Sterowanie Mechanizmów Wieloczłonowych

Jakub Możaryn, Jan Klimaszewski

24 września 2018

Spis treści

Rozdział 1

Opis mechanizmów wieloczłonowych - Janek+Kuba

W rozdziale przedstawiono podstawy matematyczne i fizyczne służące do opisu robotów o wielu stopniach swobody (mechniazmów wieloczłonowych). Przedstawiono model robota w przestrzeni zmiennych stanu. Opisano sposób symulacji komputerowej dynamiki robota.

1.1 Model geometrii, kinematyki i dynamiki mechanizmów wieloczłonowych.

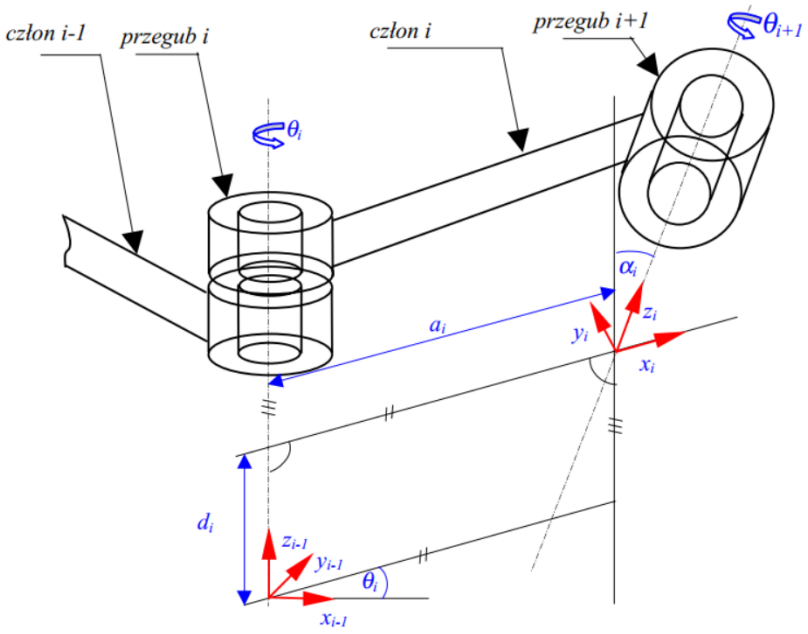
1.1.1 Model geometrii - Janek

W pracy rozważany jest robot o pojedynczym łańcuchu kinematycznym, składający się z członów połączonych przegubami. Każdy z przegubów posiada jeden stopień swobody. W dalszej części pracy rozważany będzie robot o n-stopniach swobody z przegubami typu obrotowego, gdyż tylko przeguby tego typu występują w rozważanym w pracy robocie PUMA 560. Przyjęto, że dla robota o n-przegubach ponumerowanych od 1 do n istnieje n+1 członów ponumerowanych od 0 do n. Człon o numerze 0 nazywany jest członem bazowym, natomiast człon n jest członem z elementem wykonawczym. Przegub i łączy człony o numerach i-1 i i. Z każdym z członów związany jest układ współrzędnych o tym samym numerze.

1.1.2 Model kinematyki - Janek

Dla określenia układów współrzędnych kartezjańskich, które będą przyporządkowane w odpowiedni sposób do członów i przegubów przyjęto notację Denavita-Hartenberga [1], której podstawy zamieszczono na Rys. 1.1:

Po określeniu parametrów układów współrzędnych kartezjańskich można wyznaczyć jednorodne macierze przekształceń określających orientację i położenie układu i względem układu $i-1$ [2]:



Rysunek 1.1: Standardowa notacja Denavita Hartenberga (przykład dwóch obrotowych przegubów i-1 oraz i), a_i - długość członu, odległość od osi $z_i - 1$ do osi z_i mierzona wzdłuż osi x_i, α_i kąt skręcenia, kąt między osiami $z_i - 1$ i z_i mierzony wokół osi x_i, d_i - odsunięcie członów, odległość od środka układu $i-1$ do osi x_i mierzona wzdłuż osi $z_i - 1$, θ_i - kąt konfiguracji, kąt między osiami $x_i - 1$ i x_i mierzony wokół osi $z_i - 1$

$$^{i-1}Ti = T_{z,\theta}T_{z,d}T_{x,a}T_{x,\alpha} \tag{1.1}$$

gdzie: ^{i-1}Ti - jednorodna macierz przekształcenia układu $i-1$ względem układu i,$T_{z,\theta}$ - jednorodna macierz obrotu układu i względem układu $i-1$ wokół osi z_i-1 o kąt θ_i, $T_{z,d}$ - jednorodna macierz przesunięcia układu i względem układu $i-1$ o d_i wzdłuż osi $z_i - 1$,$T_{x,a}$ - jednorodna macierz przesunięcia układu i względem układu $i-1$ o a_i wzdłuż osi x_i. ,$T_{x,\alpha}$-jednorodna macierz obrotu układu i względem układu $i-1$ wokół osi x_i o kąt α_i.

Macierze te wyrażone są następującymi zależnościami:

$$T_{z,\theta} = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & 0 & 0 \\ \sin\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$$T_{z,d} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.3}$$

$$T_{x,a} = \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.4}$$

$$T_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\sin\alpha_i & 0 \\ 0 & \sin\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1.5}$$

Wykorzystując jednorodne macierze przekształceń można obliczyć współrzędne punktu w m-tym układzie, znając współrzędne tego samego punktu w k-tym układzie (zakładając, że $k > m$), jest to tzw. proste zadanie kinematyki. Zależność tą wyznacza się korzystając z wyrażenia [2]:

$$P_{Am} = {}^{m}T_k P_{Ak}, k > m \tag{1.6}$$

gdzie:

$${}^{m}T_k = {}^{m}T_{m+1} \quad {}^{m+1}T_{m+2} \ldots {}^{k-1}T_k \tag{1.7}$$

$P_{Ak} = \begin{bmatrix} x_{Ak} \\ y_{Ak} \\ z_{Ak} \\ 1 \end{bmatrix}$ - jednorodny wektor współrzędnych punktu A w k-tym układzie współrzędnych

$P_{Am} = \begin{bmatrix} x_{Am} \\ y_{Am} \\ z_{Am} \\ 1 \end{bmatrix}$ - jednorodny wektor współrzędnych punktu A w m-tym układzie współrzędnych

1.1.3 Model dynamiki - Kuba

Zależności opisujące dynamikę robota o n-stopniach swobody można wyznaczyć wykorzystując uogólnione równanie Lagrange'a - Eulera o postaci [2]

$$\tau_i = \frac{d}{dt}\left(\frac{\delta L}{\delta \dot{q}_i}\right) - \frac{\delta L}{\delta q_i}, i = 1, 2, \ldots, n \tag{1.8}$$

gdzie: L - funkcja Lagrangea (lagrangian) funkcja potencjału kinetycznego.

$$L = K - P \tag{1.9}$$

K - całkowita energia kinetyczna układu, P - całkowita energia potencjalna układu, q_i - uogólnione współrzędne układu w przegubie i, τ_i - uogólnione wymuszenia (siły lub momenty) działające w przegubie i [1].

Korzystając z równań Lagrangea-Eulera, zależności opisujące dynamikę robota z wszystkimi przegubami obrotowymi, można przedstawić w następujący sposób [2].

$$\tau = M(\theta)\ddot{\theta} + V(\theta, \dot{\theta}) + G(\theta) \tag{1.10}$$

[1] W robocie o przegubach obrotowych, jako uogólnione współrzędne przyjmuje się położenia kątowe w przegubach, natomiast jako uogólnione wymuszenia przyjmuje się momenty napędowe w przegubach.

gdzie: $\tau \in R^n$ - wektor momentów napędowych w przegubach, n - ilość stopni swobody robota, $M(\theta) \in R^{n\times n}$ - macierz bezwładności robota. $V(\theta,\dot{\theta})$ - wektor wyrazów zawierających składowe momentu zależne od sił odśrodkowych i Coriolisa, $G(\theta) \in R^n$- wektor wyrazów zawierających składowe momentu zależne od sił grawitacji, $\theta \in R^n$ - wektor położeń kątowych w poszczególnych przegubach.

Macierz $M(\theta) \in R^{n\times n}$

Elementy macierzy bezwładności wyznacza się korzystając z zależności:

$$m_{ik} = \sum_{j=\max(i,k)}^{n} Tr(U_{jk} J_j U_{jk}^T), i,k = 1,2,\ldots,n \tag{1.11}$$

gdzie:

$$U_{jk} \equiv \frac{\delta^0 T_i}{\delta\theta_j} = \begin{cases} {}^0T_{j-1} Q^{j-1} T_i, \text{dla} j \leqslant i \\ 0, \text{dla} j > i \end{cases} \tag{1.12}$$

$$Q = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{- dla przegubu obrotowego} \tag{1.13}$$

J_i - jednorodna macierz inercji członu i wyrażona zależnością:

$$J_i = \int r_i r_i^T dm = \begin{bmatrix} \frac{-I_{XXi}+I_{YYi}+I_{ZZi}}{2} & I_{XYi} & I_{XZi} & m_i\overline{x}_i \\ I_{XYi} & \frac{I_{XXi}-I_{YYi}+I_{ZZi}}{2} & I_{YZi} & m_i\overline{y}_i \\ I_{XZi} & I_{YZi} & \frac{I_{XXi}+I_{YYi}-I_{ZZi}}{2} & m_i\overline{z}_i \\ m_i\overline{x}_i & m_i\overline{y}_i & m_i\overline{z}_i & m_i \end{bmatrix} \tag{1.14}$$

I_{XXi},I_{YYi},I_{ZZi} - masowe momenty bezwładności członu i, I_{XYi},I_{XZi},I_{YZi} - masowe momenty dewiacji członu i, $r_i = [x_i \quad y_i] \quad z_i]^T$ - jednorodny wektor współrzędnych środka masy członu i wyrażony w i-tym układzie współrzędnych, $Tr(A) = \sum_{i=1}^{n} a_{ii}$ - ślad macierzy kwadratowej $A \in R^{n\times n}$.

Wektor $V(\theta,\dot{\theta})$ **ma postać**

$$V(\theta,\dot{\theta}) = [V_1, V_2, \ldots, V_n]^T \tag{1.15}$$

Elementy wektora $V(\theta,\dot{\theta})$ można wyznaczyć korzystając z następujących zależności:

$$V_i = \dot{\theta}^T H_i \dot{\theta} \tag{1.16}$$

$$H_i \in R^{n\times n}, i = 1,2,\ldots,n \tag{1.17}$$

$$h_{ikl} = \sum_{j=\max(i,k,l)}^{n} Tr(U_{jkl} J_j U_{ji}^T), i,k,l = 1,2,\ldots,n \tag{1.18}$$

gdzie:

$$U_{ijk} \equiv \frac{\delta^0 U_{ij}}{\delta\theta_k} = \begin{cases} {}^0T_{j-1}Q_j^{j-1}T_{k-1}Q_k^{k-1}T_i, \text{dla} j \leqslant k \leqslant i \\ {}^0T_{k-1}Q_k^{k-1}T_{j-1}Q_j^{j-1}T_i, \text{dla } k \leqslant j \leqslant i \\ 0, \text{dla } j > i \text{ lub } i < k \end{cases} \tag{1.19}$$

Wektor $G(\theta)$ ma postać

Elementy wektora $G(\theta)$ można wyznaczyć korzystając z zależności:

$$g_i = \sum_{j=i}^{n}(-m_j\overline{g}U_{ij}^j\overline{r}_j), i = 1, 2, \ldots, n \tag{1.20}$$

gdzie: $\overline{g} = [g_x \quad g_y \quad g_x \quad 0]^T$ -jednorodny wektor grawitacji wyrażony w bazowym układzie współrzędnych, $\sqrt{\overline{g}\overline{g}^T} = 9.8062[m/s2]$ - na powierzcni ziemi, m_i masa członu i.

Należy dodać, że istnieje szereg innych metod wyznaczania równań dynamiki robota, np. metoda Newtona-Eulera, uogólniona metoda dAlamberta [2], metoda Christoffela-Lagrangea, metoda Walkera-Orina [3]. Ich powstanie wynikło głównie z potrzeby stworzenia szybkich algorytmów do obliczeń numerycznych. Dokładniejszą analizę i porównanie algorytmów można znaleźć w [2, 3]. W dalszej części pracy wykorzystywany jest algorytm Lagrangea-Eulera. Wyznaczona tym sposobem postać równań dynamiki 1.1.3 pozwala na łatwą analizę właściwości robota, zaprojektowania uniwersalnego algorytmu do jego symulacji, oraz syntezę układu sterowania.

Model tarcia

W uk?adach mechanicznych wyrózniamy rózne rodzaje si?. Pierwsze z nich, to **si?y zachowawcze**, dla których której praca wykonana przy przemieszczeniu cia?a po torze zamkni?tym o dowolnym kszta?cie równa jest zeru. Si?y nie spe?niaj?ce tego warunku to tzw **si?y dyssypatywne**, inaczej nazywane **si?ami rozpraszaj?cymi**.

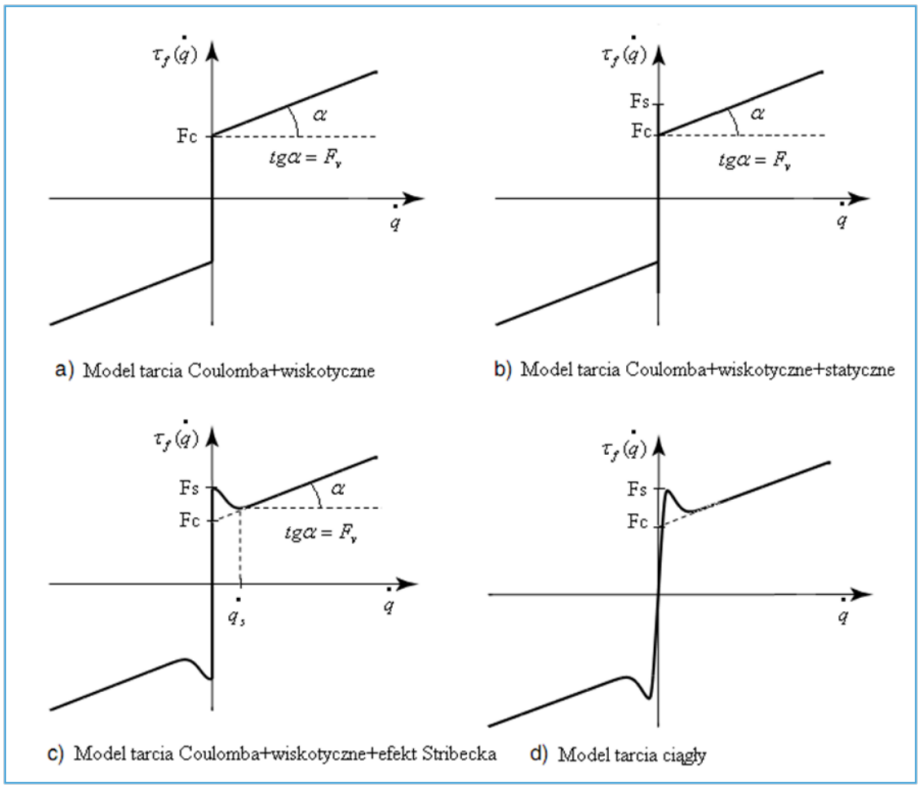
Si?y tarcia, wystepuj?ce w uk?adach mechanicznych, mo?emy zaliczy? do si? dyssypatywnych. Wyst?puj?ce w uk?adach mechanicznych tarcie jest nieliniowym i niestacjonarnym zjawiskiem o parametrach roz?o?onych. W systemach mechatronicznych zmniejsza ono dok?adno?? oraz pogarsza jako?? przebiegów dynamicznych.

Modele tarcia najogólniej mo?na podzieli? na trzy grupy [21]:

- white-box - wykorzystuj? podstawy fizyczne badanego zjawiska i dzieli si? je na statyczne oraz dynamiczne.
- black-box - modele parametryczne, bazuj?ce na danych eksperymentalnych.
- grey-box - ??cz? cechy modeli white-box i black-box.

W modelowaniu fizycznym przy modelowaniu mechanizmów wielocz?oowych stosuje si? zwykle statyczne modele typu white-box, które przy pomocy równa? algebraicznych, opisuj? podstawowe w?asno?ci tarcia. Wyró?nia sie w?ród nich

modele klasyczne uwzgl?dniaj?ce ró?ne kombinacje tarcia *Coulomba* (parametr F_c), wiskotycznego (parametr F_v), statycznego (parametr F_s) oraz efektu *Stribecka*. Na rys. 1.1.3 przedstawiono ró?ne modele tarcia statycznego.



Rysunek 1.2: Modele tarcia statycznego

Modele tarcia, przedstawione na rys. 1.1.3 opisywane s? nast?puj?cymi zale?no?ciami:

- Model (a)

$$\tau_f(\dot{q}) = F_v\dot{q} + F_C \text{sign}\dot{q} \tag{1.21}$$

- Model (b)

$$\tau_f(\dot{q}) = \begin{cases} \pm F_s & \text{dla} \quad \dot{q} = 0 \\ F_v\dot{q} + F_C \text{sign}\dot{q} & \text{dla} \quad \dot{q} \neq 0 \end{cases} \tag{1.22}$$

- Model (c)

$$\tau_f(\dot{q}) = \begin{cases} \pm F_s & \text{dla} \quad \dot{q} = 0 \\ F_v\dot{q} + \left(F_C + (F_s - F_C)\exp\left(-\dfrac{\dot{q}}{\dot{q}_s}\right)^2\right)\text{sign}(\dot{q}) & \text{dla} \quad \dot{q} \neq 0 \end{cases} \quad 0 \tag{1.23}$$

- Model (d) - w którym zdefiniowano niewielkie otoczenie, wewn?trz którego przyjmuje si? zerow? warto?? pr?dko?ci. W tym przypadku tarcie dla jest

zale?ne od zewn?trznych si? (F_e) utrzymuj?cych uk?ad w spoczynku, za? poza przyj?tym otoczeniem jest zazwyczaj opisywane w funkcji pr?dko?ci (Model (a), (b), lub (c))

$$\tau_f(\dot{q}, F_e) = \begin{cases} \tau_f(\dot{q}) & \text{dla} \quad |\dot{q}| \geqslant \beta \\ \tau_f(\dot{F}_e) & \text{dla} \quad |\dot{q}| \geqslant \beta \end{cases} \tag{1.24}$$

1.1.4 Właściwości strukturalne modelu dynamiki

Model matematyczny robota (1.1.3) posiada charakterystyczne własności strukturalne. Są one często wykorzystywane podczas weryfikacji poprawności modeli otrzymanych różnymi metodami w trakcie projektowania i analizy układów sterowania robotów. Podstawowe własności, które będą wykorzystywane w pracy, są następujące [4, 5, 6, 7]

Własności strukturalne modelu robota (WMR)

WMR.1 Macierz inercji $M(q)$ jest symetryczna

$$M(q) = M^T(q) \tag{1.25}$$

WMR.2 Macierz inercji $M(q)$ jest dodatnio określona

$$x^T M(q) x > 0 \qquad \forall x = [x_i]_{n\times 1}, x \neq 0 \tag{1.26}$$

WMR.3 Elementy $m_{i,j}(q)$ macierzy inercji $M(q)$ nie zależą od przemieszczeń uogólnionych w 'aktualnym' przegubie i przegubach 'poprzednich'

$$m_{i,j}(q) = m_{i,j}(q_{k+1}, q_{k+2}, \ldots, q_n), \text{ gdzie } k = \min(i,j), \ \forall i,j = 1,\ldots,n \tag{1.27}$$

Z własności *WMR.1* - *WMR.3* wynikają następujące wnioski

1. Macierz $M(q)$ jest nieosobliwa

$$\det[M(q)] \neq 0 \tag{1.28}$$

2. Można dokonać dekompozycji macierzy $M(q)$

$$M(q) = P^T P \qquad \det P \neq 0 \tag{1.29}$$

3. Żaden element macierzy inercji nie zależy od przemieszczenia uogólnionego w pierwszym przegubie, natomiast element $m_{n,n}$ nie zależy od przemieszczeń uogólnionych w żadnym z przegubów i ma wartość stałą.

4. Odwrócona macierz inercji $\bar{M}(q) = [\bar{m}_{i,j}(q)]_{n\times n}$ jest symetryczna

$$\bar{M}(q) = M(q)^{-1} = \bar{M}^T(q) \tag{1.30}$$

5. Odwrócona macierz inercji $\bar{M}(q)$ jest dodatnio określona

$$x^T \bar{M}(q) x > 0 \qquad \forall x = [x_i]_{n\times 1}, x \neq 0 \tag{1.31}$$

6. Każdy element odwróconej macierzy inercji $\bar{M}(q)$ jest funkcją przemieszczeń uogólnionych we wszystkich przegubach oprócz pierwszego

$$\bar{m}_{i,j}(q) = \bar{m}_{i,j}(q_2, \ldots, q_n) \qquad \forall i, j = 1, \ldots, n \tag{1.32}$$

Model matematyczny robota o n-stopniach swobody z czasem dyskretnym można otrzymać aproksymując pochodne przemieszczeń uogólnionych []kurek:neural (różniczkowanie metodą *Eulera*)

$$\dot{q}(k) \approx \frac{q(k) - q(k-1)}{T_p} \tag{1.33}$$

$$\ddot{q}(k) \approx \frac{\dot{q}(k+1) - \dot{q}(k)}{T_p} \approx \frac{q(k+1) - 2q(k) + q(k-1)}{{T_p}^2} \tag{1.34}$$

gdzie T_p - okres próbkowania, k - czas dyskretny, $t = kT_p$.

Po podstawieniu zależności (1.33) i (1.34) do równania (1.1.3) otrzymuje się model z czasem dyskretnym robota w postaci równań *Lagrange'a - Eulera*

$$M[q(k)]\frac{q(k+1) - 2q(k) + q(k-1)}{{T_p}^2} + V[q(k), q(k-1)] + G[q(k)] = \tau(k) \tag{1.35}$$

gdzie, analogicznie jak w modelu (1.1.3), $M[q(k)] = [m_{i,j}[q(k)]]_{n\times n}$ - macierz inercji (bezwładności) robota, $V[q(k), q(k-1)] = [v_i[q(k), q(k-1)]]_{n\times 1}$ - wektor oddziaływań zależnych od sił odśrodkowych i sił *Coriolisa*, $G[q(k)] = [g_i[q(k)]]_{n\times 1}$ - wektor oddziaływań zależnych od sił grawitacji.

Model z czasem dyskretnym robota (1.35) zachowuje własności strukturalne modelu robota *WMR.1 - WMR.3.*

1.2 Opis mechanizmów wieloczłonowych w przestrzeni stanu.

Równanie dynamiki robota w przestrzeni zmiennych stanu można wyznaczyć w oparciu o macierzowe równanie dynamiki (1.1.3). Definiując wektor stanu dla robota w postaci [2]:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{1.36}$$

oraz przyjmując, że wejściami do układu są sygnały sterujące τ , natomiast wyjściami położenia kątowe θ , równanie dynamiki (1.1.3) można wyrazić w przestrzeni zmiennych stanu jako

$$\begin{aligned} \dot{x} &= \begin{bmatrix} x_2 \\ -M^{-1}(\theta)[V(\theta, \dot{\theta}) + G(\theta)] \end{bmatrix} + \begin{bmatrix} 0 \\ M^{-1}(\theta) \end{bmatrix} \tau \\ y &= [I \quad 0]\, x \end{aligned} \tag{1.37}$$

Stosując standardowe oznaczenia modeli stanu, układ równań (1.37) zapisuje się następująco

$$\begin{aligned} \dot{x} &= A(x) + B(x)\tau \\ y &= Cx \end{aligned} \tag{1.38}$$

gdzie:

$$A(x) = \begin{bmatrix} x_2 \\ -M^{-1}(\theta)[V(\theta,\dot{\theta}) + G(\theta)] \end{bmatrix} \tag{1.39}$$

$$B(x) = \begin{bmatrix} 0 \\ M^{-1}(\theta) \end{bmatrix} \tag{1.40}$$

$$C(x) = [I \quad 0] \tag{1.41}$$

Charakterystycznymi właściwościami robota jako obiektu sterowania są nieliniowości wynikające z istnienia sił Coriolisa i sił odśrodkowych, zmienność parametrów układu w czasie, oraz sprzężenia pomiędzy wszystkimi stopniami swobody.

1.3 Identyfikacja parametrów kinematycznych i dynamicznych robotów.

1.4 Zastosowanie kwaternionów.

1.5 Otwarte i zamknięte łańcuchy kinematyczne.

1.6 Układy niedosterowane.

Układy niedosterowane (ang. *underactuated systems*) należą do szerokiej klasy urządzeń i systemów występujących w robotyce oraz teorii sterowania. W ogólności przez **układy niedosterowane** rozumie się systemy, w których liczba możliwych wymuszeń (wejść układu) jest mniejsza od liczby stopni swobody (wyjść układu). W przypadku układów sterowania, omawiane zagadnienie sprowadza się do procesów, w których liczba sterowanych wyjść jest mniejsza od liczby dostępnych sygnałów sterujących.

Układy niedosterowane w teorii sterowania są interesujące ze względu na swoje własności. Niedobór elementów wykonawczych powoduje, że generowane sterowania działają jednocześnie na kilka wyjść i nie ma możliwości na niezależne wpływanie na każde z nich.

W efekcie algorytm regulacji kontrolujący takie urządzenie musi uwzględniać wiele wymagań naraz i wybierać rozwiązanie dla zadanej sytuacji. Badania takich układów były prowadzone między innymi w zakresie wykorzystania teorii chaosu do sterowania niedosterowanym systemem satelitów [**?**], sterowania ślizgowego nieliniowymi układami niedosterowanymi [**?**] lub użycia adaptacyjnego modelu z predykcją do sterowania robotem i minimalizacji zużycia energii [**?**]. Jednym z najprostszych urządzeń niedosterowanych jest wahadło odwrócone. Konstrukcja wahadła pozwala na niezależny ruch wzdłuż osi prowadnicy, na której jest zamontowane oraz obrót wokół własnej osi. Pomimo prostej konstrukcji wahadło odwrócone jest układem często wykorzystywanym w celu testowania różnego rodzaju algorytmów regulacji.

Ze względu na swoją budowę, wahadło posiada chwiejny punkt równowagi w pozycji pionowej, którego opuszczenie powoduje utratę stabilności i upadek do stabilnej pozycji dolnej. Pomimo prostego opisu matematycznego jego zachowań, jest to obiekt nieliniowy, dla którego realizacja sterowania nie jest prosta.

1.7 Układy nieholonomiczne.

1.8 Kolizje.

Rozdział 2

Standardowe modele układów wieloczłonowych - Janek

2.1 Wahadło odwrócone na wózku - Kuba

Wahadło odwrócone składa się z pręta swobodnie przymocowanego do wózka jeżdżącego po prowadnicy. Ruch wózkiem pozwala na realizację przemieszczenia liniowego wahadła, natomiast ruch kątowy pręta jest wynikiem działania sił bezwładności. Wahadło posiada punkt stabilności w dolnym położeniu pręta oraz punkt chwiejnej równowagi w pozycji górnej. W momencie gdy układ jest utrzymywany w górnej pozycji, mówi się o wahadle odwróconym.

Z przedstawionego opisu wynika, że jest to obiekt regulacji typu SIMO (ang. *single-input, multi-output*) i stanowi przykad układu niedosterowanego. Jako sygnał sterujący przyjmuje się moment silnika powodujący przemieszczenie wózka, natomiast sygnałami wyjściowymi jest położenie liniowe i kątowe wahadła.

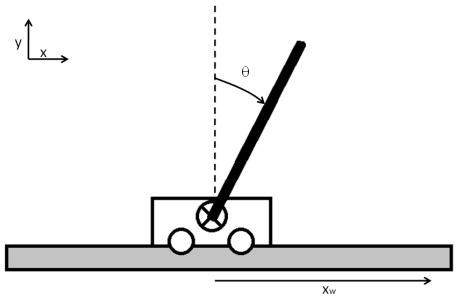
2.1.1 Model nieliniowy w oparciu o równania fizyczne

Model wahadła został wyznaczony w oparciu o schemat przedstawiony na rys.2.1. Początek układu współrzędnych przyjęto w początkowym położeniu wózka. Wózek wahadła ma możliwość przemieszczania się w osi x. Pręt wahadła zamocowany jest jednym końcem na wózku i obraca się wokół osi przechodzącej prostopadle przez wózek o kąt θ. Jest to układ dwóch ciał poruszających się zależnie od siebie. Z tego powodu w dalszych rozważaniach parametry związane z wózkiem zostały oznaczone indeksem $_w$, natomiast parametry związane z prętem indeksem $_p$. Układ sił działających na wózek wahadła przedstawiono na rys. 2.2.

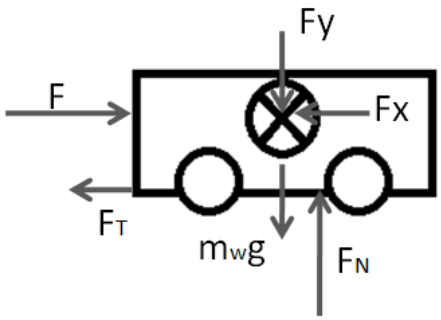
$$m_W \dot{x}_W = \sum F \tag{2.1}$$

$$m_W \dot{x}_W = F - F_T - F_x \tag{2.2}$$

gdzie: m_W oznacza masę wózka, x_W położenie wózka w osi x, F siłę nadaną przez silnik, F_T siłę tarcia wózka o prowadnicę, F_x siłę reakcji pręta wahadła



Rysunek 2.1: Schemat wahadła odwróconego



Rysunek 2.2: Określenie układu sił działających na wózek wahadła

na wózek.

Ponadto przyjęto, że tarcie wózka o prowadnicę jest proporcjonalne do prędkości wózka, tak więc:

$$F_T = b\dot{x}_W \tag{2.3}$$

gdzie: b oznacza współczynnik tarcia wiskotycznego.

Dodatkowo w badanym układzie sygnałem sterującym nie jest siła działająca na wózek, lecz moment siły generowany na wale silnika. Zatem siłę F można określić jako:

$$F = \frac{M_s}{r} \tag{2.4}$$

gdzie: M_s oznacza moment silnika, r promień koła zębatego na silniku.

Ostatnia siła w równaniu (2.1.1) - F_x, będąca oddziaływaniem pręta na wózek, wynika z przyspieszeń działających na poruszający się pręt:

$$F_x = m_P\dot{x}_P \tag{2.5}$$

Położenie środka ciężkości pręta x_P w przyjętym układzie współrzędnych można określić w zależności od aktualnego położenia wózka zgodnie z zależnością:

$$x_P = x_W + l\sin(\theta) \tag{2.6}$$

gdzie: l jest odległością środka ciężkości pręta od wózka.

Znając zależność (2.1.1) można wyznaczyć kolejne pochodne x_P po czasie, odpowiednio:

$$\dot{x}_P = \frac{dx_P}{dt} = \frac{dx_W + l\sin(\theta)}{dt} = \frac{dx_W}{dt} + l\frac{d\sin(\theta)}{dt} = \frac{dx_W}{dt} + l\cos(\theta)\frac{d\theta}{dt} = \dot{x}_W + l\dot{\theta}\cos\theta \tag{2.7}$$

oraz

$$\ddot{x}_P = \frac{d}{dt}\left(\frac{dx_W + l\sin(\theta)}{dt}\right) = \frac{d^2x_W}{dt^2} + \frac{d}{dt}\left(l\cos\theta\frac{\theta}{dt}\right) = \frac{d^2x_W}{dt^2} - l\sin(\theta)\left(\frac{d\theta}{dt}\right) + l\cos\theta\left(\frac{d^2\theta}{dt^2}\right) \tag{2.8}$$

Podstawiając równanie (2.1.1) do zależności (2.1.1) otrzymuje się:

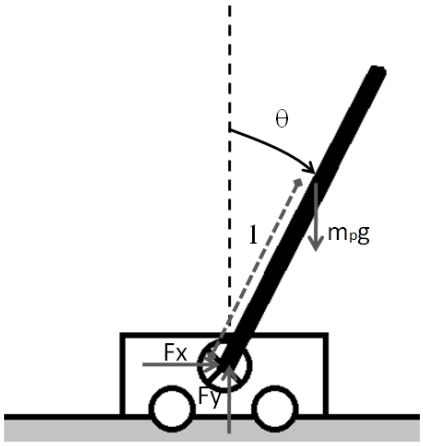
$$F_x = m_P\ddot{x}_W - m_P l\ddot{\theta}^2\sin(\theta) + m_P l\ddot{\theta}\cos(\theta) \tag{2.9}$$

Znając wszystkie siły działające na wózek można ostatecznie zapisać, że:

$$m_W\ddot{x}_W = \frac{M}{r} - b\dot{x}_W - m_P\ddot{x}_W + m_P l\ddot{\theta}^2\sin\theta - m_P l\ddot{\theta}\cos\theta \tag{2.10}$$

$$(m_W + m_P)\,\ddot{x}_W + b\dot{x}_W = \frac{M}{r} + m_P l\dot{\theta}^2\sin\theta - m_P l\ddot{\theta}\cos(\theta) \tag{2.11}$$

Położenie środka ciężkości pręta w osi poziomej x może zostać wyznaczone zgod-



Rysunek 2.3: Schemat wahadła - siły działające na wahadło

nie z (2.1.1) na podstawie położenia wózka. Aby wyznaczyć położenie wahadła w osi pionowej y, należy rozpatrzyć siły działające na pręt w tej osi zgodnie z rys. 2.3:

$$F_y - m_P g = m_P\ddot{y}_P \tag{2.12}$$

$$y_p = y_W + l\cos(\theta) \tag{2.13}$$

Pochodne położenia pionowego $\dot{y}_P$ oraz $\ddot{y}_P$ zostały wyznaczone analogicznie do $\dot{x}_P$ oraz $\ddot{x}_P$. Ponieważ wózek jest nieruchomy w osi y przyjęto, że $y_W = 0$, $\dot{y}_W = 0$, $\ddot{y}_W = 0$, z czego wynika, że:

$$\dot{y}_P = -l\dot{\theta}\sin(\theta) \tag{2.14}$$

$$\ddot{y}_P = -l\dot{\theta}^2\cos(\theta) - l\ddot{\theta}\sin(\theta) \tag{2.15}$$

Podstawiając zależność (2.1.1) w równaniu (2.1.1) uzyskuje się:

$$F_y = m_P g - m_P l\dot{\theta}^2\cos(\theta) - m_P l\ddot{\theta}\sin(\theta) \tag{2.16}$$

Ze względu, iż wahadło porusza się ruchem obrotowym, należy dla niego również rozpatrzeć równanie momentów:

$$I\ddot{\theta} = \sum \overline{M}, \tag{2.17}$$

gdzie: I jest momentem bezwładności wahadła, $\overline{M}$ to momenty działające na pręt wahadła prostopadle do osi obrotu.

Traktując pręt wahadła jako ciało sztywne można przyjąć, że momenty $\overline{M}$ powstają na skutek sił działających na ramieniu względem środka ciężkości wahadła:

$$\overline{M} = \overline{F} \times \overline{r}, \tag{2.18}$$

gdzie: $\overline{F}$ to siła działająca na wahadło, natomiast $\overline{r}$ to ramię, na którym działa ta siła.

Siły generujące momenty na wahadle to $\overline{F}_x$ oraz $\overline{F}_y$, ponieważ dla skupionej siły grawitacji wahadła ramię $r = 0$. Zatem można napisać, że:

$$I\ddot{\theta} = -F_x l\cos(\theta) + F_y l\sin(\theta) \tag{2.19}$$

Podstawiając równania (2.1.1) oraz (2.1.1) do równania (2.1.1) otrzymuje się:

$$\begin{aligned} I\ddot{\theta} &= -\left(m_P\ddot{x}_W - m_P l\ddot{\theta}^2\sin(\theta) + m_P l\ddot{\theta}\cos(\theta)\right) l\cos(\theta) + \\ &+ \left(m_P g - m_P l\dot{\theta}^2\cos(\theta) - m_P l\ddot{\theta}\sin(\theta)\right) l\sin(\theta) = \\ &= -m_P l\ddot{x}_W\cos(\theta) + m_P l\ddot{\theta}^2\sin(\theta)\cos(\theta) - m_P l^2\ddot{\theta}\cos(\theta)^2 + \\ &+ m_P g l\sin(\theta) - m_P l^2\dot{\theta}^2\cos(\theta)\sin(\theta) - m_P l^2\ddot{\theta}\sin(\theta)^2 = \\ &= -m_P l\ddot{x}_W\cos(\theta) - m_P l^2\ddot{\theta}\cos(\theta)^2 + m_P g l\sin(\theta) - m_P l^2\ddot{\theta}\sin(\theta)^2 = \\ &= -m_P l^2\ddot{\theta} + m_P g l\sin(\theta) - m_P l\ddot{x}_W\cos(\theta) \end{aligned} \tag{2.20}$$

$$\left(I + m_P l^2\right)\ddot{\theta} = m_P g l\sin(\theta) - m_P l\ddot{x}_W\cos(\theta) \tag{2.21}$$

Równania (2.1.1) oraz (2.1.1) umożliwiają w pełni opisać ruch wózka z wahadłem. Jednakże przedstawiają one opis obiektu w postaci uwikłanej. Można je przekształcić do następującego układu równań, umożliwiającego bezpośrednie obliczenie przyspieszeń $\ddot{x}_W$ i $\ddot{\theta}$:

$$\begin{cases} x_W = \dfrac{\dfrac{M}{r} - F_x - b\dot{x}_W}{M} \\ \dot{\theta} = \frac{-F_x l\cos(\theta) + F_y l\sin(\theta)}{I} \\ F_x = m_P\ddot{x}_W - m_P l\dot{\theta}^2\sin(\theta) + m_P l\ddot{\theta}\cos(\theta) \\ F_y = m_P g - m_P l\dot{\theta}^2\cos(\theta) - m_P l\ddot{\theta}\sin(\theta) \end{cases} \tag{2.22}$$

2.1.2 Model liniowy dla wybranego punktu pracy

Wyznaczony model nieliniowy obiektu pozwala w pełni opisać zachowanie i ruch wahadła. Jednakże zgodnie z założeniami z rozdziału 3 do wyznaczenia wybranego regulatora predykcyjnego wymagany jest model liniowy. Z tego powodu, wyznaczony model nieliniowy został zlinearyzowany w otoczeniu wybranego punktu pracy. Ponieważ celem regulatora jest utrzymanie wahadła w pozycji pionowej górnej, przyjęto ją jako punkt pracy, czyli $\theta = 0$.

2.2 Acrobot

2.3 Maszyny kroczące

Rozdział 3

Planowanie trajektorii ruchu robota - Janek

3.1 Planowanie trajektorii ruchu robota w przestrzeni wewnętrznej, zewnętrznej i kartezjańskiej

3.2 Koordynacja ruchu robotów w przestrzeni zadań

3.3 Planowanie ruchu jako przeszukiwanie

Rozdział 4

Sterowanie w przestrzeni przegubów - Kuba

4.1 Serwomechanizmy przegubów

4.2 Sterowanie w przestrzeni przegubów

4.2.1 Metoda odwrotnego modelu

Jedn? z najbardziej popularnych metod sterowania pozycyjnego w robotyce jest metoda odwrotnego modelu. Metoda modelu odwrotnego jest stosowan? w robotyce metod? linearyzacji i dekompozycji modelu matematycznego manipulatora, dzi?ki której mo?na sterowa? niezale?nie wszystkimi ramionami robota z wykorzystaniem technik sterowania obiektami liniowymi. Metoda odwrotnego modelu ma t? zalet? w porównaniu z innymi metodami linearyzacji (np. rozwini?cie w szereg Taylora) modelu, ?e kompensuje nieliniowo?ci w ca?ym zakresie zmian wspó?rz?dnych z??czowych, a nie tylko w pobli?u punktu, wokó? którego linearyzujemy model.

Na rys. **??** przedstawiony jest ogólny schemat sterowania robota (mechanizmu wielocz?onowego) z wykorzystanim modelu odwrotnego.

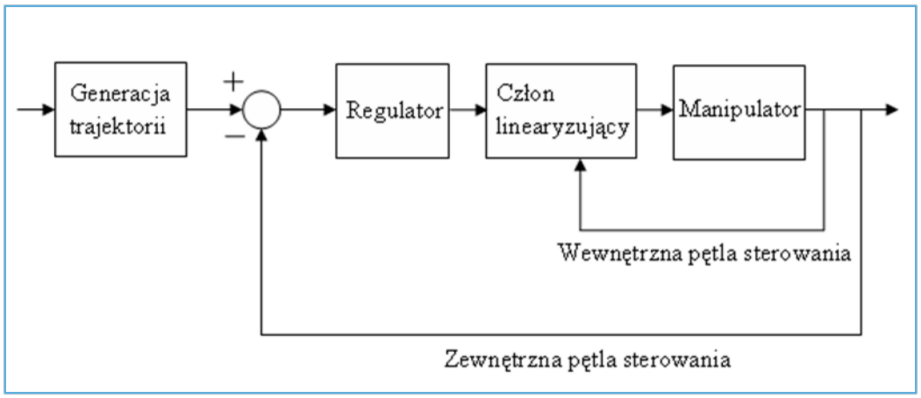
W strukturze sterowania mo?emy wyró?ni? dwie p?tle sprz??enia zwrotnego:

- P?tl? wewn?trzn? linearyzuj?c? i rozdzielaj?ca prawo sterowania na cze?? modelow? oraz sprz??eniow?
- P?tl? zewn?trzn? sprz??enia zwrotnego pozwalaj?c? na sterowanie manipulatora w ??dany sposób

Aby wyznaczy? odpowiednie sygna?y steruj?ce, nale?y wszystkie uk?ady zwi?zane z przegubami uniezale?ni? od siebie. Pozwoli to na uproszczenie problemu wyznaczenia sterowania do wyznaczenia sygna?ów steruj?cych dla sze?ciu odsprz?gni?tych uk?adów drugiego rz?du. Metody odsprz?gania przegubów robota opisano m.in. w [2], [8].

Dany jest model robota opisany równaniem 1.1.3. Przyjmuj?c posta? wektora sygna?ów steruj?cych:

$$\tau = P(x,t)\hat{\tau} + R(x,t), \tag{4.1}$$



Rysunek 4.1: Schemat sterowania robota z wykorzystaniem modelu odwrotnego

oraz przyjmuj?c, ?e:

$$P(x,t) = \hat{M}(x,t) \tag{4.2}$$

$$R(x,t) = \hat{V}(x,t) + \hat{G}(x,t) \tag{4.3}$$

mo?na rozdzieli? uk?ady zwi?zane z przegubami w równaniu (1.1.3). Macierze $\hat{M}(x,t)$, $\hat{V}(x,t)$,$\hat{G}(x,t)$ s? oszacowaniami (estymatami) macierzy $M(x,t)$, $V(x,t)$,$G(x,t)$, w równaniach Lagrange?a-Eulera, liczonymi wed?ug wzorów podanych w rozdziale 1, na podstawie oszacowanych parametrów robota.

Równanie opisuj?ce dynamik? robota mo?na zapisa?, wykorzystuj?c zale?no?ci (3.10) i (4.2.1), w postaci:

$$M(x,t)\ddot{q} + G(x) + V(x) = P(x,t)\hat{\tau} + R(x) \tag{4.4}$$

Zak?adaj?c, ?e oszacowane parametry robota maj? te same warto?ci co parametry rzeczywiste, równanie (4.2.1) upraszcza si? do postaci:

$$M(x,t)\ddot{q} = \hat{M}(x,t)\hat{\tau} \tag{4.5}$$

Zak?adaj?c, ?e macierz $M(x)$ jest macierz? nieosobliw?, oraz $M(x) = \hat{M}(x)$, w wyniku odsprz?gni?cia uzyskuje si? sze?? niezale?nych uk?adów opisanych równaniami drugiego rz?du, postaci:

$$\ddot{q} = \hat{\tau} \tag{4.6}$$

Uk?ad (4.2.1) mo?na opisa? równaniami w przestrzeni zmiennych stanu jako

$$\begin{cases} \dot{x_i} = Ax_i + B\hat{tau} \\ y_i = Cx_i \\ i = 1, \ldots, n \end{cases} \tag{4.7}$$

gdzie

$$x_i = \begin{bmatrix} q_i \\ \dot{q_i} \end{bmatrix}, A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, c = [1 \quad 0], \tag{4.8}$$

G?ównym wymaganiem przy stosowaniu metody odwrotnego modelu do sterowania jest konieczno?? zapewnienia dok?adnych oraz szybkich pomiarów wspó?rz?dnych z??czowych. Jest to silne za?o?enie, poniewa? pomiar parametrów robota jest zawsze obarczony jest pewnym b??dem to nie wszystkie parametry modelu s? dok?adnie okre?lone.

4.2.2 Transmitancja przegubu - Kuba

4.2.3 Regulator położenia przegubu - Kuba

4.2.4 Regulator momentu przegubu - Janek

4.2.5 Stabilność układu regulacji i wskaźniki jakości regulacji - Kuba

4.3 Sterowanie adaptacyjne - Kuba

Rozdział 5

Sterowanie w przestrzeni stanu - Kuba

Rozdział 6

Sterowanie siłowe - Janek

6.1 Sterowanie impedancyjne

6.2 Sterowanie siłowe

6.3 Sterowanie pozycyjno-siłowe

6.4 Modele tarcia

Rozdział 7

Sterowanie optymalne i sterowanie odporne - Kuba

7.1 Sterowanie o zmiennej strukturze mechanizmów wieloczłonowych

7.1.1 Wst?p do sterowania ?lizgowego

Niech $s(x)$ b?dzie funkcj?, nazywan? funkcj? prze??czaj?c?, opisan? zale?no?ci?:

$$s(x) = Hx(t) \tag{7.1}$$

gdzie: $H \in R^{m \times n}$ i jest pe?nego rz?du, n - liczba zmiennych stanu uk?adu, m - liczba sygna?ów steruj?cych.

Niech zbiór punktów w przestrzeni zmiennych stanu Ψ b?dzie opisany jako:

$$\Psi = \{x \in R^n : s(x) = 0\} \tag{7.2}$$

Zbiór Ψ nazywany jest powierzchni? ?lizgow?. Mo?na go te? nazwa? hiperpowierzchni?, lub quasipowierzchni? ?lizgow?, gdy? wymiar Ψ zale?y od liczby zmiennych stanu i liczby wej?? steruj?cych zgodnie z zale?no?ci?:

$$n_\Psi = n - m \tag{7.3}$$

Przyk?adowo dla dwóch zmiennych stanu i jednego wej?cia steruj?cego zbiór Ψ jest krzyw? o wymiarze 1, dla uk?adu o trzech zmiennych stanu i jednym wej?ciu steruj?cym jest p?aszczyzn?, dla uk?adu o sze?ciu zmiennych stanu i dwóch wej?ciach steruj?cych ma wymiar $n_\Psi = 4$

Przyjmuj?c, za [9]:

$$s(x) = [s_1(x), \ldots, s_m(x)]^T, H = [H_1, \ldots, H_m] \tag{7.4}$$

gdzie: $H_i = [h_{i1} \quad h_{i2} \ldots h_{in}], i = 1, \ldots, m$ równanie (4.5) mo?na przedstawi? w postaci i równa?:

$$s_i(x) = H_i x \quad \text{gdzie} i = 1, \ldots, m \tag{7.5}$$

Zbiór punktów $\Psi_i = \{x \in R^n : s_i(x) = 0\}$ opisuje powierzchni? Ψ_i zwan? i-t? sk?adow? powierzchni? ?lizgow?.

Metoda sterowania ?lizgowego polega na odpowiednim generowaniu sygna?u steruj?cego zale?nego od po?o?enia uk?adu w przestrzeni zmiennych stanu wzgl?dem powierzchni ?lizgowej Ψ. Uk?ad ze sterowaniem ?lizgowym jest tak zaprojektowany, aby jego trajektoria kierowa?a si? zawsze w stron? powierzchni ?lizgowej ?. W momencie gdy stan uk?adu j? osi?gnie, zaczyna "?lizga?" si? wzd?u? tej powierzchni, tzn. ca?y czas przechodzi z jednej jej strony na drug?. Uk?ad znajduje si? wtedy w tzw. trybie ?lizgowym. Zalet? tego trybu pracy uk?adu jest odporno?? na zak?ócenia i niedok?adno?ci w wyznaczaniu modelu obiektu sterowania.

Do momentu osi?gni?cia powierzchni Ψ uk?ad znajduje si? w tzw. trybie osi?gania. W tym trybie uk?ad nie posiada w?a?ciwo?ci charakterystycznych dla uk?adu w trybie ?lizgowym.

Dla uk?adu o wielu wej?ciach, stan uk?adu ?lizga si? po powierzchni ?lizgowej dopiero wtedy, gdy ?lizga si? po wszystkich sk?adowych powierzchniach ?lizgowych Ψi $(i = 1, \ldots, m)$. W sterowaniu ?lizgowym wykorzystywany jest sygna? steruj?cy o zmiennej strukturze, który kieruje trajektori? uk?adu zawsze w stron? ka?dej z tych powierzchni. Pozwala on na realizacj? obydwu trybów pracy uk?adu. Sterowanie jest przyjmowane w postaci:

$$u_i = \begin{cases} u_i^+(x) & \text{gdy} \quad s_i(x) > 0 \\ u_i^-(x) & \text{gdy} \quad s_i(x) < 0 \end{cases}, i = 1, \ldots, m \tag{7.6}$$

Projektowanie uk?adu sterowania ?lizgowego powinno sk?ada? si? z nast?puj?cych etapów:

- ETAP 1: Nale?y odpowiednio zaprojektowa? p?aszczyzn? ?lizgow? Ψ. Jej posta? okre?la si? poprzez dobór warto?ci w?asnych uk?adu w trybie ?lizgu.
- ETAP 2: Nale?y zaprojektowa? takie prawo sterowania, które pozwoli na zrealizowanie przez uk?ad trybu osi?gania i trybu ?lizgowego.

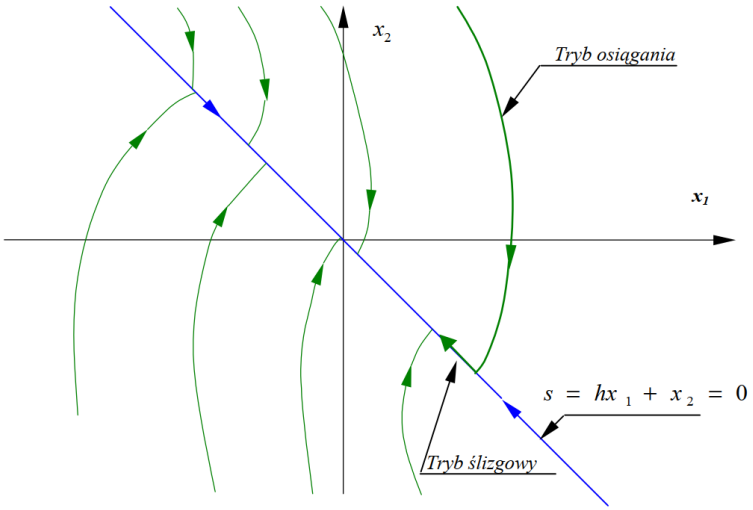
Trajektorie przyk?adowego uk?adu (oznaczone kolorem zielonym), o jednym wej?ciu steruj?cym i dwóch zmiennych stanu x_1, x_2 ze sterowaniem ?lizgowym przedstawiono na rysunku 4.1. Przyj?to ?e parametry obiektu s? zgodne z parametrami modelu, na podstawie którego wyznacza si? sygna? steruj?cy. Przyk?adowa funkcja prze??czaj?ca (oznaczona kolorem niebieskim) ma posta?:

$$s(x) = x_2 + hx_1, h > 0 \tag{7.7}$$

7.1.2 Projektowanie funkcji prze??czaj?cej dla mechanizmu wielocz?onowego

W przypadku rozwa?anego robota, równanie funkcji prze??czaj?cej proponowane jest w postaci [116], [50], [53]:

$$s(\tilde{s}) = H\tilde{x} = [\Lambda I][e_q \quad e_{\dot{q}}]^T = \Lambda e_q + e_{\dot{q}} \tag{7.8}$$



Rysunek 7.1: Przyk?adowe trajektorie w przestrzeni zmiennych stanu uk?adu ze sterowaniem ?lizgowym, z wyró?nieniem trybu osi?gania i trybu ?lizgowego.

gdzie: $\tilde{x} = [e_q \quad e_{\dot{q}}]^T$, $\Lambda = \mathrm{diag}[h_1 \ldots h_n]$ Powierzchnia ?lizgowa wyznaczona dla robota Ψ_{rob} opisywana jest przez zbiór punktów w przestrzeni zmiennych stanu robota takich, ?e:

$$\Psi_{\mathrm{rob}} = x \in R^{2n} : s(\tilde{x}) = 0 \tag{7.9}$$

Aby stan uk?adu zd??a? zawsze w kierunku powierzchni Ψ_{rob} , czyli znajdowa? si? w trybie osi?gania musz? zosta? spe?nione odpowiednie warunki osi?gania powierzchni ?lizgowej. Maj? one posta? [22]:

$$\dot{s}_i \tilde{x} s_i(\tilde{x}) < 0, \qquad , i = 1 \ldots n \tag{7.10}$$

gdzie:

$$s_i(\tilde{x}) = h_i e_{qi} + e_{\dot{q}i}, \qquad , i = 1 \ldots n \tag{7.11}$$

Warunek (4.15) mówi, ?e ka?da sk?adowa powierzchnia ?lizgowa Ψ_i , taka ?e:

$$\Psi_i = \tilde{x} \in R^{2n} : s_i(\tilde{x}) = 0 \tag{7.12}$$

musi by? przynajmniej lokalnie przyci?gaj?ca i w pewnym obszarze wokó? tej powierzchni trajektoria uk?adu ze sterowaniem ?lizgowym musi kierowa? si? zawsze w jej stron?.

Powierzchnia ?lizgowa Ψ_{rob} mo?e by? opisana jako zbiór punktów pochodz?cy z wzajemnego przecinania si? wszystkich powierzchni Ψ_i co mo?na wyrazi? zale?no?ci?:

$$\Psi_{\mathrm{rob}} = \Psi_1 \cap \Psi_2 \cap \ldots \Psi_n \tag{7.13}$$

7.1.3 Sterowanie równowa?ne

Aby uk?ad realizowa? tryb ?lizgowy, nale?y okre?li? odpowiednie sterowanie. Jednym z proponowanych rozwi?za? jest tzw.?sterowanie równowa?ne? [22]. Sterowanie równowa?ne wykorzystywane jest przy analizie uk?adu ze sterowaniem ?lizgowym. Sterowanie to jest sterowaniem ci?g?ym odpowiadaj?cym sterowaniu o zmiennej strukturze, które wymusza ruch stanu uk?adu po powierzchni ?lizgowej Ψ_{rob}, w momencie gdy stan uk?adu j? osi?gnie.

Podczas poni?szych wyprowadze? przyj?to, ?e parametry modelu obiektu sterowania, na podstawie którego oblicza si? sygna? steruj?cy, s? takie same jak parametry sterowanego obiektu.

Gdy stan uk?adu znajduje si? na powierzchni Ψ_{rob} i zachodzi idealny ruch ?lizgowy, spe?nione s? nast?puj?ce zale?no?ci [22]:

$$s(\tilde{x}) = H\tilde{x} = 0 \tag{7.14}$$

$$\dot{s}(\tilde{x}) = H\dot{\tilde{x}} = 0, \qquad t \geqslant t_s \tag{7.15}$$

gdzie t_s to czas, po którym uk?ad znajduje si? w trybie ?lizgu.

W przypadku modelu robota opisanego równaniem (4.1) mo?na wyznaczy? sterowanie równowa?ne korzystaj?c z równania (4.13) i oznacze? (4.2) i (4.3). Przyjmuj?c ?e w trybie ?lizgu spe?niona jest zale?no?? (4.19), mo?na napisa? ?e:

$$\dot{s}(\tilde{x}) = H\dot{\tilde{x}} = \Lambda\dot{e}_q + \dot{e}_{\dot{q}} = \Lambda(\dot{q} - \dot{q}_k) + \hat{F}_2(x) + \hat{E}(x_1)\tau - \ddot{(q)}_k = 0 \tag{7.16}$$

Z powy?szego równania uzyskuje si? sterowanie równowa?ne w postaci:

$$\tau_{eq} = -\left(\hat{E}(x_1)\right)^{-1}\left(\hat{F}_2(x) + \Lambda(x_2 - \dot{q}_k) - \ddot{q}_k\right) \tag{7.17}$$

Je?li stan opisuj?cy dynamik? robota znajdzie si? na powierzchni ?lizgowej Ψ_{rob}, to stan uk?adu pod wp?ywem sterowania równowa?nego (4.21) powinien ??lizga?? si? wzd?u? tej powierzchni. Równania opisuj?ce dynamik? robota w przestrzeni zmiennych stanu, w którym zastosowano sterowanie równowa?ne s? nast?puj?ce:

$$\dot{x} = \begin{bmatrix} x_2 \\ F_2(x) \end{bmatrix} + \begin{bmatrix} 0 \\ E(x_1) \end{bmatrix}\tau_{eq} = \begin{bmatrix} x_2 \\ F_2(x) \end{bmatrix} + \begin{bmatrix} x_2 \\ E(x_1)\left[\left(\hat{E}(x_1)\right)^{-1}\left(-\hat{F}_2(x) - \Lambda(x_2 - \dot{q}_k) + \ddot{q}_k\right)\right] \end{bmatrix} \tag{7.18}$$

W przypadku, gdy parametry modelu robota odpowiadaj? parametrom rzeczywistego robota mo?na napisa?, ?e:

$$\hat{F}_2(x) = F_2(x), \hat{E}(x_1) = E(x_1) \tag{7.19}$$

i równanie (4.22) upraszcza si? do postaci:

$$\dot{x} = \begin{bmatrix} x_2 \\ -\Lambda(x_2 - \dot{q}_k) + \ddot{q}_k \end{bmatrix} \tag{7.20}$$

Zazwyczaj w po?o?eniu zadanym przyjmuje si?, ?e: $\dot{q}_k = 0$, $\ddot{q}_k = 0$ i uk?ad opisany jest zale?no?ci?

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & I \\ 0 & -\Lambda \end{bmatrix} \\ y &= [I \quad 0]x \end{aligned} \tag{7.21}$$

Równanie (4.25) uk?adu w trybie ?lizgu, w którym dzia?a sterowanie równowa?ne opisuje n odsprz?gni?tych uk?adów, które s? zwi?zane z poszczególnymi przegubami robota.

Ka?dy z uk?adów opisany jest zale?no?ci?:

$$\begin{aligned} \dot{x}_i &= A_i x_i \\ y &= [1 \quad 0] x_i \end{aligned} \tag{7.22}$$

gdzie: $x_i = [x_{1i} \quad x_{2i}]^T$,$A_i = \begin{bmatrix} 0 & 1 \\ 0 & -h_i \end{bmatrix}$ Równanie charakterystyczne ka?dego z uk?adów w trybie ?lizgu ma posta?:

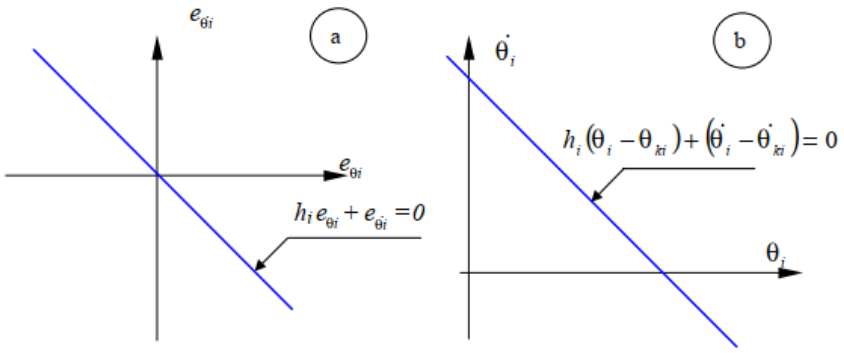
$$\det[Iz - Ai] = z(z + h_i) \tag{7.23}$$

W trybie ?lizgu równanie charakterystyczne i-tego uk?adu posiada dwa pierwiastki:

$$z_{i1} = 0, z_{i2} = -h_i \tag{7.24}$$

Wybieraj?c warto?ci $h_i > 0$ uk?ady opisane zale?no?ci? (4.26) s? stabilne.

Wygl?d funkcji prze??czaj?cej dla i-tego przegubu pokazano na rysunku 4.2.



Rysunek 7.2: Wygl?d funkcji $s_i(\tilde{x} = 0$, w i-tym uk?adzie wspó?rz?dnych stanu, jako funkcji (a) b??dów (4.2) i (4.3) i (b) wspó?rz?dnych stanu (po?o?enie k?towe, pr?dko?? k?towa), przy za?o?eniu, ?e $\dot{q}_{ki} > 0$, $q_{ki} > 0$

7.1.4 Algorytm sterowania ?lizgowego

Przyjmuj?c przedstawione za?o?enia mo?na wyznaczy? sygna?y steruj?ce w ka?dym z przegubów robota. Powinny one spowodowa? osi?gni?cie przez stan opisuj?cy uk?ad powierzchni ?lizgowej ?rob (tryb osi?gania), a nast?pnie ruch ?lizgowy po trajektorii znajduj?cej si? na tej powierzchni do zadanego punktu ?k , (?"k ,?""k tryb ?lizgowy). Algorytm wyznaczania sygna?ów steruj?cych w danej chwili t jest nast?puj?cy:

Algorytm 4.1. 1. Odczyta? z obiektu warto?ci po?o?e? k?towych i pr?dko?ci k?towych ?( ) ( ) t , we ?" t wszystkich przegubach. 2. Poda? zadane warto?ci

po?o?e?, pr?dko?ci i przyspiesze? k?towych w ka?dym z przegubów: ?( ) ( ) ( ) t k , . 3. Obliczy? warto?ci funkcji prze??czaj?cej s( ) x korzystaj?c z zale?no?ci (4.20). 4. Obliczy? macierze i wektory w równaniach Lagrange?a-Eulera na podstawie oszacowanych parametrów robota: M? ( ) ( ) ( ) x1 , V? x , G? x1 , F?2[ ] [ ] x , E? x1 . 5. Znaj?c warto?ci funkcji prze??czaj?cej s( ) x (okre?laj?cej po?o?enie uk?adu wzgl?dem powierzchni ?rob) okre?li? warto?ci elementów wektora sygna?u steruj?cego ? . W tym celu wykorzystuje si? ró?ne struktury sygna?ów steruj?cych. W pracy zbadane zostan? dwie struktury, pierwsz? jest sterowanie prze??czaj?ce (Fu, Gonzales, Lee,1987, Hung, Gao, 1993), którego opis i symulacje zawarte b?d? w rozdziale 4.8.1, drug? jest uzupe?nione sterowanie równowa?ne (Hung, Gao, 1993, Edwards, Spurgeon, 1998) którego opis i symulacje zawarte b?d? w rozdziale 4.8.2. 6. Zastosowa? w uk?adzie sygna? steruj?cy ? wyznaczony w kroku 5.

Uproszczony schemat blokowy robota z generatorem trajektorii i uk?adem sterowania wykorzystuj?cym algorytm sterowania ?lizgowego pokazano na rysunku 4.7

7.2 Sterowanie predykcyjne mechanizmów wieloczłonowych

7.3 Sterowanie LQR/LQG mechanizmów wieloczłonowych

Rozdział 8

Metody sztucznej inteligencji w sterowaniu mechanizmów wieloczłonowych - Janek+Kuba

8.1 Sieci neuronowe do modelowania mechanizmów wieloczłonowych

Sztuczne sieci neuronowe są jedną z popularnych metod stosowanych do aproksymacji funkcji nieliniowych. Inspiracją do ich opracowania były badania struktur biologicznych oraz sztucznej inteligencji. Budowa oraz sposób działania sztucznych sieci neuronowych pozwoliły na wykorzystanie ich w wielu dziedzinach niezwiązanych bezpośrednio z biologią m.in. w naukach technicznych, fizyce i ekonomii.

8.1.1 Model neuronu

Podstawowym elementem sieci neuronowej jest pojedyncza struktura nazywana **neuronem**, który składa się z elementu sumującego wagowo sygnały wejściowe, tzw. sumatora, oraz elementu przetwarzającego sygnał wyjściowy sumatora. Ogólny model matematyczny neuronu opiera się na klasycznym modelu *McCullocha - Pittsa* [55].

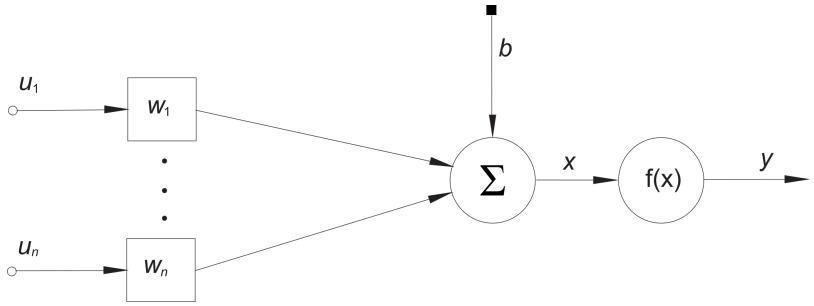
Model matematyczny pojedynczego neuronu jest opisany następującą zależnością

$$\begin{aligned} x &= b + \sum_{i=1}^{n} w_i u_i \\ y &= f(x) \end{aligned} \tag{8.1}$$

gdzie x - sygnał wyjściowy sumatora, u_i - i-ty sygnał wejściowy neuronu, b -

przesunięcie sygnału x, n - liczba sygnałów wejściowych neuronu, w_i - waga i-tego sygnału wejściowego neuronu, y - sygnał wyjściowy neuronu, $f(x)$ - funkcja aktywacji neuronu.

Na rys. 8.1 przedstawiono strukturę modelu neuronu (8.1)



Rysunek 8.1: Struktura modelu neuronu

Funkcją aktywacji pojedynczego neuronu może być funkcja liniowa - neuron liniowy, lub nieliniowa - neuron nieliniowy. W większości zastosowań przyjmuje się, że jest to funkcja ciągła i różniczkowalna. W praktyce najczęściej wykorzystywane są następujące funkcje aktywacji:

- **liniowa funkcja aktywacji (neuron liniowy)**

$$f(x) = a_{\mathrm{ln}} x \tag{8.2}$$

gdzie a_{ln} - współczynnik nachylenia funkcji.

Funkcja liniowa przyjmuje wartości w przedziale $(-\infty\ ,\ +\infty)$ (rys. 8.2.a), a jej pochodna względem x ma postać

$$f'_x(x) = \frac{\partial f(x)}{\partial x} = a_{\mathrm{ln}} \tag{8.3}$$

- **sigmoidalna funkcja aktywacji (neuron nieliniowy)**

Funkcja sigmoidalna może mieć postać unipolarną

$$f(x) = \frac{1}{1 + e^{-a_{\mathrm{su}} x}} \tag{8.4}$$

gdzie a_{su} - parametr funkcji sigmoidalnej unipolarnej.

Funkcja sigmoidalna unipolarna przyjmuje wartości w przedziale (0,1) (rys. 8.2.b), a jej pochodna względem x ma postać

$$f'_x(x) = \frac{\partial f(x)}{\partial x} = a_{\mathrm{su}} f(x)(1 - f(x)) \tag{8.5}$$

Funkcja sigmoidalna może mieć również postać bipolarną

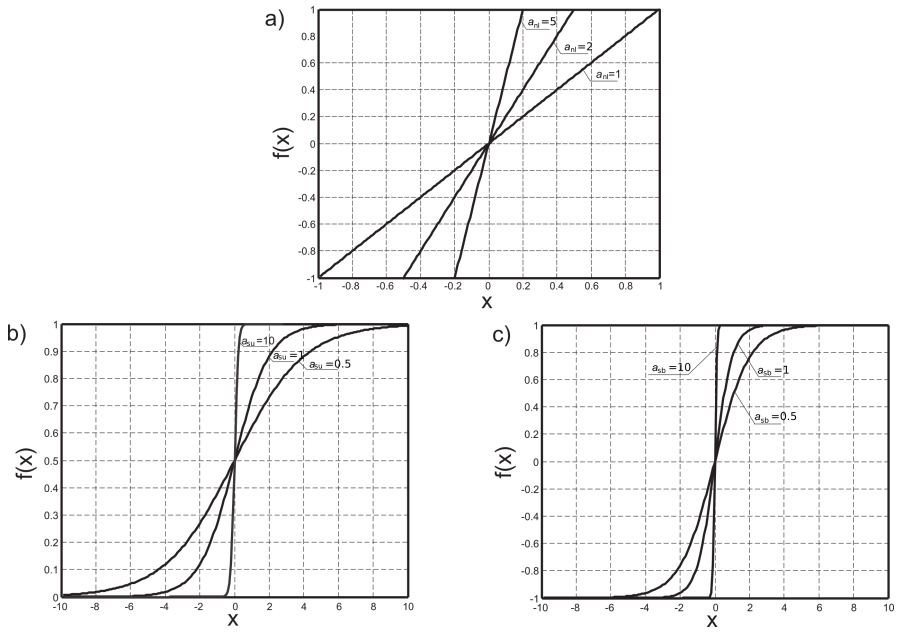
$$f(x) = tanh(a_{\mathrm{sb}} x) = \frac{e^{a_{\mathrm{sb}} x} - e^{-a_{\mathrm{sb}} x}}{e^{a_{\mathrm{sb}} x} + e^{-a_{\mathrm{sb}} x}} \tag{8.6}$$

gdzie a_{sb} - parametr funkcji sigmoidalnej bipolarnej.

Funkcja sigmoidalna bipolarna przyjmuje wartości w przedziale (-1,1) (rys. 8.2.c), a jej pochodna względem x ma postać

$$f'_x(x) = \frac{\partial f(x)}{\partial x} = tanh'(a_{\mathrm{sb}}x) = a_{\mathrm{sb}}(1 - f^2(x)) \tag{8.7}$$

Na rys. 8.2 przedstawiono wykresy funkcji aktywacji (8.2), (8.4), (8.6).



Rysunek 8.2: Wykresy funkcji aktywacji neuronu: a) funkcja liniowa (8.2), b) funkcja sigmoidalna unipolarna (8.4), c) funkcja sigmoidalna bipolarna (8.6)

8.1.2 Modele sztucznych sieci neuronowych

8.1.3 Sieć jednowarstwowa jednokierunkowa

Sieć jednowarstwowa jednokierunkowa (tzw. perceptron jednowarstwowy) posiada n sygnałów wejściowych $u = [u_i]_{n\times 1}$ oraz m neuronów, które nie są ze sobą połączone. Z każdym neuronem związane są: wagi $w_{j,i}$ sygnałów wejściowych, sygnał wyjściowy sumatora x_j, przesunięcie b_j sygnału x_j, funkcja aktywacji $f_j(x_j)$, oraz sygnał wyjściowy y_j. Tak więc perceptron jednowarstwowy posiada m sygnałów wyjściowych $y = [y_j]_{m\times 1}$.

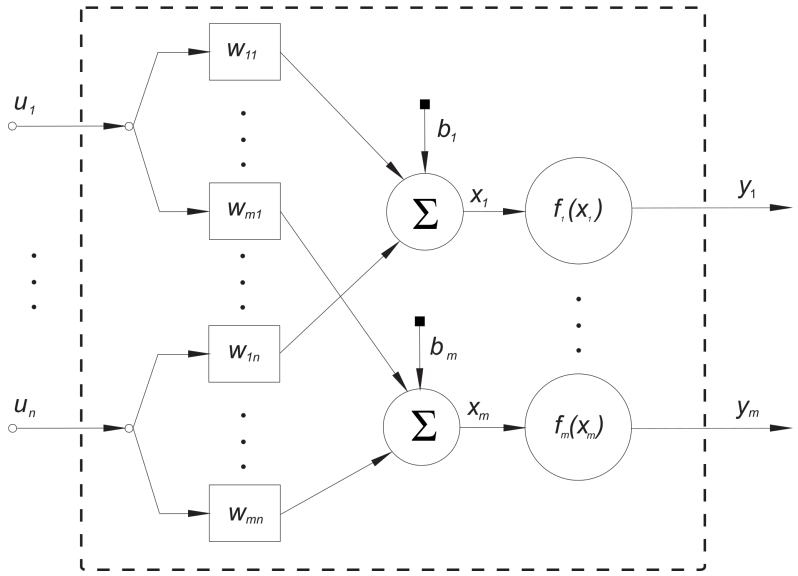
Przetwarzanie danych zachodzące w perceptronie jednowarstwowym opisane jest następująco

$$\begin{aligned} x &= b + Wu \\ y &= f(x) \end{aligned} \tag{8.8}$$

gdzie $u = [u_i]_{n\times 1}$ - wektor sygnałów wejściowych sieci, $x = [x_j]_{m\times 1}$ - wektor sygnałów wyjściowych sumatorów związanych z poszczególnymi neuronami,

$b = [b_j]_{m\times 1}$ - wektor przesunięć sygnałów wyjściowych sumatorów związanych z poszczególnymi neuronami, $W = [w_{j,i}]_{m\times n}$ - macierz wag sygnałów wejściowych sieci, $y = [y_j]_{m\times 1}$ - wektor sygnałów wyjściowych sieci, $f(x) = [f_j(x_j)]_{m\times 1}$ - wektor funkcji aktywacji neuronów.

Na rys. 8.3 przedstawiono strukturę jednowarstwowej jednokierunkowej sieci neuronowej.



Rysunek 8.3: Struktura jednowarstwowej jednokierunkowej sieci neuronowej

8.1.4 Sieć wielowarstwowa jednokierunkowa

Najczęściej wykorzystywanym modelem sztucznej sieci neuronowej jest sieć wielowarstwowa jednokierunkowa (ang. *feedforward neural network*), która ma n sygnałów wejściowych i m sygnałów wyjściowych. Sieć jednokierunkowa wielowarstwowa nazywana jest także perceptronem wielowarstwowym - MLP (ang. *multi layer perceptron*) i składa się z szeregowo połączonych warstw neuronów. Każda z warstw ma strukturę perceptronu jednowarstwowego (rys. 8.3). Warstwa nazywana **warstwą wyjściową** wyznacza sygnały wyjściowe z sieci neuronowej. Warstwy, których sygnały wyjściowe nie są sygnałami wyjściowymi sieci nazywane są **warstwami ukrytymi**. Przepływ sygnałów we wszystkich warstwach jest jednokierunkowy, od wejścia do wyjścia. W sieci wielowarstwowej jednokierunkowej istnieje przynajmniej jedna warstwa ukryta.

W przypadku sieci neuronowej L warstwowej, każda warstwa posiada $n^{(i)}$ sygnałów wejściowych oraz $m^{(i)}$ neuronów i $m^{(i)}$ sygnałów wyjściowych, gdzie i oznacza numer warstwy. W przypadku najczęściej stosowanej sieci zupełnej każdy sygnał wejściowy do warstwy połączony jest z każdym neuronem wagowo, a każdy z sygnałów wyjściowych z poprzedniej warstwy jest sygnałem wejściowym

do kolejnej warstwy

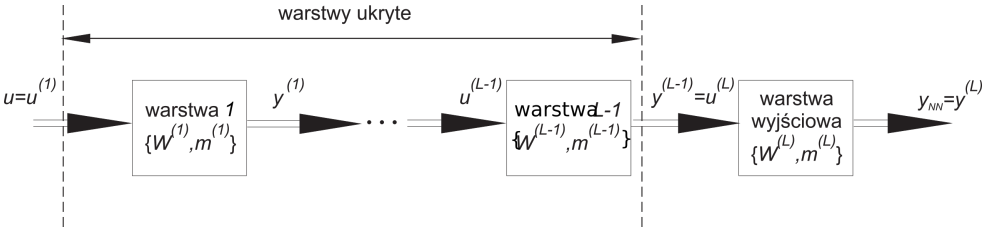
$$\begin{aligned} &u^{(1)} = [u_j^{(1)}]_{n^{(1)}\times 1} = u,\ y^{(1)} = [y_j^{(1)}]_{m^{(1)}\times 1},\ n^{(1)} = n \\ &u^{(i)} = [u_j^{(i)}]_{n^{(i)}\times 1} = y^{(i-1)},\ y^{(i)} = [y_j^{(i)}]_{m^{(i)}\times 1},\ n^{(i)} = m^{(i-1)},\ i = 2,\ldots,L \end{aligned} \tag{8.9}$$

gdzie $u^{(1)}$ - wektor sygnałów wejściowych do sieci, $u^{(i)}$ - wektor sygnałów wejściowych do warstwy i, $y^{(i)}$ - wektor sygnałów wyjściowych z warstwy i.

Sygnałami wyjściowymi z sieci neuronowej są sygnały wyjściowe z ostatniej warstwy

$$y_{\mathrm{NN}} = [y_{\mathrm{NN}i}]_{m^{(L)}\times 1} = y^{(L)},\ \text{gdzie } m^{(L)} = m \tag{8.10}$$

Model wielowarstwowej jednokierunkowej sztucznej sieci neuronowej, składającej się z połączonych szeregowo L sieci jednowarstwowych jednokierunkowych, został przedstawiony na rys. 8.4.



Rysunek 8.4: Struktura wielowarstwowej jednokierunkowej sieci neuronowej

Przetwarzanie danych zachodzące w warstwie i sieci neuronowej wielowarstwowej jednokierunkowej zgodnie z przyjętymi funkcjami aktywacji, opisane jest następująco

$$\begin{aligned} x^{(i)} &= b^{(i)} + W^{(i)}u^{(i)} \\ y^{(i)} &= f^{(i)}(x^{(i)}) \end{aligned},\ i = 1,\ldots,L \tag{8.11}$$

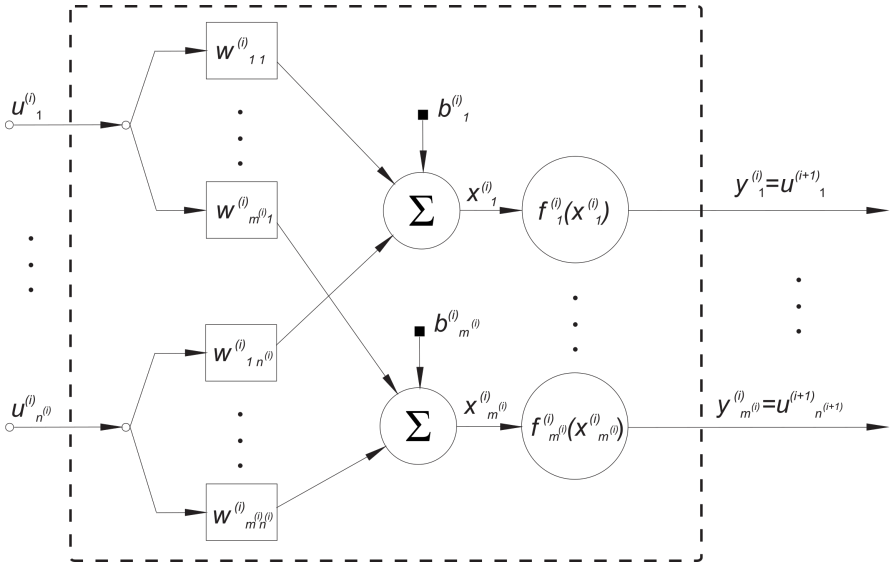
gdzie $u^{(i)} = [u_j^{(i)}]_{n^{(i)}\times 1}$ - wektor sygnałów wejściowych do warstwy, $x^{(i)} = [x_j^{(i)}]_{m^{(i)}\times 1}$ - wektor sygnałów wyjściowych sumatorów związanych z poszczególnymi neuronami warstwy, $y^{(i)} = [y_j^{(i)}]_{m^{(i)}\times 1}$ - wektor sygnałów wyjściowych z warstwy, $W^{(i)} = [w_{j,k}^{(i)}]_{m^{(i)}\times n^{(i)}}$ - macierz wag warstwy, $b^{(i)} = [b_j^{(i)}]_{m^{(i)}\times 1}$ - wektor przesunięć sygnałów wyjściowych sumatorów związanych z poszczególnymi neuronami warstwy, $f^{(i)}(x^{(i)}) = [f_k^{(i)}(x_k^{(i)})]_{m^{(i)}\times 1}$ - wektor funkcji aktywacji neuronów warstwy.

Struktura i-tej warstwy sieci neuronowej, opisana równaniami (8.11), została przedstawiona na rys. 8.5. Jest ona analogiczna do struktury przedstawionej na rys. (8.3).

Wykorzystując przedstawiony model sztucznej sieci neuronowej można zaprojektować sieć neuronową realizującą następujące przekształcenie

$$y_{\mathrm{NN}} = F_{\mathrm{NN}}(u), \qquad F_{\mathrm{NN}} : R^n \to R^m, \qquad y_{\mathrm{NN}} \in R^m, u \in R^n \tag{8.12}$$

gdzie y_{NN} - wektor sygnałów wyjściowych (8.10) sieci neuronowej, u - wektor sygnałów wejściowych sieci neuronowej.



Rysunek 8.5: Struktura i-tej warstwy sztucznej sieci neuronowej

8.1.5 Aproksymacja funkcji za pomocą sieci neuronowej

Sieć neuronowa może być wykorzystana do aproksymacji nieznanej funkcji wielowymiarowej postaci

$$y = F(u), \qquad F : R^n \to R^m, \qquad y \in R^m, u \in R^n \tag{8.13}$$

gdzie u - wektor argumentów funkcji $F(u)$, y - wektor wartości funkcji $F(u)$.

Aproksymacją w przypadku sieci neuronowej będziemy nazywali funkcję $y_{\mathrm{NN}} = F_{\mathrm{NN}}(u)$ realizowaną przez sieć neuronową (8.12), przybliżającą nieznaną funkcję (8.13).

Przyjmijmy, że istnieje zbiór N próbek $s(k)$ wartości funkcji (8.13) w postaci par uporządkowanych

$$s(k) = \{u(k), y(k) : y(k) = F[u(k)]\}, \qquad k = 1, \ldots, N \tag{8.14}$$

gdzie k - numer próbki.

Jeżeli dla każdej próbki ze zbioru (8.14) zostanie spełniony warunek

$$|y_{\mathrm{NN}}(k) - y(k)| < \varepsilon, \tag{8.15}$$

to można przyjąć, że sieć aproksymuje funkcję (8.13) z dokładnością $\varepsilon = [\varepsilon_i]_{m\times 1}$.

Zadanie aproksymacji funkcji (8.13) za pomocą sieci neuronowej można przedstawić następująco. Przyjmuje się funkcję celu

$$E = f[e(1), \ldots, e(N)] \tag{8.16}$$

gdzie $e(k)$ - wektor błędu aproksymacji dla k-tej próbki ma postać

$$e(k) = [e_i(k)]_{m\times 1} = y_{\mathrm{NN}}(k) - y(k) \tag{8.17}$$

Na ogół, podczas trenowania sieci neuronowej, minimalizowana jest funkcji celu (8.16), którą można zapisać jako funkcję wektora wszystkich wag sieci

$$E = E(W) \tag{8.18}$$

gdzie

$$W = [b_1^{(1)}, w_{1,1}^{(1)}, \ldots, w_{m^{(1)},n^{(1)}}^{(1)}, b_1^{(2)}, \ldots, w_{m^{(L)},n^{(L)}}^{(L)}]^T = [w_i]_{n_\mathrm{W}\times 1} = [w_1, \ldots, w_{n_\mathrm{W}}]^T \tag{8.19}$$

oraz n_W - liczba wszystkich wag i przesunięć sieci neuronowej, która dla L warstwowej jednokierunkowej sieci zupełnej wynosi

$$n_\mathrm{W} = \sum_{l=1}^{L} m^{(l)}(n^{(l)} + 1) \tag{8.20}$$

Podczas trenowania należy tak dobrać wartości wag sieci neuronowej W, żeby funkcja celu (8.16) osiągała minimum.

Jeśli funkcja celu osiąga zbyt duże wartości, to można zaprojektować inną sieć np. o zmienionej liczbie neuronów lub warstw i podjąć próbę aproksymacji funkcji (8.13) za pomocą nowej sieci.

Często formułuje się zadanie aproksymacji, w tym dobór struktury sieci i wag tak, że wymaga się, aby funkcja celu osiągała wartość mniejszą niż pewna przyjęta wartość zadana E_0.

Jako funkcja celu często wykorzystywana jest średnia suma kwadratów błędów pomiędzy wyjściem z sieci, a wartością aproksymowanej funkcji po N próbkach, tzw. średni błąd kwadratowy - MSE (ang. *mean squared error,*)

$$E_\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{m} e_i(k)^2 \tag{8.21}$$

Inne funkcje celu wykorzystywane podczas aproksymacji to np. średni błąd bezwzględny - MAE (ang. *mean absolute error*), pierwiastek średniego błędu kwadratowego - RMSE (ang. *root mean squared error*), średni błąd kwadratowy z regularyzacją - MSEREG (ang. *mean squared error with regularization*) [20, 30, 21, 31].

Zaletą sieci neuronowej jest możliwość realizacji sieci aproksymującej z zadaną dokładnością prawie każdą funkcję ciągłą określoną na zbiorze zwartym [72, 73, 74]. Należy jednak zwrócić uwagę, że dokładność aproksymacji za pomocą sieci neuronowej zależy od liczby neuronów, rodzaju funkcji aktywacji neuronów w warstwach ukrytych, liczby próbek uczących, algorytmu doboru wag oraz architektury sieci. Każde z tych zagadnień wymaga rozważenia podczas projektowania sztucznej sieci neuronowej dla konkretnego zastosowania.

8.2 Trenowanie sztucznej sieci neuronowej

Parametrami sieci neuronowej jest zbiór wag W (8.19), który należy wyznaczyć tak, aby funkcja (8.12) realizowana przez sieć jak najlepiej aproksymowała nieznaną funkcję (8.13). Proces doboru wag W nazywany jest **procesem uczenia** lub **procesem trenowania** sieci neuronowej.

Proces doboru neuronów i liczby warstw będziemy nazywali **procesem projektowania** sieci neuronowej.

8.2.1 Adaptacja wag sieci neuronowej

Do minimalizacji funkcji celu (8.18) można wykorzystać iteracyjne algorytmy rozwiązywania zadania programowania nieliniowego bez ograniczeń [14]. W przypadku sieci neuronowej jednokierunkowej wielowarstwowej zadanie to polega na znalezieniu wektora wag $\hat{W}$ spełniającego zależność

$$E(\hat{W}) = \min_{W} E(W) \tag{8.22}$$

W iteracyjnych algorytmach, w każdej iteracji, wartości wag są modyfikowane wg zależności

$$W_{(i+1)} = W_{(i)} + \eta_{(i)} p_{(i)} \tag{8.23}$$

gdzie $W_{(i)}$ - macierz wag w i-tej iteracji, $p_{(i)} \in R^{n_\mathrm{W}}$ - kierunek poprawy, $\eta_{(i)} \in R$ - współczynnik kroku.

Kierunek poprawy oraz współczynnik kroku są wyznaczane tak, aby

$$E(W_{(i)} + \eta_{(i)} p_{(i)}) < E(W_{(i)}) \tag{8.24}$$

Minimum funkcji celu charakteryzuje się zerową wartością gradientu funkcji celu.

W celu określenia w kolejnych iteracjach kierunku minimalizacji funkcji celu oraz współczynnika kroku, w procesie trenowania sieci neuronowej, wykorzystuje się tzw. *metody kierunków poprawy* np. metodę największego spadku, gradientów sprzężonych, *Levenberga - Marquardta* [20, 30, 31, 14].

Ponieważ liczba iteracji może być bardzo duża, w stosowanych algorytmach wyznaczania wag sieci neuronowej określa się warunki zatrzymania działania algorytmu - tzw. warunki stopu. Zwykle podstawowymi wielkościami branymi pod uwagę jest założona minimalna wartość funkcji celu E_0 oraz założona maksymalna liczba iteracji treningowych $i_{\max}$.

Ogólną postać gradientowego algorytmu trenowania sieci neuronowej można przedstawić następująco:

Algorytm 4.1. Gradientowy algorytm trenowania sieci neuronowej

1. *Przyjąć żądaną wartość funkcji celu sieci neuronowej* E_0,
   przyjąć maksymalną liczbę iteracji treningowych $i_{\max}$,
   przyjąć wartość początkową optymalizowanego wektora wag $W_{(0)}$,
   przyjąć numer iteracji początkowej $i = 0$.

2. *Przyjąć* $i = i + 1$

3. *Sprawdzić warunek*
   $$i \geqslant i_{\max} \tag{8.25}$$
   Jeśli warunek (8.25) jest spełniony zakończyć trenowanie sieci.

4. *Obliczyć wartość funkcji celu sieci neuronowej* $E(W_{(i)})$ *a następnie sprawdzić warunek:*
   $$E(W_{(i)}) \leqslant E_0 \tag{8.26}$$
   Jeżeli warunek (8.26) jest spełniony zakończyć trenowanie sieci.

5. *Wyznaczyć kierunek poprawy* $p_{(i)}$ *zgodnie z wybraną metodą optymalizacyjną.*

6. *Wyznaczyć wartość współczynnika kroku $\eta_{(i)}$ zgodnie z wybraną metodą optymalizacyjną.*
7. *Określić wartości elementów wektora wag $W_{(i+1)}$ wg zależności*

$$W_{(i+1)} = W_{(i)} + \eta_{(i)} p_{(i)} \tag{8.27}$$

8. *Wrócić do punktu 2.*

□

Zaawansowane optymalizacyjne algorytmy gradientowe wykorzystują rozwinięcie funkcji celu (8.18) sieci neuronowej w szereg *Taylora* w punkcie $W_{(i)}$, ze szczególnym uwzględnieniem trzech pierwszych wyrazów

$$\begin{aligned} E(W) &= E(W_{(i)}) + [\tfrac{\partial E}{\partial W}|_{W=W(i)}]^T (W - W_{(i)}) + \tfrac{1}{2}(W - W_{(i)})^T [\tfrac{\partial^2 E}{\partial W^2}|_{W=W(i)}](W - W_{(i)}) + R(W_{(i)}) \\ &= E(W_{(i)}) + \nabla E(W_{(i)})^T (W - W_{(i)}) + \tfrac{1}{2}(W - W_{(i)})^T H(W_{(i)})(W - W_{(i)}) + R(W_{(i)}) \end{aligned} \tag{8.28}$$

gdzie $R(W_{(i)})$ - reszta rozwinięcia funkcji celu (8.18) w szereg *Taylora*, $\nabla E(W_{(i)})$ - wektor gradientu funkcji celu (8.18) postaci

$$\nabla E(W) = \left.\frac{\partial E}{\partial W}\right|_{W=W(i)} = \left.\begin{bmatrix} \frac{\partial E(W)}{\partial w_1} \\ \vdots \\ \frac{\partial E(W)}{\partial w_{n_\mathrm{W}}} \end{bmatrix}_{n_\mathrm{W} \times n_\mathrm{W}}\right|_{W=W(i)} \tag{8.29}$$

oraz $H(W_{(i)})$ - *hesjan*, macierz drugich pochodnych funkcji celu (8.18) względem wag sieci

$$H(W_{(i)}) = \left.\frac{\partial^2 E}{\partial^2 W}\right|_{W=W(i)} = \left.\begin{bmatrix} \frac{\partial^2 E(W)}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E(W)}{\partial w_1 \partial w_{n_\mathrm{W}}} \\ \vdots & & \vdots \\ \frac{\partial^2 E(W)}{\partial w_{n_\mathrm{W}} \partial w_1} & \cdots & \frac{\partial^2 E(W)}{\partial w_{n_\mathrm{W}} \partial w_{n_\mathrm{W}}} \end{bmatrix}_{n_\mathrm{W} \times n_\mathrm{W}}\right|_{W=W(i)} \tag{8.30}$$

W przypadku funkcji celu MSE (8.21) hesjan (8.30) jest symetryczny.

Metoda wstecznej propagacji błędu

Gradientowe metody optymalizacji stosowane do doboru wag W sieci neuronowej w procesie trenowania wymagają znajomości wartości gradientu funkcji celu względem każdej z wag sieci (8.29). Do jego wyznaczenia w sieci wielowarstwowej wykorzystywany jest algorytm **wstecznej propagacji błędu** (ang. *error backpropagation*) [58]. W algorytmie wstecznej propagacji błędu wartości szukanych elementów wektora gradientu $\nabla E(W)$ (8.29) obliczane są w kierunku od warstwy wyjściowej do warstwy wejściowej sieci neuronowej.

W metodzie wstecznej propagacji błędu wymaga się, aby funkcja aktywacji każdego neuronu była różniczkowalna i znana była wartość jej pochodnej względem argumentu. Algorytm wstecznej propagacji błędów można zapisać następująco [20, 30].

Algorytm 4.2. Wyznaczanie wektora gradientu funkcji celu wielowarstwowej jednokierunkowej sieci neuronowej metodą wstecznej propagacji błędu

1. *Dana jest próbka* $s = \{u, y\}$*, liczba sygnałów wejściowych do sieci neuronowej* $n^{(1)} = n$*,liczba warstw sieci* L*, liczba neuronów w warstwach sieci* $m^{(i)}, i = 1, \ldots, L$*, sygnały wejściowe do sieci* $u_j^{(1)} = u_j, j = 1, \ldots, n$*, macierze wag i wektory przesunięć sygnałów wyjściowych sumatorów warstw sieci* $W^{(i)}, b^{(i)}, i = 1, \ldots, L$*, liczba sygnałów wyjściowych z sieci* $m^{(L)} = m$*, sygnały wyjściowe z sieci* $y_{\mathrm{NN}} = [y_{\mathrm{NN}i}]_{m^{(L)} \times 1}$

2. *Obliczyć* $x^{(l)}, y^{(l)}$ *wg (8.11) dla* $l = 1, \ldots, L$*.*

3. *Przyjąć* l*=*L*.*

4. *Obliczyć wartość pochodnej funkcji celu sieci neuronowej względem sygnałów wyjściowych z sieci*

$$E'_{y_{\mathrm{NN}i}} = \frac{\partial E}{\partial y_{\mathrm{NN}i}}, \qquad i = 1, \ldots, m^{(L)} \tag{8.31}$$

5. *Wyznaczyć wartość pochodnej funkcji celu sieci neuronowej względem sygnałów* $x_i^{(l)}$
   Jeśli $l = L$

$$E'_{x_i^{(L)}} = \frac{\partial E}{\partial x_i^{(L)}} = \frac{\partial f_i^{(L)}(x_i^{(L)})}{\partial x_i^{(L)}} E'_{y_{\mathrm{NN}i}}, \qquad i = 1, \ldots, m^{(L)}, \tag{8.32}$$

   jeśli $l < L$ wyznaczyć

$$E'_{x_i^{(l)}} = \frac{\partial E}{\partial x_i^{(l)}} = \frac{\partial f_i^{(l)}(x_i^{(l)})}{\partial x_i^{(l)}} \sum_{j=1}^{m^{(l+1)}} [w_{j,i}^{(l+1)} E'_{x_j^{(l+1)}}], \qquad i = 1, \ldots, m^{(l)} \tag{8.33}$$

6. *Wyznaczyć pochodne funkcji celu sieci neuronowej względem wag i przesunięć sygnałów wyjściowych sumatorów warstwy* l *sieci, będące szukanymi elementami wektora gradientu* $\nabla E(W)$ *(8.29)*

$$E'_{w_{j,i}^{(l)}} = \frac{\partial E}{\partial w_{j,i}^{(l)}} = u_i^{(l)} E'_{x_j^{(l)}}, \qquad i = 1, \ldots, n^{(l)}, j = 1, \ldots, m^{(l)} \tag{8.34}$$

$$E'_{b_j^{(l)}} = \frac{\partial E}{\partial b_j^{(l)}} = E'_{x_j^{(l)}}, \qquad j = 1, \ldots, m^{(l)} \tag{8.35}$$

7. *Przyjąć* $l = l - 1$*, jeśli* $l > 0$ *przejść do punktu 4, w przeciwnym przypadku zakończyć algorytm.*

□

Odpowiedni wybór funkcji celu sieci neuronowej powinien pozwalać na łatwe wyznaczenie gradientu (8.31) tej funkcji względem każdego z sygnałów wyjściowych sieci. W aplikacjach sieci neuronowych często jest stosowana funkcja celu E_{MSE} (8.21), dla której

$$E'_{y_{\mathrm{NN}}} = \left[E'_{y_{\mathrm{NN}i}}\right]_{m^{(L)} \times 1} = E'_{(\mathrm{MSE})y_{\mathrm{NN}}} = \left[\sum_{k=1}^{N} \frac{\partial E_{\mathrm{MSE}}}{\partial y_{\mathrm{NN}i}(k)}\right]_{m^{(L)} \times 1} = \left[\frac{2}{N} \sum_{k=1}^{N} e_i(k)\right]_{m^{(L)} \times 1} \tag{8.36}$$

Metoda największego spadku

W metodzie największego spadku przyjmowany jest kierunek poprawy $p_{(i)}$ postaci

$$p_{(i)} = -\nabla E(W_{(i)}), \tag{8.37}$$

natomiast wartość współczynnika kroku powinna spełniać zależność

$$\eta_{(i)} > 0 \tag{8.38}$$

W najprostszym przypadku metody największego spadku przyjmowana jest odpowiednio mała stała wartość $\eta_{(i)}$, jednak może to spowodować, że funkcja celu sieci neuronowej będzie wolno zbiegała do optymalnego rozwiązania. W celu poprawy zbieżności i szybszego znalezienia rozwiązania bliskiego optymalnemu, do wyznaczenia w kolejnych iteracjach wartości współczynnika kroku $\eta_{(i)}$, wykorzystuje się algorytmy minimalizacji funkcji celu wzdłuż wyznaczonego kierunku $p_{(i)}$ np. metodą złotego podziału, interpolacji kwadratowej, *Charalambousa*, *Brenta* [14, 31]. Stosowane są także inne modyfikacje algorytmu największego spadku, do których należą m.in. metoda największego spadku z momentem [30, 29, 31], metoda QUICKPROP (ang. *quick backpropagation*) [85], metoda RPROP (ang. *resilent backpropagation*) [84].

Metoda gradientów sprzężonych

Metoda gradientów sprzężonych, w przypadku nieliniowego układu równań (np. sieci neuronowej), jest modyfikacją metody gradientów sprzężonych dla układu równań liniowych [37]. Wykorzystuje model kwadratowy funkcji celu sieci neuronowej $E(W)$ w otoczeniu wartości $W_{(i)}$, który otrzymuje się w wyniku obcięcia rozwinięcia (8.28) do trzech pierwszych wyrazów.

W pierwszej iteracji algorytmu gradientów sprzężonych obliczany jest kierunek poprawy

$$p_{(0)} = -\nabla E(W_{(0)}) \tag{8.39}$$

W kolejnych iteracjach określane są wartości elementów wektora $W_{(i)}$

$$W_{(i)} = W_{(i-1)} + \eta_{(i-1)} p_{(i-1)} \tag{8.40}$$

oraz kierunek poprawy w iteracji i

$$p_{(i)} = -\nabla E(W_{(i)}) + \beta_{(i-1)} p_{(i-1)} \tag{8.41}$$

gdzie $\beta_{(i-1)}$ - współczynnik obliczany wg odpowiednich formuł, do których należą m.in. formuła *Fletchera-Reevesa* lub formuła *Polaka-Ribierè'a* [14, 37].

Do wyznaczenia w kolejnych iteracjach wartości współczynnika kroku $\eta_{(i)}$ wykorzystuje się algorytmy minimalizacji funkcji celu wzdłuż wyznaczonego kierunku $p_{(i)}$ np. metoda *Charalambous'a*, metoda *Brendt'a*, metoda złotego podziału [31].

Algorytm gradientów sprzężonych charakteryzuje się skończoną liczbą iteracji n_{iter} [37, 10], która jest ograniczona przez liczbę modyfikowanych parametrów (wag sieci) n_{W}

$$n_{\mathrm{iter}} \leqslant n_{\mathrm{W}} \tag{8.42}$$

Metoda *Levenberga - Marquardta*

Metoda *Levenberga - Marquardta* [56, 57, 30, 29, 31] wykorzystuje do modyfikacji wag sieci neuronowej trzy pierwsze wyrazy i oszacowanie reszty rozwinięcia (8.28) funkcji celu. W metodzie *Levenberga - Marquardta* nie jest wyznaczany kierunek poprawy i współczynnik kroku, tak jak ma to miejsce w metodach największego spadku, lub gradientów sprzężonych. Metoda *Levenberga - Marquardta* jest szczególnym przypadkiem *metody zmiennej metryki* [14] gdzie zmiany wag W sieci neuronowej w i-tej iteracji wyznacza się wg zależności

$$W_{(i+1)} = W_{(i)} - H(W_{(i)})^{-1}\nabla E(W_{(i)}) \tag{8.43}$$

Przybliżenia hesjanu i gradientu funkcji celu względem wag sieci, stosowane w metodzie *Levenberga - Marquardta*, są następujące

$$H(W_{(i)}) \cong [K(W_{(i)})^T K(W_{(i)}) + R(W_{(i)})] \tag{8.44}$$

oraz

$$\nabla E(W_{(i)}) \cong K(W_{(i)})^T e_{(i)} \tag{8.45}$$

gdzie $K(W_{(i)})$ - *jakobian*, macierz pochodnych elementów wektora błędu $e_{(i)}$ (8.17) względem każdej z wag sieci

$$K(W_{(i)}) = \left.\frac{\partial e}{\partial W}\right|_{W=W(i)} = \left[\begin{array}{cccc} \frac{\partial e_1}{\partial w_1} & \frac{\partial e_1}{\partial w_2} & \cdots & \frac{\partial e_1}{\partial w_{n_W}} \\ \frac{\partial e_2}{\partial w_1} & \frac{\partial e_2}{\partial w_2} & \cdots & \frac{\partial e_2}{\partial w_{n_W}} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_m}{\partial w_1} & \frac{\partial e_m}{\partial w_2} & \cdots & \frac{\partial e_m}{\partial w_{n_W}} \end{array}\right]_{n_W \times n_W}\Bigg|_{W=W(i)} \tag{8.46}$$

W metodzie *Levenberga - Marquardta* w celu uzyskania dodatnio określonego przybliżonego hesjanu (8.44) stosuje się następujące przybliżenie reszty $R(W_{(i)})$ rozwinięcia (8.28)

$$R(W_{(i)}) \cong \mu_{(i)}\mathbf{I}_{n_W \times n_W} \tag{8.47}$$

gdzie $\mu_{(i)}$ - czynnik regularyzacyjny, tzw. *parametr Marquardta*.

Tak więc, w metodzie *Levenberga - Marquardta*, zmiany wag sieci neuronowej w i-tej iteracji treningowej wyznacza się wg zależności

$$W_{(i+1)} = W_{(i)} - [K(W_{(i)})^T K(W_{(i)}) + \mu_{(i)}\mathbf{I}]^{-1} K(W_{(i)})^T e_{(i)} \tag{8.48}$$

Minimalizacja funkcji celu polega na zmianach wartości czynnika regularyzacyjnego $\mu_{(i)}$ i wag sieci neuronowej podczas kolejnych iteracji w zależności od wartości funkcji celu [29].

Sposoby prezentacji próbek i modyfikacji wag podczas trenowania sieci neuronowej

W przypadku trenowania sieci neuronowej rozróżnia się dwie podstawowe metody prezentacji próbek ze zbioru treningowego i modyfikacji wag sieci [11].

Pierwsza metoda nazywana jest **trenowaniem przyrostowym**, inkrementalnym (ang. *incremental training*) lub trenowaniem w trybie *on-line*. W tym przypadku wagi modyfikowane są po prezentacji próbki danych $s(k) = \{u(k), y(k)\}$.

Wykorzystywana jest wtedy funkcja celu sieci neuronowej liczona dla jednej lub kilku próbek (np. $N = 1$ lub $N = 5$ w (8.21)).

Druga metoda doboru wag sieci neuronowej nazywana jest **trenowaniem kumulacyjnym**, wsadowym (ang. *batch training*) lub trenowaniem w trybie *off-line*. W tym przypadku, w jednym cyklu tzw. epoce (ang. *epoch*) prezentowane są wszystkie próbki danych ze zbioru treningowego, a następnie obliczana jest funkcja celu sieci neuronowej dla wszystkich próbek.

8.2.2 Zdolność uogólniania sieci neuronowej

W trenowaniu kumulacyjnym sieci neuronowej, do trenowania wykorzystuje się tzw. *zbiór treningowy* próbek $s(k)$ (8.14) wartości funkcji (8.13) - ozn. Z_{train}. Na jego podstawie można także określić jak dobrze układ estymuje dane wykorzystane podczas trenowania sieci. Wartość funkcji celu dla zbioru treningowego jest oznaczana E_{train} i nazywana *błędem aproksymacji sieci neuronowej*.

Wytrenowana sieć neuronowa powinna charakteryzować się dobrą zdolnością generalizacji (uogólniania), czyli estymacji wartości funkcji $y(k)$ (8.13), które nie były znane w procesie trenowania sieci, na podstawie sygnałów wejściowych $u(k)$. Do oceny uogólniania można wykorzystać *zbiór testowy* Z_{test} próbek $s(k)$ o innych wartościach funkcji (8.13) niż zbiór treningowy. Wartość funkcji celu dla zbioru testowego jest nazywana *błędem uogólniania sieci neuronowej* lub *błędem generalizacji sieci neuronowej* i oznaczana E_{test}.

Na ogół, gdy liczba próbek w zbiorze testowym i w zbiorze treningowym jest taka sama, błąd uogólniania E_{test} jest większy niż błąd aproksymacji E_{train}.

Własności aproksymacji i uogólniania sieci neuronowej ściśle zależą od jej budowy (liczby wag, neuronów i warstw) oraz liczby próbek w zbiorze treningowym. Jednymi z warunków uzyskania dobrego uogólniania jest odpowiednia liczebność i różnorodność zbioru treningowego. Liczebność zbioru treningowego można określić wykorzystując miarę *Vapnika - Chervonenkisa* VCdim [42] dla danej struktury sieci neuronowej. Ponieważ jest to trudne zagadnienie, to w praktycznych rozwiązaniach stosuje się oszacowanie wartości miary VCdim. Granice miary VCdim zostały określone dla różnych struktur sieci neuronowych [30, 29]. W praktyce podczas trenowania sieci neuronowej należy przyjąć zbiór treningowy, w którym liczba próbek będzie przewyższała wartość VCdim.

Sieci neuronowe wykorzystywane w pracy są sieciami jednokierunkowymi wielowarstwowymi. W ich przypadku na podstawie znajomości granic miary VCdim [29] przyjmuje się oszacowanie wartości VCdim jako liczbę wszystkich wag sieci (n_{W}), natomiast minimalną liczbę próbek zbioru treningowego N_{train} jako

$$N_{\text{train}} \approx 2\text{VCdim} \approx 2n_{\text{W}} \tag{8.49}$$

Warunkiem uzyskania sieci dobrze uogólniającej, czyli dobrze aproksymującej dane testowe, jest więc przyjęcie liczby próbek w zbiorze treningowym przynajmniej dwukrotnie większej niż liczba wszystkich wag sieci

$$N_{\text{train}} > 2n_{\text{W}} \tag{8.50}$$

8.3 Regulator stanu mechanizmu wieloczłoowego oparty o sieci neuronowe

8.4 Logika rozmyta w sterowaniu mechanizmów wieloczlonowych

8.5 Sterowanie ze wzmocnieniem mechanizmów wieloczłonowych

Bibliografia

[1] R.S. Hartenberg and J. Denavit. A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics*, 77:215–221, 1955.

[2] K. S. Fu, R.C. Gonzalez, and C. S. G. Lee. *Robotics: control, sensing, vision, and inteligence.* McGraw-Hill Book Company, USA, 1987.

[3] W. Hejmo. *Sterowanie robotami i manipulatorami przemysłowymi: metody i modele matematyczne.* Wydawnictwo Politechniki Krakowskiej, Kraków, 1997.

[4] V. D. Tourassis and C. P. Neuman. Properties and structure of dynamic robot models for control engeenering applications. *Mechanism and Machine Theory*, 20(1):27–40, 1985.

[5] K. Świder. *Automatyczne Generowanie i Redukcja Symbolicznych Modeli Dynamiki Robotów - Rozprawa Doktorska.* Wydział Elektroniki, Politechnika Warszawska, 1992.

[6] A. Morecki and J. Knapczyk. *Podstawy Robotyki. Teoria i Elementy Manipulatorów i Robotów (praca zbiorowa).* WNT Warszawa, 1993.

[7] F. L. Lewis, S. Jagannathan, and A. Yesildirek. *Neural Network Control of Robot Manipulators and Nonlinear Systems.* Taylor and Francis, 1999.

[8] Craig J. J. *Wprowadzenie do robotyki, mechanika i sterowanie.* WNT, Warszawa, 1995.

[9] Hung J. Y., Gao W., and Hung J. C. Variable structure control: A survey. *IEEE Transactions on Industrial Electronics*, 40:2–22, 1993.

[10] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer Science+Business Media, LLC., New York, 2006.

[11] J. C. Principe, N. R. Euliano, and W. C. Lefebvre. *Neural and Adaptive Systems: Fundamentals Through Simulations.* Wiley, New York, 2000.

[12] Ch. Hansen. *Rank Deficient and Discrete Ill Posed Problems, Numerical Aspects of Linear Inversion.* SIAM, 1998.

[13] V. B. Glasko. *Inverse Problems of Mathematical Physics.* American Institute of Physics, 1988.

[14] W. Findeisen, J. Szymanowski, and A. Wierzbicki. *Teoria i Metody Obliczeniowe Optymalizacji*. PWN, Warszawa, 1980.

[15] G. H. Golub and C. F. Van Loan. *Matrix Computations. 3rd ed.* Johns Hopkins University Press, Baltimore, 1996.

[16] T. Kaczorek. *Wektory i Macierze w Automatyce i Elektrotechnice*. WNT, Warszawa, 1998.

[17] L. Ljung. *System Identification: Theory for the User*. Upper Saddle River, NJ, USA:Prentice-Hall, 1999.

[18] K. Kozłowski. *Mathematical Dynamic Robot Models and Identification of Their Parameters*. Technical University of Poznan Press, 1992.

[19] K. Kozłowski. *Modelling and Identification in Robotics*. Springer Verlag, Berlin, 1998.

[20] T. Masters. *Sieci Neuronowe w Praktyce, Programowanie w Języku C++*. WNT, Warszawa, 1996.

[21] N. Jankowski. *Ontogeniczne Sieci Neuronowe, o Sieciach Zmieniających Swoją Strukturę*. Akademicka Oficyna Wydawnicza EXIT, Warszawa, 2003.

[22] C. Edwards and S. K. Spurgeon. *Sliding Mode Control: Theory and Applications*. Taylor and Francis, 1998.

[23] T. Söderström and P. Stoica. *Identyfikacja Systemów*. Wydawnictwo Naukowe PWN, Warszawa, 1997.

[24] J. Korbicz, A. Obuchowicz, and D. Uciński. *Sztuczne Sieci Neuronowe: Postawy i Zastosowania*. Akademicka Oficyna Wydawnicza PLJ, Warszawa, 1994.

[25] K. S. Fu, R. C. Gonzalez, and C. S. G. Lee. *Robotics: Control, Sensing, Vision, and Inteligence*. McGraw-Hill Book Company, 1987.

[26] M. J. Giergiel, Z. Hendzel, and W. Żylski. *Modelowanie i Sterowanie Mobilnych Robotów Kołowych*. PWN Warszawa, 2002.

[27] J. J. Craig. *Wprowadzenie do Robotyki*. WNT Warszawa, 1995.

[28] P. I. Corke. *Matlab Robotics Toolbox - Release 5*. CSIRO, Australia, 1999.

[29] S. Osowski. *Sieci Neuronowe Do Przetwarzania Informacji*. OWPW, Warszawa, 2000.

[30] S. Osowski. *Sieci Neuronowe w Ujęciu Algorytmicznym*. WNT, Warszawa, 1996.

[31] H. Demuth, M. Beale, and M. Hagan. *Neural Network Toolbox For use with MATLAB: User's Guide*. The Mathworks, 2006.

[32] M. Norrlöf. *Iterative Learning Control: Analysis, Design, and Experiments - PhD Thesis*. Linköping Studies in Science and Technology, Linköping, Sweden, 2000.

[33] P. Wawrzyński. *Intensive Reinforcement Learning - Rozprawa Doktorska*. Wydział Elektroniki, Politechnika Warszawska, 2005.

[34] J. Możaryn. *Projektowanie układu sterowania ślizgowego dla robota PUMA 560 - Praca Dyplomowa Magisterska*. Politechnika Warszawska, 2001.

[35] Norrlöf M. Modelling of industrial robots. Technical Report LiTH-ISY-R-2208, Linköping University, Linköping, Sweden, 1999.

[36] W. C. Davidon. Variable metric method for minimization. Technical Report ANL-5990, A.E.C. Research and Development Report, 1959.

[37] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, 1994.

[38] A. Neumayer. Solving ill-conditioned and singular linear systems, a tutorial on regularization. *SIAM Review*, 40:636–666, 1998.

[39] H. J. Wesley, A. V. Gribok, A. M. Urmanov, and M. A. Buckner. Selection of multiple regularization parameters in local ridge regression using evolutionary algorithms and prediction risk optimization. *Inverse Problems in Engineering*, 11(3):215–227, 2003.

[40] A. N. Tikhonov. Solution of incorrectly formulated problems and regularization method. *Soviet. Math. Dokl.*, 4:1035–1038, 1963.

[41] C. R. Johnson. Positive definite matrices. *American Mathematical Monthly*, 77:259–264, 1970.

[42] V. Vapnik and A. Chervonenkis. On the uniform convergence of relative frequencies of events of their probabilities. *Theory of Probability and its Applications*, 16:264–280, 1971.

[43] V. Vapnik. Principle of risk minimization for learning theory. *NIPS4*, pages 831–838, 1992.

[44] B. Horne and D. Hush. Progress in supervised neural networks. *IEEE Signal Processing Magazine*, pages 8–39, 1993.

[45] K. S. Narendra and K. Parthasarathy. Identification and control of dynamical systems using neural networks. *IEEE Transactions on Neural Networks*, 1(1):4–27, 1990.

[46] W. Gao, Y. Wang, and A. Homaifa. Discrete-time variable structure control systems. *IEEE Transactions on Industrial Electronics*, 42(2):117–122, 1995.

[47] J.E. Kurek and M. B. Zaremba. Iterative learning control synthesis based on 2-d system theory. *IEEE Transactions on Automatic Control*, 38(1):121–124, 1993.

[48] J.E. Kurek. Iteracyjne uczące się sterowanie robota puma 560. *Pomiary Automatyka Kontrola*, 8:32–35, 1999.

[49] V. I. Utkin. Variable structure systems with sliding modes. *IEEE Transactions on Automatic Control*, 22:212–222, 1977.

[50] J.-J. E. Slotine and S. S. Sastry. Tracking control of nonlinear systems using sliding surfaces with aplication to robot manipulators. *International Journal of Control*, 38(2):465–492, 1983.

[51] H. Mayeda, K. Yoshida, and K. Osuka. Base parameters of manipulator dynamic models. *IEEE Transactions on Robotics and Automation*, 6(3):312–321, 1990.

[52] J.-J. E. Slotine and C. de Wit. Sliding observers for robot manipulators. *Automatica*, 27(5):859–864, 1991.

[53] W. Gao and J.C. Hung. Variable structure control of nonliner systems: A new approach. *IEEE Transactions on Industrial Electronics*, 40(1):45–55, 1993.

[54] J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Journal of Applied Mechanics*, 23:215–221, 1955.

[55] W. S. McCulloch and W. H. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–119, 1943.

[56] K. Levenberg. A method for the solution of certain problems in least squares. *The Quarterly of Applied Mathematics*, 2:164–168, 1944.

[57] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.

[58] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representation by error propagation. *Parallel distributed processing: Explorations in the Microstructures of Cognition, Rumelhart, D.E., McClelland, (Eds.), MIT Press, Cambridge*, 1:318–362, 1986.

[59] C. G. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[60] R. Fletcher. A new approach to variable metric algorithms. *The Computer Journal*, 13:317–322, 1970.

[61] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computing*, 24:23–26, 1970.

[62] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computing*, 24:647–656, 1970.

[63] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *Computer Journal*, 6:163–168, 1963.

[64] R. Fletcher and C. M. Reeves. Function minimization by conjugate gradients. *Computer Journal*, 7:149–154, 1964.

[65] F. L. Lewis, K. Liu, and A. Yesildirek. Neural net robot controller with guaranteed tracking performance. *IEEE Transactions on Neural Networks*, 6(3):703–715, 1995.

[66] F. L. Lewis, K. Liu, and A. Yesildirek. Neural net robot controller: Structure and stability proofs. *Journal on Intelligent and Robotic Systems*, 12:277–299, 1995.

[67] F. L. Lewis, K. Liu, and A. Yesildirek. Multilayer neural net robot controller: Structure and stability proofs. *IEEE Transactions on Neural Networks*, 7:1–12, 1996.

[68] F. L. Lewis. Neural network control of robot manipulators. *IEEE Expert, special track on Intelligent Control*, 11(3):64–75, 1996.

[69] S. Jagannathan and F. L. Lewis. Multilayer discrete-time neural-net controller. *IEEE Transactions on Neural Networks*, 7(1):107–130, 1996.

[70] K.M.W Tang and V. D. Tourassis. Mathematical deficiencies of numericaly simplified dynamic robot models. *IEEE Transactions on Automatic Control*, 34(10):1109–1111, 1989.

[71] P. I. Corke. A symbolic and numeric procedure for manipulator rigid-body dynamic significance analysis and simplification. *Robotica*, 16:589–594, 1998.

[72] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(15):359–366, 1989.

[73] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems (MCSS)*, 2(4):303–314, 1989.

[74] V. Kurkova. Kolmogorov's theorem and multilayer networks. *Neural Networks*, 5(3):501–506, 1992.

[75] K. Kozłowski. Robot dynamics models in terms of generalized and quasicoordinates. a comparison. *Applied Mathematics and Computer Science*, (2):305–328, 1995.

[76] M. Gautier and W. Khalil. Exciting trajectories for the identification of base inertial parameters of robots. *International Journal of Robotic Research*, 11(4):362–375, 1992.

[77] N. Sadegh. A perceptron network for functional identification and control of nonlinear systems. *IEEE Transation on Neural Networks*, 4(6):982–988, 1993.

[78] M. Gautier and P. Poignet. Extended kalman filtering and weighted least squares dynamic identification of robot. *Control Engeenering Practice*, 9(12):1361–1372, 2001.

[79] M. Kawato, H.M. Miyamoto, T. Setoyama, and T. Suzuki. Feedback-error-learning neural network for trajectory control of a robotic manipulator. *Neural Networks*, 11:251–265, 1988.

[80] T. Furuhashi, S. Okuma, T. Ozaki, and T. Suzuki. Trajectory control of robotic manipulator using neural networks. *IEEE Transactions on Industrial Electronics*, 38(3):195–202, 1991.

[81] J.-J. E. Slotine and R.M. Sanner. Stable adaptive control of robot manipulators using neural networks. *Neural Computation*, 7(4):753–790, 1995.

[82] J.-J. E. Slotine and R.M. Sanner. Stable adaptive control and recursive identification using radial gaissian networks. *Proceedings of the 30th Conference on Decision and Control, Brighton, England*, pages 2116–2123, 1991.

[83] K.M.W. Tang and V. D. Tourassis. Systematic simplification of dynamic robot models. *Proceedings of Midwest Symposium on Ciruits and Systems, Syracuse, NY, USA*, pages 1031–1034, 1987.

[84] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. *Proceedings of the IEEE International Conference on Neural Networks, San Francisco, USA*, 1993.

[85] S.E. Fahlman. Faster learning variations on backpropagation: An empirical study. *Proceedings 1988 Connectionist Model Summer School, Morgan Kaufmann, Los Altos, USA*, pages 38–51, 1988.

[86] J. Możaryn, C. Wildner, and J. E. Kurek. Wyznaczanie parametrów modelu robota przemysłowego przy pomocy sieci neuronowych. *XIV Krajowa Konferencja Automatyki 2002, Zielona Góra, Polska*, pages 675–678, 2002.

[87] J. Możaryn and J. E. Kurek. Synthesis of sliding mode control of robot with neural network model. *12th IFAC IEEE Methods and Models in Automation and Robotics MMAR 2006, Międzyzdroje, Polska*, 2006.

[88] J. Możaryn and J. E. Kurek. Relative error indices for comparison of neural models of different robots. *8th International Conference, Mechatronics 2009, Luhacovice, Czech Republic,*, 2009.

[89] J. Możaryn and J. E. Kurek. Design of a neural network for an identification of a robot model with a positive definite inertia matrix. *Proc. 10th Int. Conf. on Artificial Intelligence and Soft Computing ICAISC 2010, Zakopane, Polska*, LNCS 6114/2010:321–328, 2010.

[90] J. Możaryn and J. E. Kurek. Sliding mode control of robot based on neural network model with positive definite inertia matrix. *20th International Conference on Artificial Neural Networks ICANN 2010, Thessaloniki, Greece*, LNCS 6353/2010:266–275, 2010.

[91] J. Możaryn and J. E. Kurek. Using tikhonov regularization to improve estimation of robot position based on uncertain robot model obtained by neural network. *Pomiary Automatyka Kontrola*, 3, 2009.

[92] J. Możaryn and J. E. Kurek. Calculation of model of the robot by neural network with robot joint distinction. *Seventh International Conference on Artificial Intelligence and Soft Computing 7th ICAISC, Zakopane, Polska*, LNAI 3070:792–797, 2004.

[93] J. Możaryn and J. E. Kurek. Neural network robot model with not inverted inertia matrix. *Methods and Models in Automation and Robotics MMAR 2004, Międzyzdroje, Polska*, pages 1021–1026, 2004.

[94] B. Armstrong. On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics. *International Journal of Robotic Research*, 8(6):28–48, 1989.

[95] J. Swevers, C. Ganseman, D.B. Tukel, J. de Schutter, and H. van Brussel. Optimal robot excitation and identification. *IEEE Transactions on Robotics and Automation*, 13(5):730–740, 1997.

[96] M.M. Olsen and H.G. Petersen. A new method for estimating parameters of a dynamic robot model. *IEEE Transactions od Robotics and Automation*, 17:95–100, 2001.

[97] J. Możaryn and J. E. Kurek. Improvement of inertia matrix in robot model identified with neural networks. *Methods and Models in Automation and Robotics MMAR 2007, Międzyzdroje, Polska*, pages 977–984, 2007.

[98] J. Możaryn and J. E. Kurek. Comparison of neural network robot models with not inverted and inverted inertia matrix. *15th International Conference on Artificial Neural Networks ICANN 2005, Warszawa, Polska*, LNCS 3697:417–422, 2005.

[99] J. Możaryn and J. E. Kurek. Design of decoupled sliding mode control for the puma 560 robot manipulator. *Third International Workshop on Robot motion and Control RoMoCo 2002, Bukowy Dworek, Polska*, pages 45–50, 2002.

[100] J. Możaryn and J. E. Kurek. Comparison of sliding mode control and decoupled sliding mode control of robot puma 560. *Methods and Models in Automation and Robotics MMAR 2003, Międzyzdroje, Polska*, pages 969–974, 2003.

[101] P. I. Corke and B. Armstrong-Helouvry. A search for consensus among model parameters reported for the puma 560 robot. *IEEE International Conference Robotics and Automation, San Diego, USA*, 2:1608–1613, 1994.

[102] D. Kostic, R. Hensen, B. de Jager, and M. Steinbuch. Modelling and identification of an rrr-robot. *40th IEEE International Conference on Decision and Control, Orlando, Florida, USA*, 2:1144–1149, 2001.

[103] M. Gautier. A comparison of filetered models for dynamic identification of robots. *35th IEEE Conference on Decision and Control, Kobe, Japan*, 1:875–880, 2001.

[104] K. Kozłowski. Identification of articulated body inertias and decoupled control of robots in terms of quasi-coordinates. *Proceedings of the 1996 IEEE International Conference on Robotics and Automation, Minneapolis, Minnesota*, 1:317–322, 1996.

[105] K. Kozłowski, J.K. Tar, I.J. Rudas, and J.F. Bitó. Non-conventional integration of the fundamental elements of soft computing and traditional methods in adaptive robot control. *Proceedings of the 2000 IEEE lnternational Conference on Robotics and Automation, San Francisco, CA, USA*, pages 3531–3536, 2000.

[106] P. Dutkiewicz, K. Kozłowski, and W. S. Wróblewski. Experimental identification of robot and load dynamic parameters. *Second lEEE Conference on Control Applications, Vancouver, B.C., Canada*, 2:767–776, 1993.

[107] M. Gautier and W. Khalil. On the identification of the inertial parameters of robots. *Proceedings IEEE Conference on Decision and Control, Austin, Texas, USA*, 3:2264–2269, 1988.

[108] B. Armstrong, O. Khatib, and J. Burdick. The explicit dynamic model and inertial parameters of the puma 560 arm. *Proceedings IEEE Conference on Robotics and Automation, San Francisco, CA, USA*, 1:510–518, 1986.

[109] J.E. Kurek. Neural net model of robot manipulator. *Neural Computation NC'98, Vienna, Austria*, pages 778–783, 1998.

[110] J.E. Kurek. Calculation of robot manipulator model using neural net. *European Control Conference ECC'99, Karlsruhe, Germany*, 1999.

[111] F.C. Sun and Z.Q. Sun. Stable sampled-data adaptive control of robot arms using neural networks. *Journal of Intelligent and Robotic Systems*, 20(2-4):131–155, 1997.

[112] J.S. Albus. A new approach to manipulator control: The cerebellar model articulation controller cmac. *ASME Journal of Dynamic Systems, Measurement and Control*, 97:220–227, 1975.

[113] R. Horowitz. Learning control of robot manipulators. *ASME Journal of Dynamic Systems, Measurement and Control*, 115:402–411, 1993.

[114] A. Lesewed and J.E. Kurek. Design of iterative learning control for simple robot based on neural network robot model. *Pomiary Automatyka Kontrola*, 3:205–208, 2009.

[115] M.M. Polycarpou and P.A. Ioannou. Identification and control of nonlinear systems using neural network models: Design and stability analysis. Technical Report 91-09-01, Departement of Electrical Egineering and Engineering Systems, Univeristy of Southern California, 1991.

[116] Young K. K. D. Controller design for a manipulator using theory of variable structure systems. *IEEE Trans. Sys. Man. And Cyb.*, SMC-8:101–109, 1978.