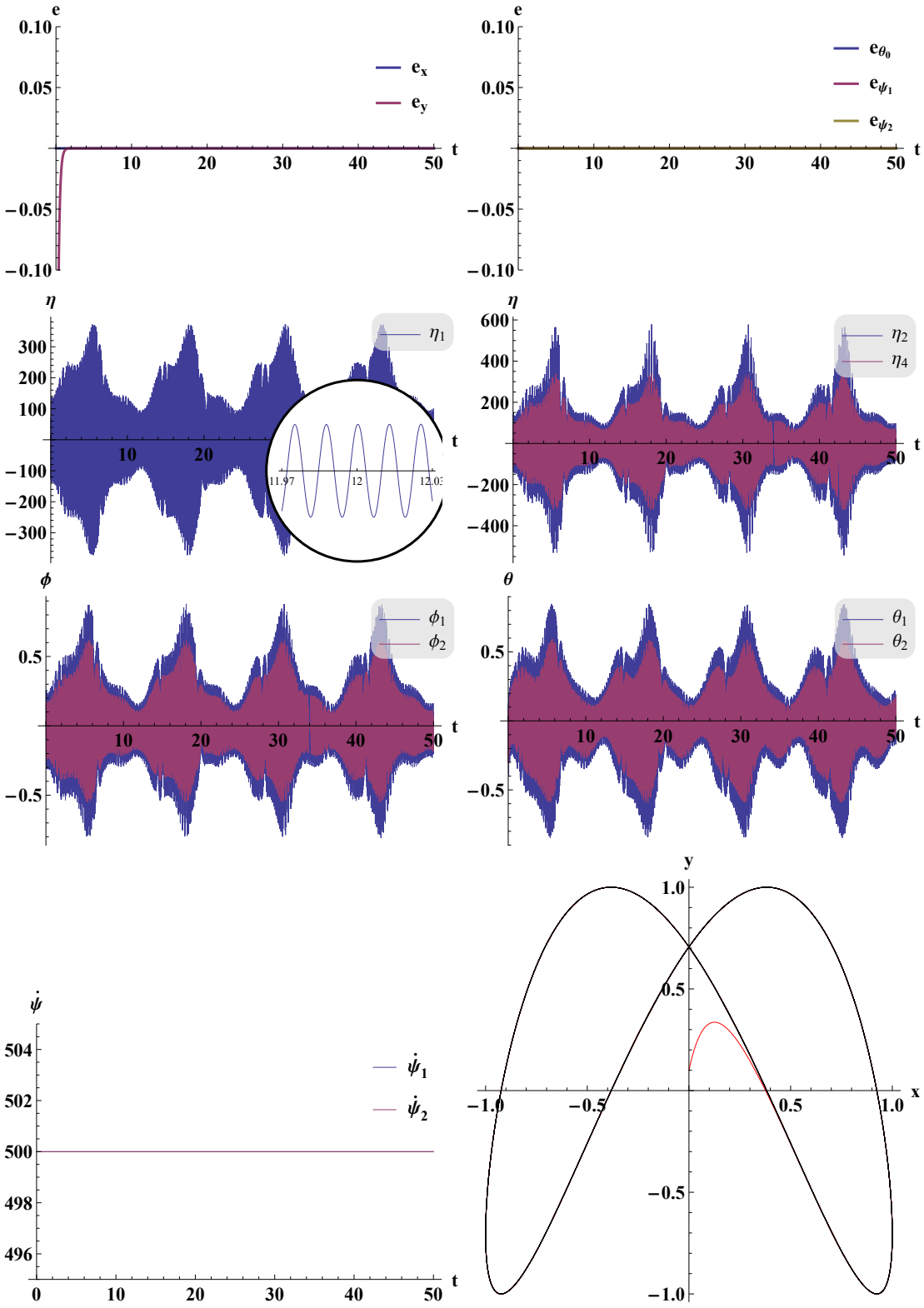
Na początku zbadany został algorytm linearyzacji statycznej. Przyjmując macierz wzmocnień K jako $K = 5I_5$, wartość zadaną wirowań $\psi_{1d} = 500t$, $\psi_{2d} = 500t$, zadaną orientację $\theta_{0d} = 0$ oraz warunki początkowe $q(0) = (0, 0.1, 0, 0.25, -0.3, 0, 0.15, -0.3, 0)^T$ otrzymano wyniki przedstawione na rysunkach 4.6 (dla modelu (3.1)) oraz 4.7 (dla modelu (3.2)). Podobnie jak przy modelu symultanicznym obserwujemy pojawienie się niepożądanych oscylacji o częstotliwości zależnej głównie od zadanej prędkości wirowania, jednak



Rysunek 4.5: Model symultaniczny – algorytm JPTD



Rysunek 4.6: Model pełny (3.1) – linearyzacja statyczna



Rysunek 4.7: Model pełny (3.2) – linearyzacja statyczna

tym razem mają one większą amplitudę. Zmiana parametrów również powoduje podobną reakcję. Zwiększenie wartości w macierzy wzmocnień przyśpiesza zbieganie się błędów a zwiększenie zadanych prędkości wirowania zmniejsza amplitudę wychyleń ale zwiększa częstotliwość oscylacji. Jedyną różnicą względem modelu symultanicznego jest fakt, że zwiększając zadane prędkości wirowania zwiększamy również wartości sterowań. Algorytm przetestowano także na innych trajektoriach z podobnymi rezultatami. Sprawdzono również zachowanie robota, gdy zażądamy, aby orientacja robota była prostopadła do trajektorii tzn. $\theta_{0d} = \arctan2(\dot{x}_d, \dot{y}_d)$ otrzymując ponownie zbliżone rezultaty. Ciekawą sytuację zaobserwowano, gdy ustawiono $\theta_{0d} = 0.5$ (zobacz rysunek 4.8). Jak widać po pewnym czasie występujące niepożądane oscylacje zaczęły spadać i ostatecznie całkowicie zanikły.

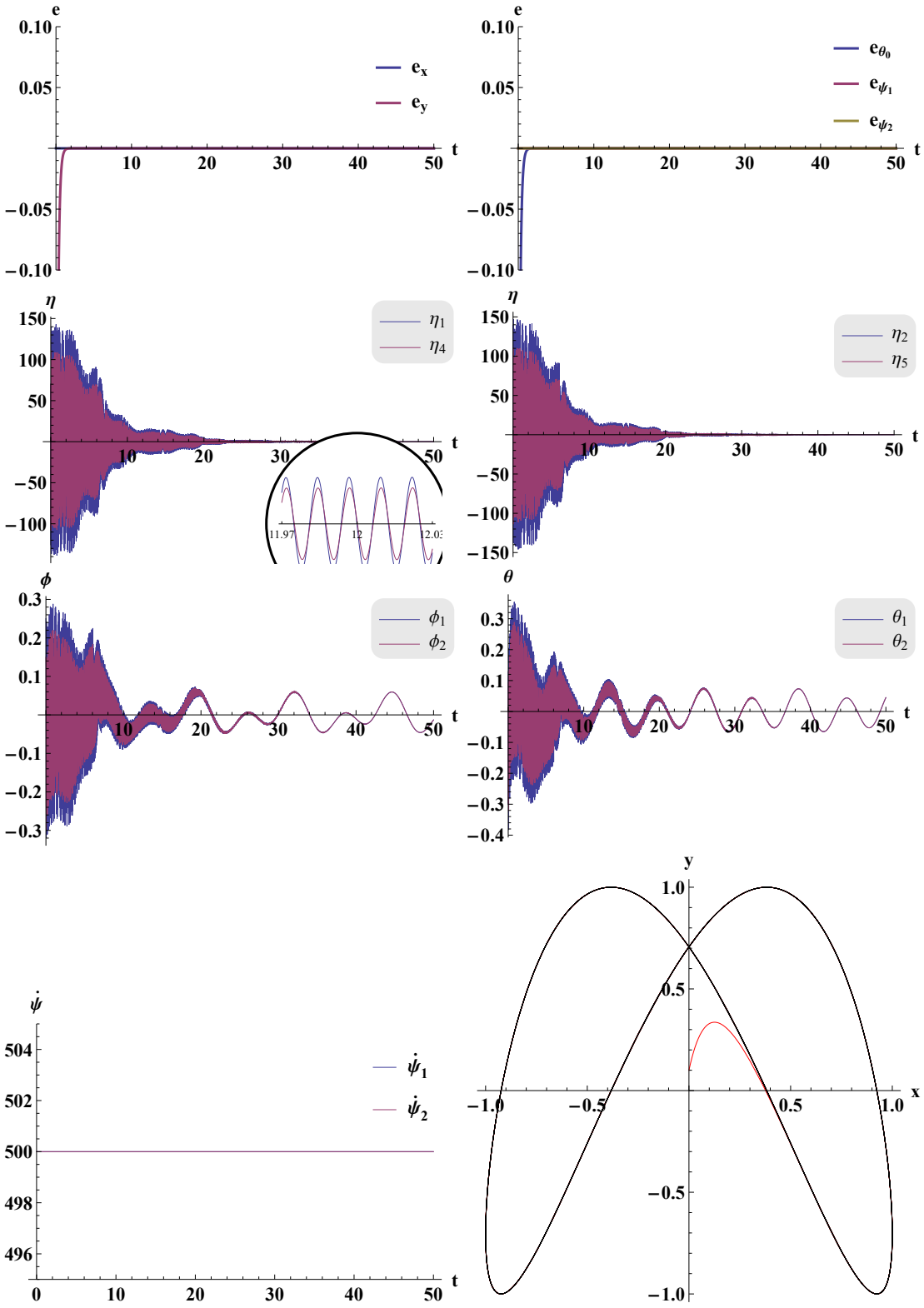
Należy tutaj wspomnieć, że z powodu ograniczeń wynikających z wartości wyznacznika macierzy odsprzęgania oraz nieprzewidywalnych wartości kątów wychyleń niejednokrotnie nie udawało się otrzymać wyników symulacji. Mimo iż w jednym przypadku udało się wygasić oscylacje, to z oczywistych względów algorytm ten nie jest możliwy do zaimplementowania na rzeczywistym robocie.

4.2.2 Linearyzacja dynamiczna

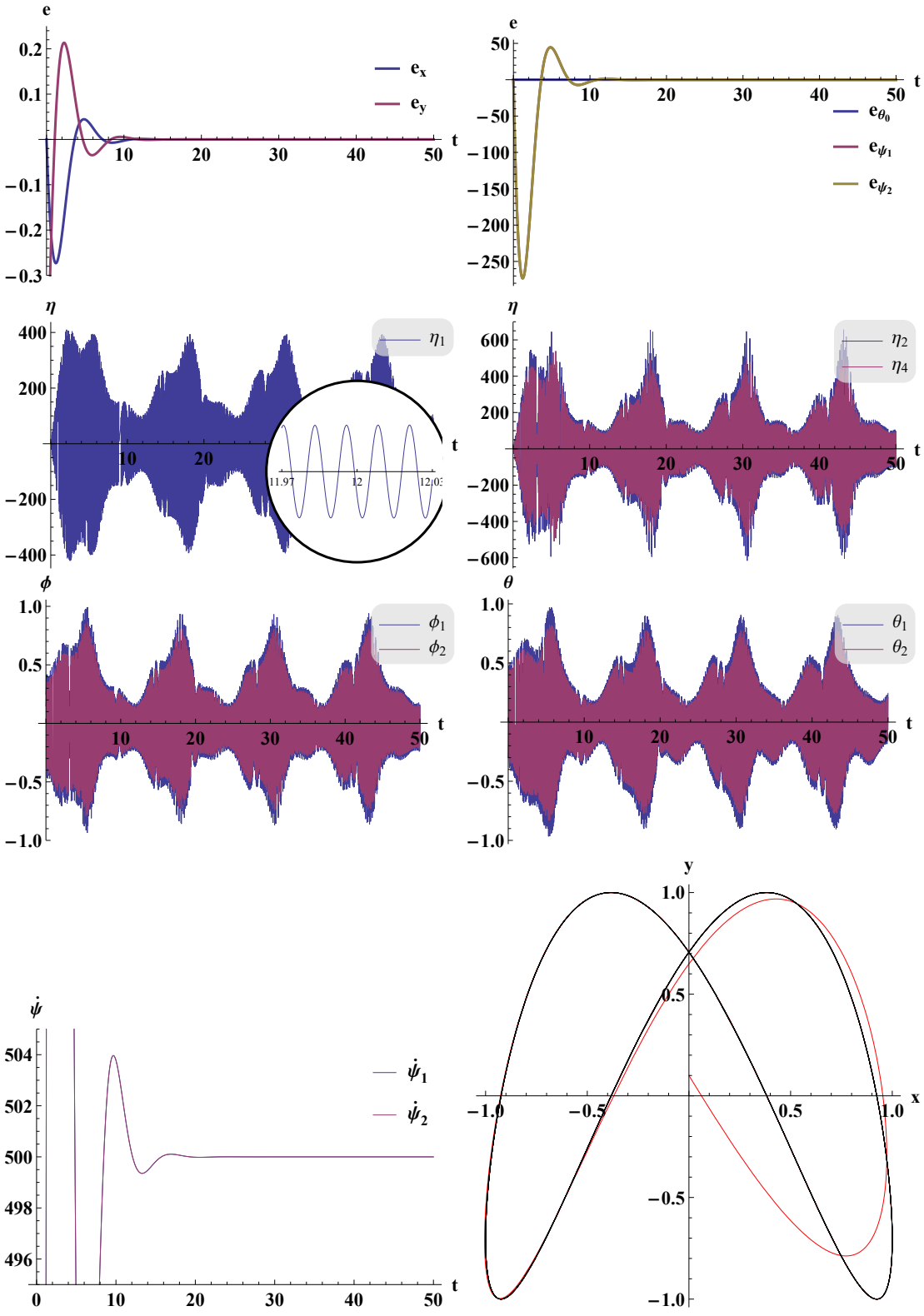
Kolejnym przetestowanym algorytmem była linearyzacja dynamiczna. Przykładowe wyniki przedstawiono na rysunku 4.9. Otrzymano je przyjmując macierze wzmocnień $K_1 = 5I_5$ oraz $K_2 = 2I_5$, $\psi_{1d} = 500t$, $\psi_{2d} = 500t$, zadaną orientację $\theta_{0d} = 0$ oraz warunki początkowe $q(0) = (0, 0.1, 0, 0.25, -0.3, 0, 0.15, -0.3, 0, 0, 0, 0, 0, 0)^T$. Ponownie otrzymaliśmy oscylacje o częstotliwości 80Hz (zależnej od ψ_d) i dużej amplitudzie. Inaczej niż w przypadku modelu symultanicznego linearyzacja dynamiczna nie przyniosła znaczącej poprawy względem statycznej. Nie udało dobrać się takich parametrów i zadanych wartości aby zmniejszyć oscylacje na kątach wychyleń. Ponownie zwiększanie wartości w macierzach wzmocnień przyśpiesza zbieganie błędów a zwiększenie zadanych prędkości wirowania zmniejsza wychylenia i zwiększa sterowania. Algorytm przetestowano na innych trajektoriach z powtarzalnymi wynikami. Identycznie jak w przypadku linearyzacji statycznej dla niektórych trajektorii i niezerowej zadanej orientacji θ_{0d} zdarzało się iż oscylacje wygasały po czasie. W wielu przypadkach w ogóle nie dało się otrzymać wyników (z powodu ograniczeń wyznacznika macierzy K_{dd} i nieprzewidywalności wychyleń). Także i w tym przypadku implementacja na robocie rzeczywistym jest niemożliwa.

4.2.3 Algorytm JPTD

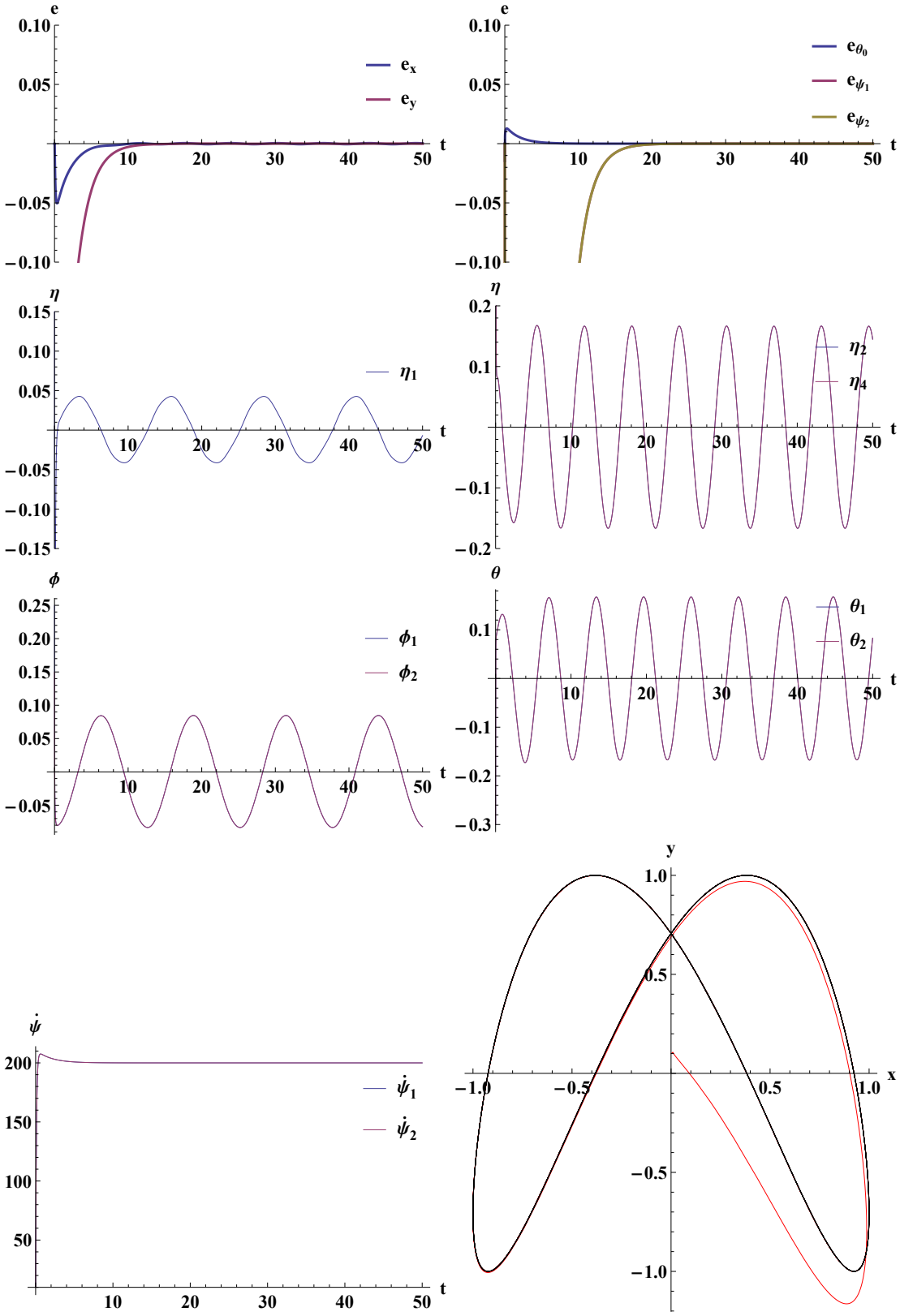
Ostatnim zbadanym algorytmem dla modelu pełnego był algorytm JPTD. Wybierając macierze wzmocnień $K_1 = 10I_5$, $K_2 = 5I_5$, zadane prędkości wirowań $\psi_{1d} = 200$, $\psi_{1d} = 200$ oraz warunki początkowe $q(0) = (0, 0.1, 0, 0.25, -0.3, 0, 0.15, -0.3, 10, 10)^T$ otrzymano wyniki przedstawione na rysunku 4.10. Zgodnie z oczekiwaniami w wynikach nie pojawiają się niepożądane oscylacje jednak błąd nie dąży do zera. Mimo to jego wielkość jest pomijalna (mniej niż 1 centymetr). Zwiększając wartości w macierzach wzmocnień możemy dodatkowo zmniejszyć końcowy błąd. Zwróćmy uwagę iż algorytm JPTD zachowuje się bardzo podobnie do algorytmu linearyzacji dynamicznej z pominięciem oscylacji. Algorytm przetestowano dla różnych zadanych trajektorii, wirowań i orientacji otrzymując zbliżone wyniki. Zaprezentowane powyżej wyniki pozwalają sądzić, że algorytm JPTD ma zastosowanie praktyczne. Niestety z powodu ograniczeń wyznacznika macierzy K_{dd} algorytm ten nie pozwala dojechać do punktu i się w nim zatrzymać.



Rysunek 4.8: Model pełny (3.1) – linearyzacja statyczna i zanikanie oscylacji



Rysunek 4.9: Model pełny (3.2) – linearyzacja dynamiczna



Rysunek 4.10: Model pełny (3.2) – algorytm JPTD

4.3 Model uproszczony

Teraz pokażemy wyniki badań przenoszenia sterowań z modelu uproszczonego do modelu pełnego. W zaprezentowanych poniżej wynikach jako błąd śledzenia trajektorii zadanej pokażemy tylko dwie pierwsze składowe odwzorowania linearyzującego, gdyż tylko one mają sens w modelu pełnym. Zaznaczmy, że jeśli na przedstawionych poniżej wykresach nie widać całego przebiegu błędu e_y to maksimum zawsze było w stanie początkowym.

4.3.1 Linearyzacja statyczna

Jako pierwszy zbadano algorytm linearyzacji statycznej. Długość dyszla zawsze przyjmowano jako $d = 0.15$. Przykładowe wyniki dla macierzy wzmocnień $K = 0.5I_5$, zadanego kąta $\theta_{u2d} = 0.4$, zerowego sterowania promieniami, warunków początkowych $q(0) = (0, 0.1, 0, 0.4, 0, 0.4, 0, 0.02, 0.02)^T$ i przeniesienia sterowań w trybie „offline" zaprezentowane na rysunkach 4.11 (dla modelu (3.1)) oraz 4.12 (dla modelu (3.2)). Możemy zaobserwować, że przeniesienie do modelu (3.1) działa bardzo dobrze, natomiast przy przenoszeniu do modelu (3.2) trajektoria robota szybko rozjeżdża się z zadaną a ponieważ jest to tryb „offline" nie ma w nim żadnych mechanizmów korygujących tę sytuację. Widzimy, że w tym przypadku błędy nie zbiegają do zera, dzieje się tak ponieważ model uproszczony nie oddaje wszystkich cech modelu pełnego. Zwiększanie wartości w macierzy wzmocnień przyśpiesza spadanie błędu ale nie ma wpływu na końcową ich wartość.

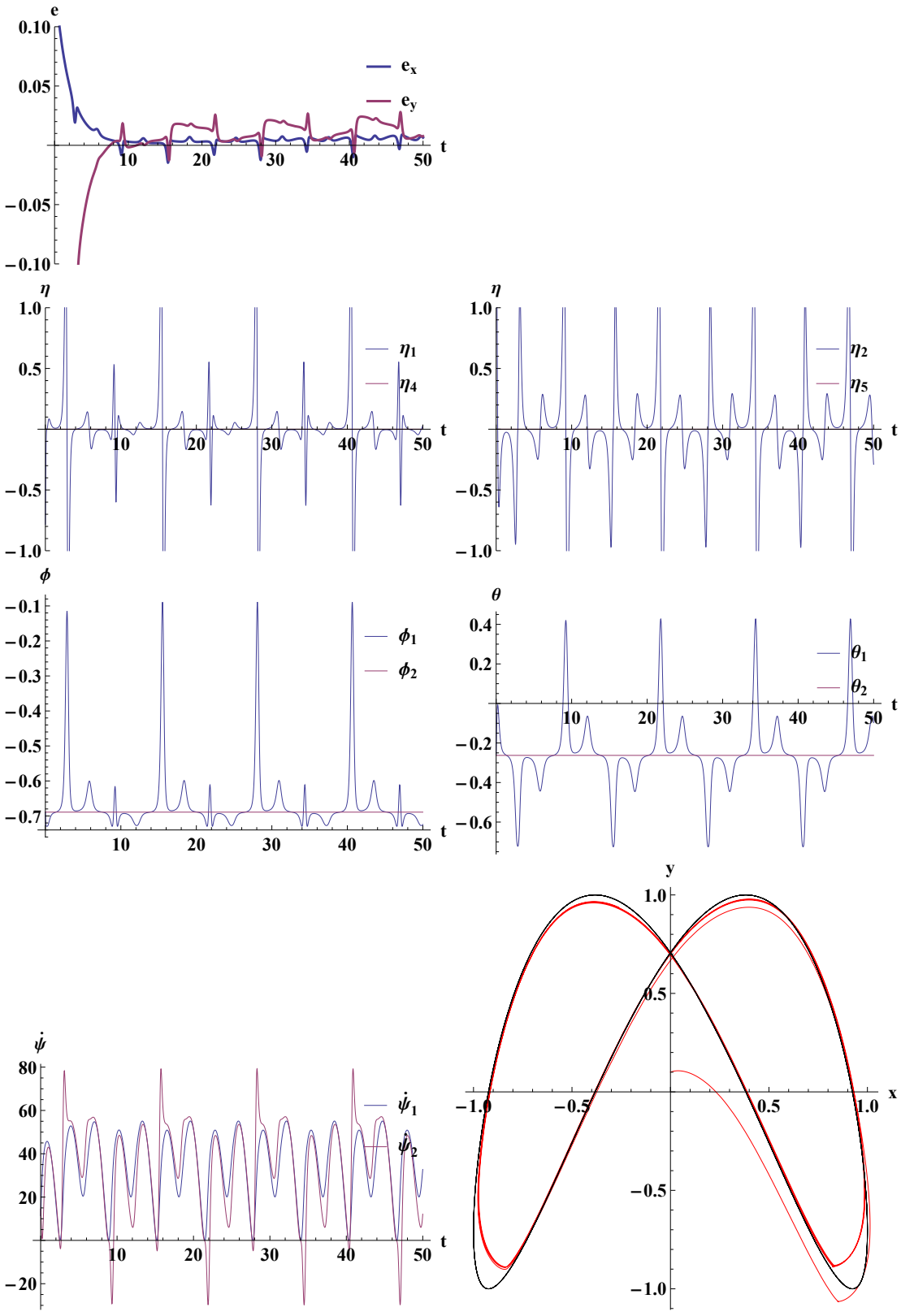
Dla tych samych parametrów zbadano również przeniesienie sterowań w trybie „online" a wyniki przedstawiono na wykresach 4.13 i 4.14. Ponownie lepiej sprawdza się model (3.1). Model (3.2) zachowuje się jednak lepiej w porównaniu z trybem „offline", gdyż pozostaje na trasie (mimo to w zakręcie kąt ϕ_2 przekracza dozwolone wartości). W trybie „online" zwiększenie wartości w macierzy wzmocnień nie tylko przyśpiesza zbieganie błędu ale także zmniejsza ich wartości końcowe.

4.3.2 Linearyzacja dynamiczna

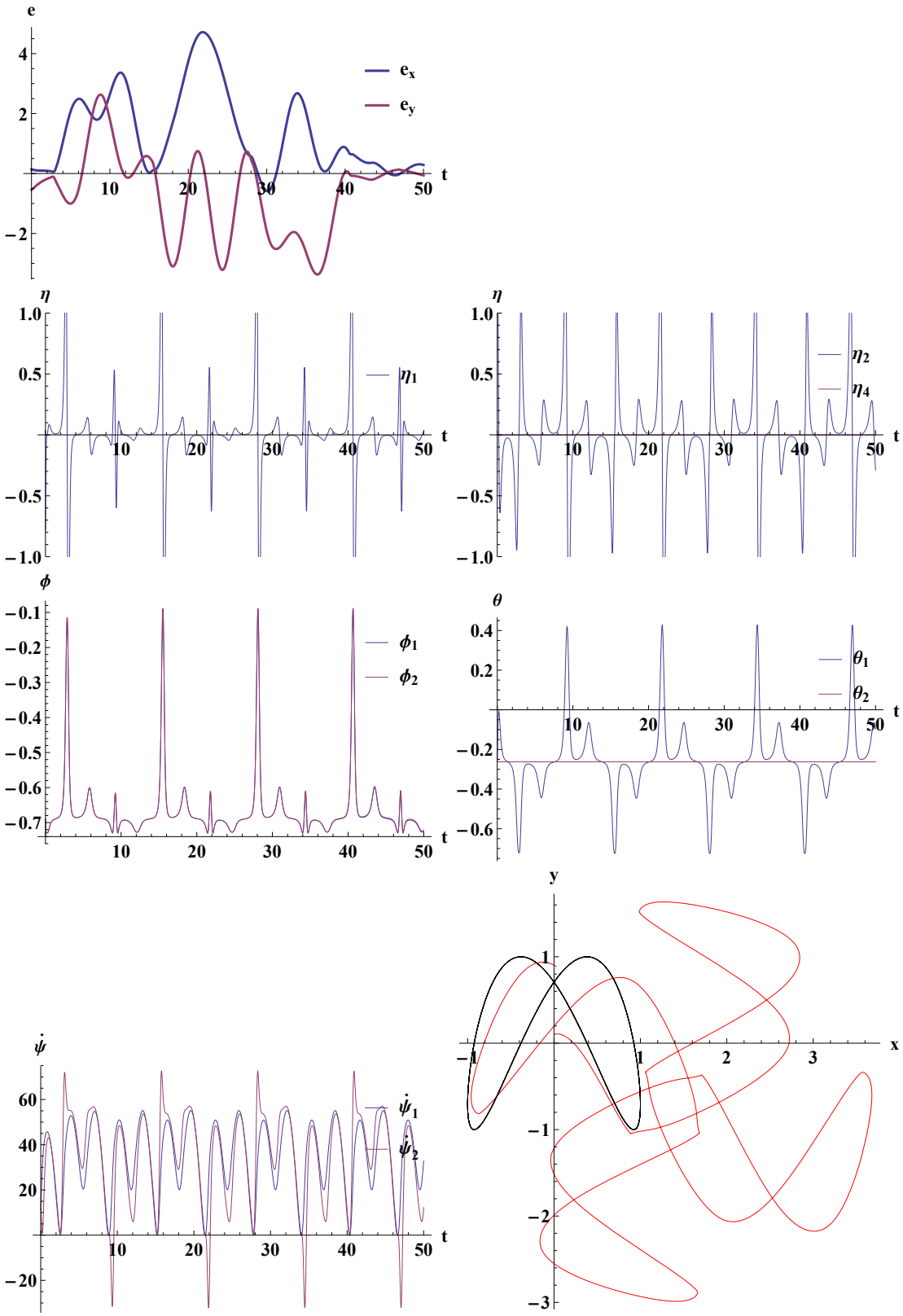
Następnie przetestowano algorytm linearyzacji dynamicznej. Przykładowe wynik przedstawiono na rysunkach 4.15 i 4.16. Uzyskano je przyjmując macierze wzmocnień $K_1 = I_5$, $K_2 = 0.5I_5$, zadany kąt $\theta_{u2d} = 0.4$, zerowe sterowania promieniami, warunki początkowe w postaci $q(0) = (0, 0.1, 0, 0.4, 0, 0.4, 0, 0.02, 0.02)^T$ i stosując przeniesienie sterowań w trybie „offline". Podobnie jak w przypadku linearyzacji statycznej przeniesienie dla modelu (3.1) działa lepiej niż dla modelu (3.2). Ponownie widzimy niegasnące do zera błędy wynikające z różnic między modelem uproszczonym i pełnym. Zwiększanie wartości w macierzy wzmocnień nieznacznie przyśpiesza zbieganie błędów, powoduje jednak wzmacnianie oscylacji, dodatkowo sprawia, że błąd końcowy się zwiększa.

4.3.3 Śledzenie ścieżki

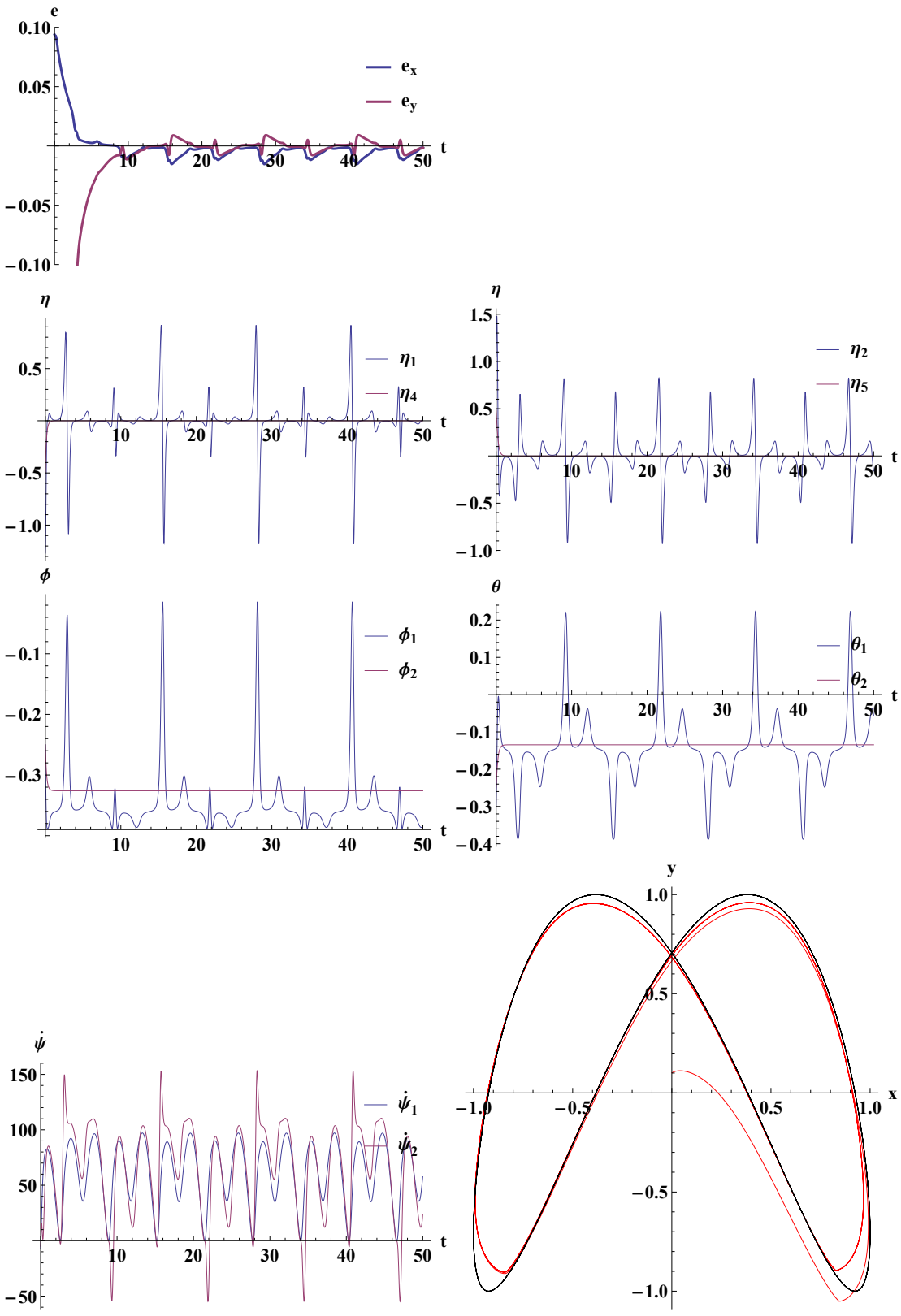
Ostatnim przetestowanym algorytmem jest algorytm Samsona. W celu zbadania tego algorytmu w pierwszym kroku „stworzono" zadaną ścieżkę poprzez parametryzację odległością krzywoliniową „ósemki". Następnie przyjmując $k_{\theta_{u1}} = 1$, $k_{p_y} = k_{v_y} = k_{p_\theta} = k_{v_\theta} = 500$, zadany kąt $\theta = \frac{\pi}{2} - 0.4$, prędkość $v = 0.2$, warunki początkowe $q(0) = (x(0), y(0), \theta_0(0), \theta_{u1}(0), \phi_{u1}(0), \theta_{u2}(0), \phi_{u2}(0)) = (0, 0, -\frac{\pi}{2}, \frac{\pi}{2} - 0.4, 0, \frac{\pi}{2} - 0.4, 0)^T$ oraz przeniesienie w trybie „offline" otrzymano wyniki przedstawione na rysunkach 4.17 i 4.18. Jak zwykle przeniesienie dla modelu (3.1) działa lepiej niż dla (3.2). Możemy zaobserwować,



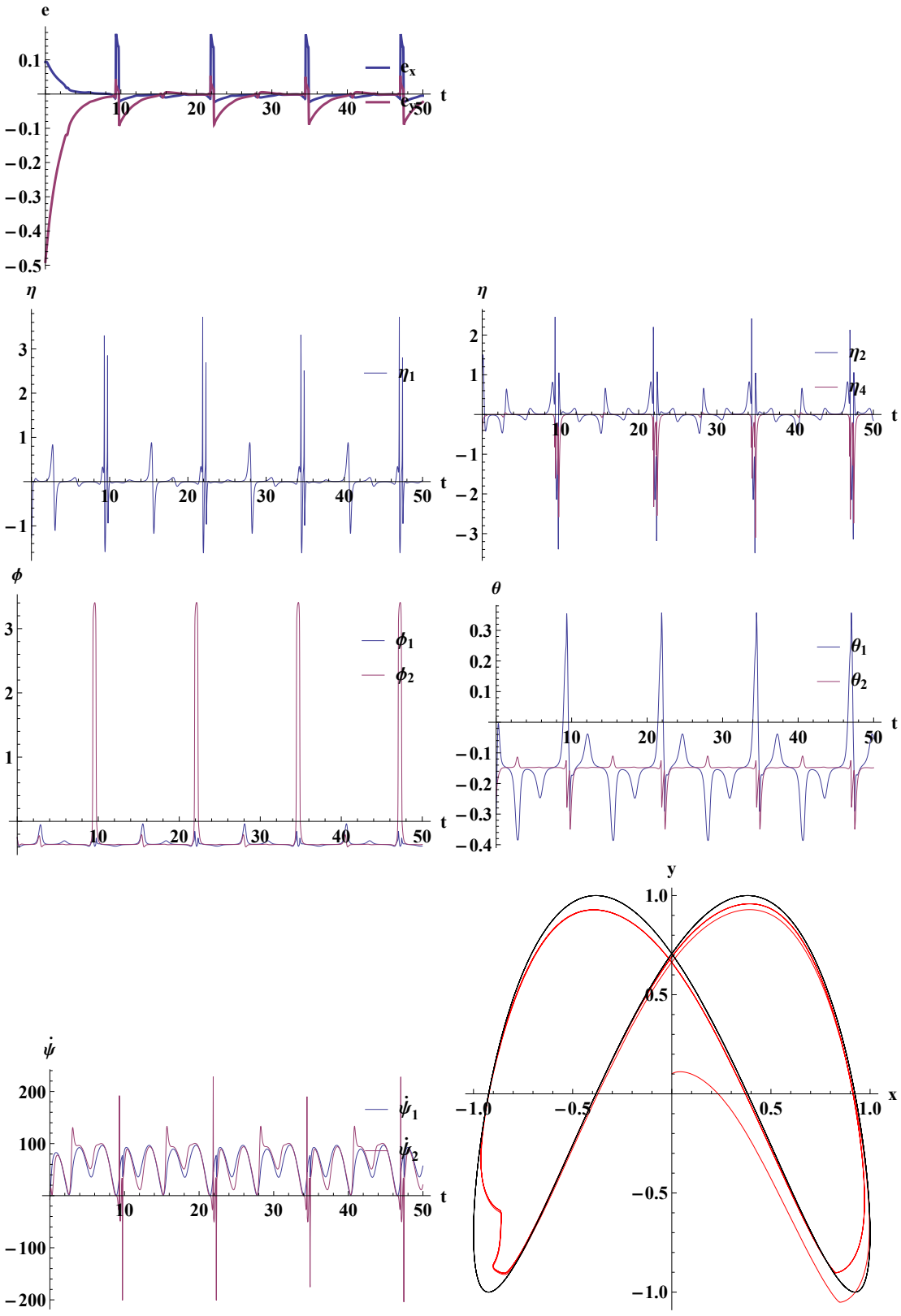
Rysunek 4.11: Przeniesienie sterowań w trybie „offline” dla modelu (3.1) – linearyzacja statyczna



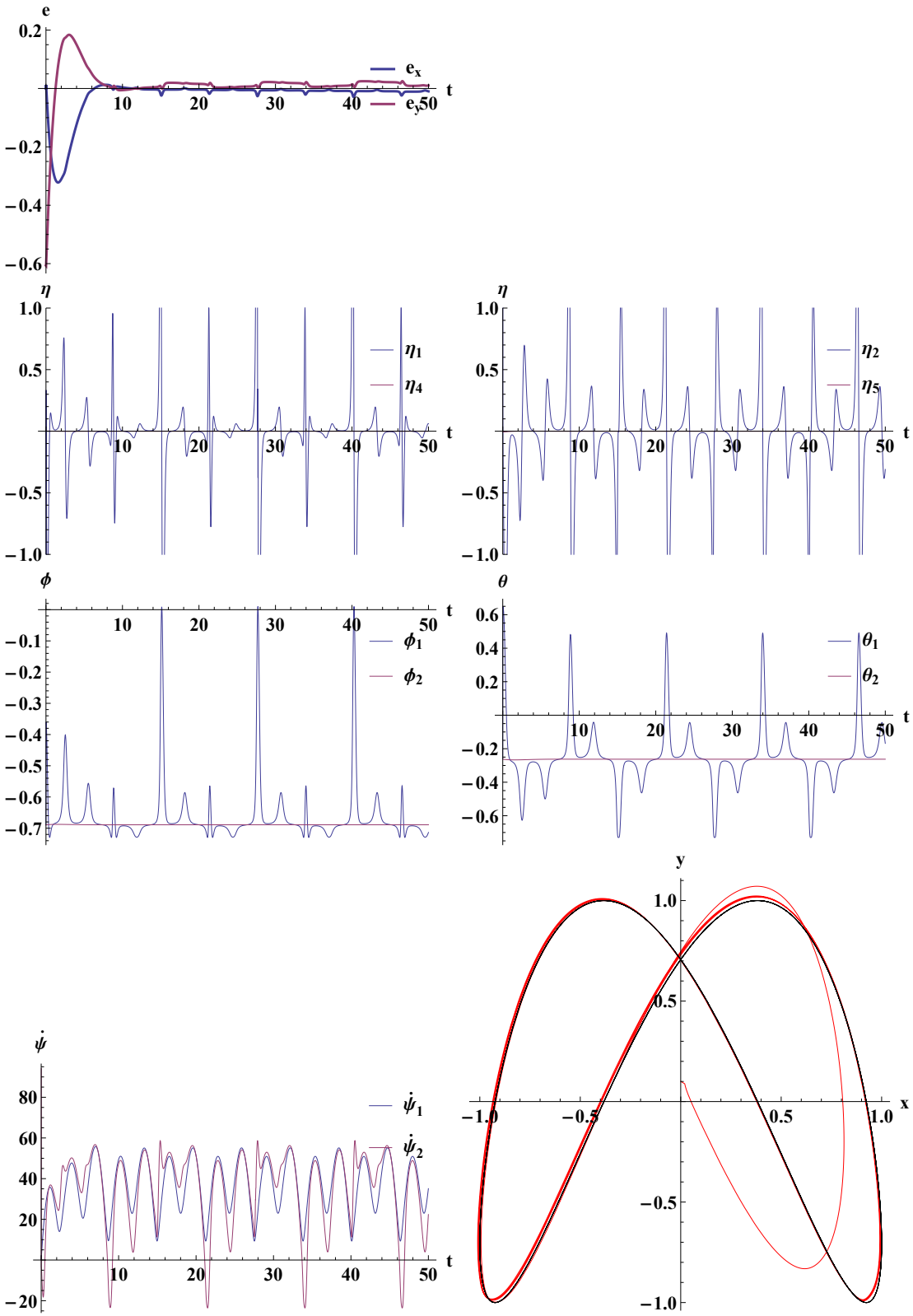
Rysunek 4.12: Przeniesienie sterowań w trybie „offline” dla modelu (3.2) – linearyzacja statyczna



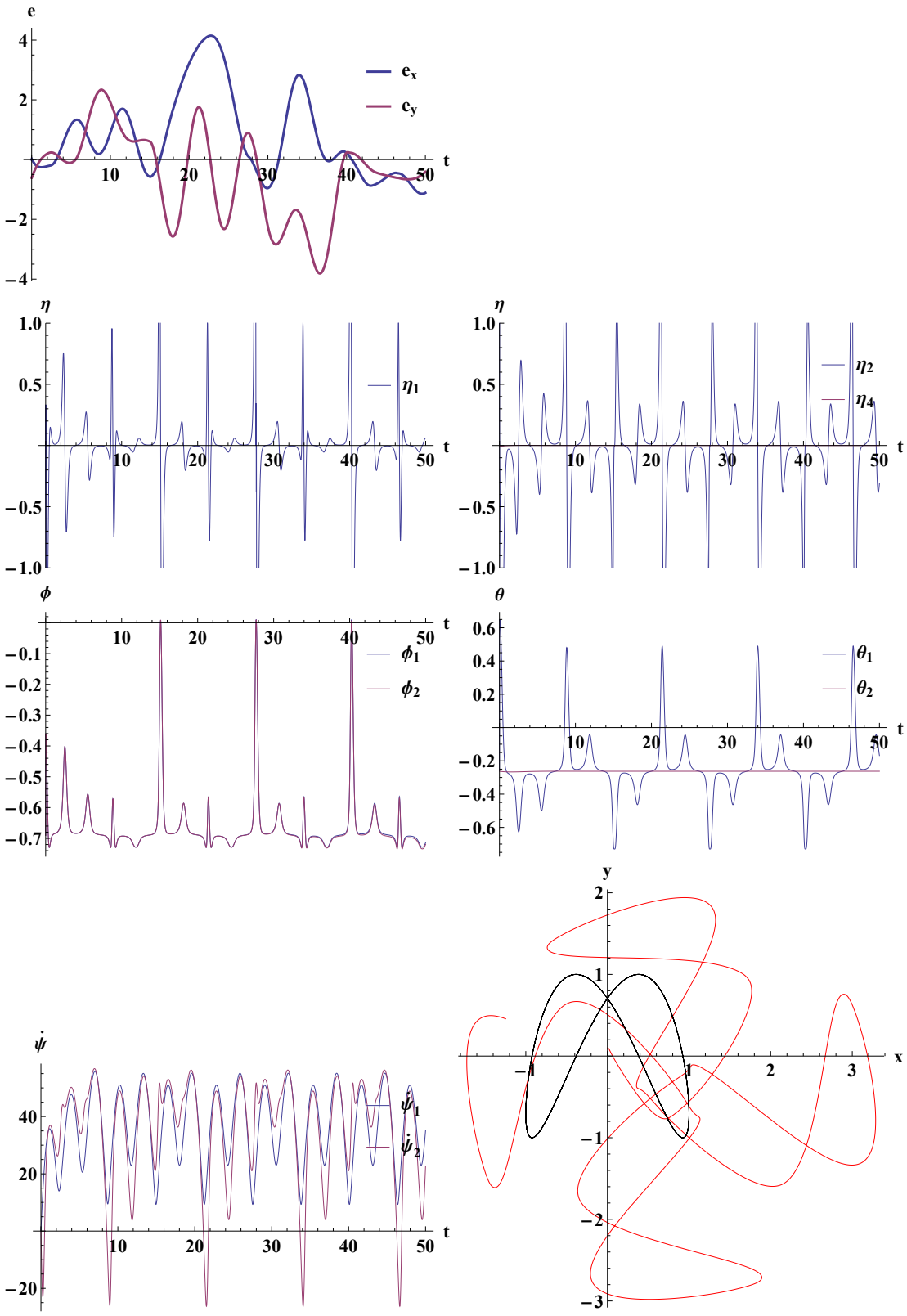
Rysunek 4.13: Przeniesienie sterowań w trybie „online” dla modelu (3.1) – linearyzacja statyczna



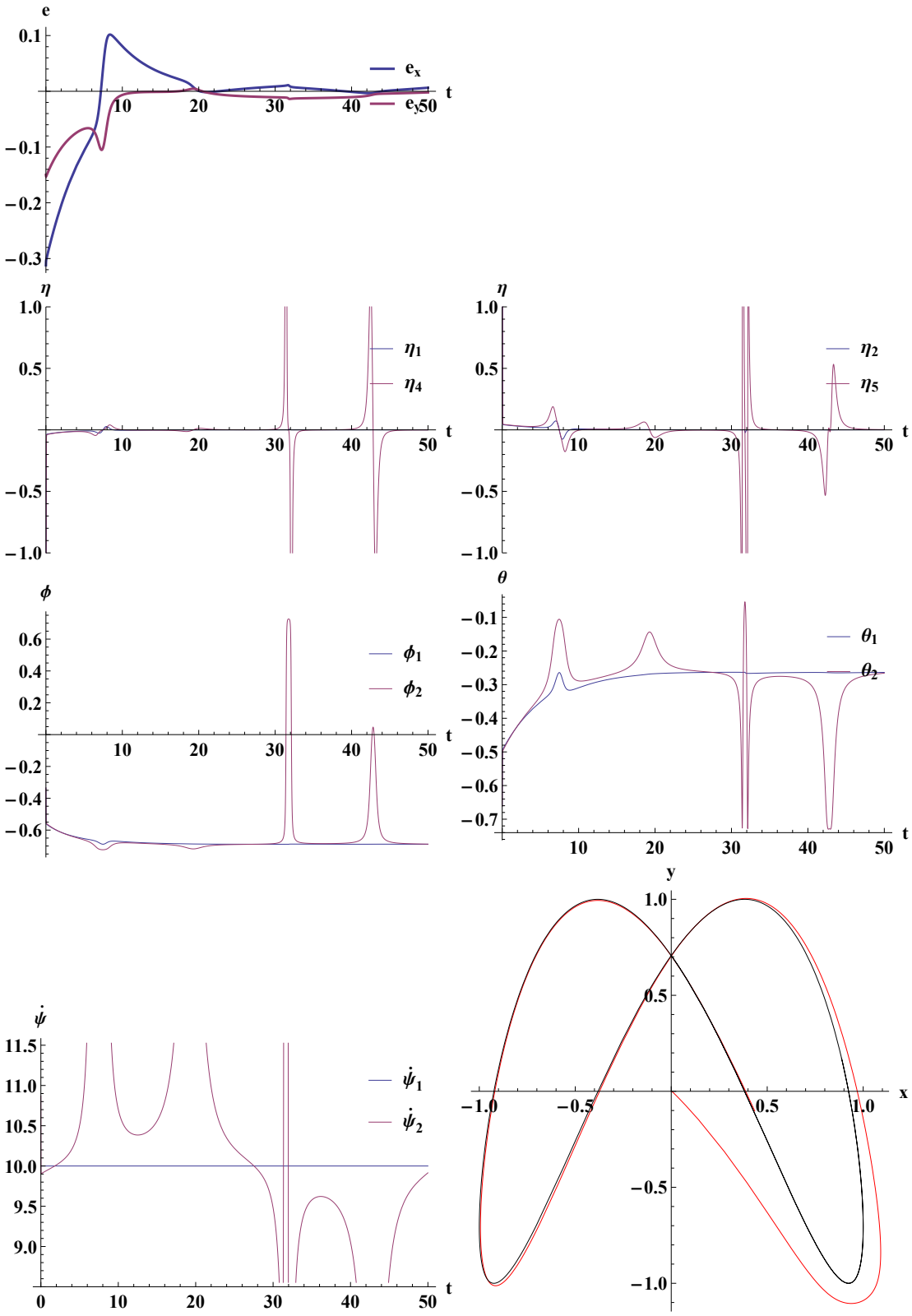
Rysunek 4.14: Przeniesienie sterowań w trybie „online” dla modelu (3.2) – linearyzacja statyczna



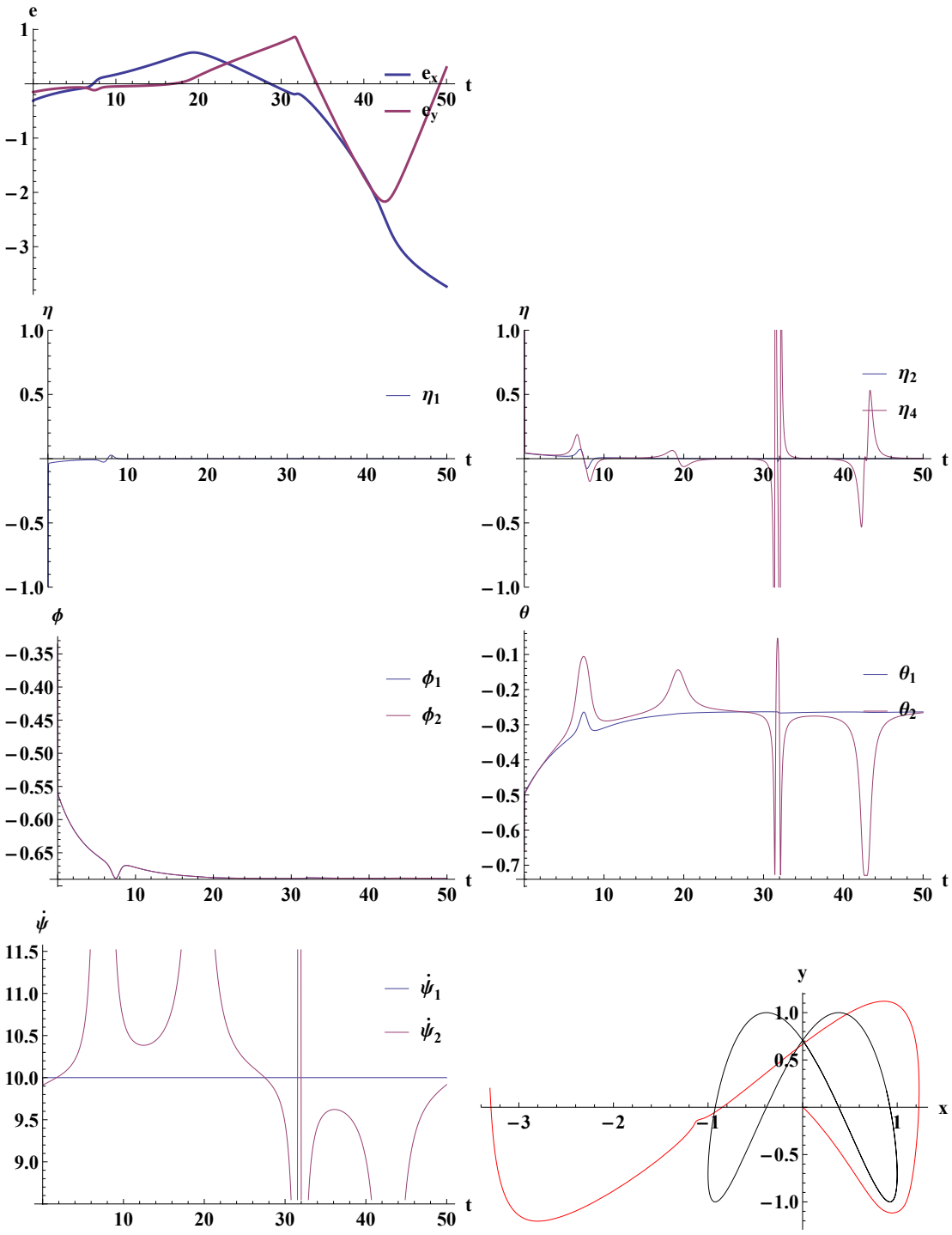
Rysunek 4.15: Przeniesienie sterowań w trybie „offline” dla modelu (3.1) – linearyzacja dynamiczna



Rysunek 4.16: Przeniesienie sterowań w trybie „offline” dla modelu (3.2) – linearyzacja dynamiczna



Rysunek 4.17: Przeniesienie sterowań w trybie „offline” dla modelu (3.1) – algorytm Samsona



Rysunek 4.18: Przeniesienie sterowań w trybie „offline” dla modelu (3.2) – algorytm Samsona

że błędy zbiegają się prawie do zera. Oczywiście zwiększenie wzmocnień przyśpiesza ten proces, jednak błędy nigdy nie spadają do zera.

Rozdział 5

Podsumowanie

Celem pracy było rozwiązanie zadania sterowania dla robota napędzanego dwiema wirującymi półsferami, sprawdzenie możliwości implementacji i działania znanych algorytmów sterowania robotami mobilnymi dla tego typu robota a także jego prostszej wersji z dwoma skrętnymi kołami oraz próba poprawienia lub zaproponowania nowych algorytmów. Wszystkie te cele zostały zrealizowane.

Przeprowadzono wiele symulacji, których celem było zbadanie i przetestowanie zaproponowanych algorytmów. Okazało się iż, mimo że linearyzacja statyczna pozwala rozwiązać zadanie śledzenia zadanej trajektorii to pojawiające się oscylacje całkowicie uniemożliwiają jej implementację na rzeczywistym robocie. Pokazano, że dla robota symultanicznego rozwiązaniem może być użycie linearyzacji dynamicznej, jednak nie sprawdziła się ona dla modelu pełnego. Udało się przedstawić skuteczny i łatwy w implementacji naiwny algorytm sterowania dla modelu symultanicznego. Ostatecznie zaprezentowano modyfikację algorytmu linearyzacji dynamicznej w postaci algorytmu JPTD. Poprzez uproszczenie macierzy G udało się zlikwidować oscylacje nie tracąc znacząco na jakości śledzenia trajektorii.

Kolejno zaprezentowano i zbadano algorytmy sterowania dla modelu robota ze skrętnymi kołami. Pokazano iż sterowania mogą tu zostać przeniesione na model pełny. Możemy tego dokonać prostszą metodą w trybie „offline" bez możliwości korekcji lub z większym nakładem obliczeniowym w trybie „online", co pozwala na bieżąco korygować błędy wynikające z różnic między modelami.

Ważnym zadaniem na przyszłość jest sprawdzenie otrzymanych tutaj wyników na rzeczywistym robocie. Należy także wyprowadzić model dynamiki robota, uwzględnić go w symulacjach i zaproponować sterowniki z jego wykorzystaniem.

Rozdział 6

Dla potomnych

Teraz autor niniejszej pracy napisze co mu na sercu leży. Taka notka dla kogoś (nie ja) kto będzie kontynuował prace nad *robotem mobilnym poruszającym się na dwóch wirujących półsferach*. Po pierwsze ta praca to głownie zbiór rzeczy których **nie należy robić**. Ale do rzeczy.

Model symultaniczny. Sprawdził się jako łatwe wejście w hoggera, ale to już dla niego koniec no chyba, że chcesz też zacząć od czegoś lżejszego.

Model pełny. Po pierwsze zapomnij o linearyzacji statycznej i dynamicznej. Z tego nic nie będzie. **Nie walcz z oscylacjami.** Algorytm JPTD (tu możemy zdradzić pochodzenie jego nazwy: „Ja pierdolę to działa") zdaje się być odpowiedzią na modlitwy. Oczywiście nie jest tak różowo. Działa tylko z modelem „phi" (3.2), ale to chyba dobra wiadomość, bo mniej roboty jest. Wstępne badania wykazały, że działa dla większości przypadków ale to należy dalej sprawdzić. Kolejno trzeba zwrócić uwagę na sam początek, bo pojawiają się spore wartości sterowań. Nie można też przegiąć z wzmocnieniami (mimo iż błędy mniejsze to coś może jebnąć). Myślę, że należałoby rozpisać cały wzór na sterowanie i zastanowić się nad nim. To co na pewno można zrobić (sprawdziłem), to po liczeniu wzorów na sterowania (dostajemy bezpośrednio sterowania na η_1, η_2, η_4 i wzory na pochodne $\dot{\eta}_3, \dot{\eta}_5$) możemy zamiast rozszerzać układ o te pochodne (są to pochodne sterowań wirowaniami) po prostu przyjąć że η_3, η_5 mają stałą wartość no bo przecież o to nam chodzi. Wstawiamy te wartości do pierwszych trzech wzorów (na η_1, η_2, η_4) a za η_3, η_5 po prostu przyjmujemy stała wartość. Oczywiście zostaje kwestia wyznacznika K_{dd}. Niby druga półsfera nie może się wyprostować, ale jakoś w symulacjach wychylenia przelatują przez zero jakby nigdy nic... Ale na pewno robot nie może się zatrzymać, to znaczy zatrzyma się pierwsza półsfera, ale druga będzie wywijać jak oszalał, wiec może jakieś sterowanie z przełączaniem?

Model uproszczony. No tu zaczyna się zabawa. No to lecimy. Linearyzacja statyczna przeniesienie „offline". Zdecydowanie tylko dla modelu „psi" (3.1). Można też spróbować jakiegoś algorytmu kawałkami przenoszenia „offline"? Co do przenoszenia „online" to zdaje się że model „psi" i tutaj działa lepiej, ale ogólnie to gówno działa... tak raz na 10. Należy porządnie policzyć pochodne przekształceń odwrotnych z uwzględnieniem znaków (jakie znaki opisałem w inż. [3], chyba nawet dobrze). Badania pokazały, że stosowanie macierzy wzmocnień w formie $k_1 I_5$ nie sprawdza się, mi wyszło że element odpowiedzialny za θ_{u2} (czyli 3.) powinien być większy nawet o rząd.

Kolejno wspomniano gdzieś tam wyżej, że w linearyzacji statycznej nie da się sterować prędkościami $\dot{\phi}_{u1}, \dot{\phi}_{u1}$ ale to nie do końca prawda. No znaczy prawda, ale łatwo to naprawiamy. Stosujemy linearyzację statyczną tylko dla pierwszych trzech współrzędnych linearyzujących tzn. $(x + dyszel, y + duszel, \theta_{u2})^T$ a dla sterowań promieniami η_4, η_5 wybieramy prosty sterownik P w postaci $\eta_4 = k_1(\dot{\phi}_{u1} - \dot{\phi}_{u1d})$, gdzie za $\dot{\phi}_{u1}$ można wstawić

wzór na η_2. Ale ogólnie nie wiem czy jest to warte zachodu.

Model uproszczony linearyzacja dynamiczna. Tu to niewiele mogę powiedzieć. Z użyciem odwzorowania sterującego prędkościami kół należy uważać, bo promienie mogą uniemożliwić przekształcenia no i fakt że można tylko „offline" trochę osłabia tę opcję. Może jakby tak model „psi" odcinkami „offline" z rozważną manipulacją prędkościami...

Algorytm Samsona. Tutaj dużo należało by jeszcze zbadać ale na pewno działa częściej niż linearyzacja statyczna czy dynamiczna. Należy się zastanowić czy dało by się zrobić to w trybie „online".

Ale tak po prawdzie to wszystko i tak na nic po, jak przyjdzie co do czego to zawsze wygrywa prostota, czyli właśnie sterowanie naiwne do poruszania robota w x,y w zadanym kierunku z zadana prędkością. Natomiast do obrotu wokół własnej osi trzeba wyprowadzić model z środkiem po środku i tam na pewno da się jakoś to zrobić.

Bibliografia

[1] C. S. Alain Micaelli. Trajectory tracking for unicycle-type and two-steering-wheels mobile robots. Raport instytutowy 2097, Institut National De Recherche En Informatique Et En Automatique, 1993.

[2] C. C. de Wit, G. Bastin, B. Siciliano, redaktorzy. *Theory of Robot Control*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, wydanie 1st, 1996.

[3] P. Joniak. Badania symulacyjne zachowania robota mobilnego napędzanego dwiema półsferami. Praca inżynierska, Politechnika Wrocławska, 2015.

[4] Wolfram Mathematica. http://www.wolfram.com/mathematica/.

[5] A. Mazur. *Sterowanie oparte na modelu dla nieholonomicznych manipulatorów mobilnych*. Oficyna Wydawnicza Politechniki Wrocławskiej, Wrocław, 2009.

[6] K. Tchoń, A. Mazur, I. Dulęba, R. Hossa, R. Muszyński. *Manipulatory i roboty mobilne: modele ruchu, sterowanie*. Akademicka Oficyna Wydawnicza, Warszawa, 2000.

[7] Unknown. Hemisphere drive speedster. *Mechanics and Handicraft*, 5(9):23,73, 1938. Przez Modern Mechanix, http://blog.modernmechanix.com/hemisphere-drive-speedster/.