

# Sterowanie Mechanizmów Wieloczłonowych

Jakub Możaryn, Jan Klimaszewski

12 sierpnia 2018



# Spis treści

<b>1</b>	<b>Opis mechanizmów wielocłonowych - Janek+Kuba</b>	<b>5</b>
1.1	Model geometrii, kinematyki i dynamiki mechanizmów wielocłonowych. . . . .	5
1.1.1	Model geometrii - Janek . . . . .	5
1.1.2	Model kinematyki - Janek . . . . .	5
1.1.3	Model dynamiki - Kuba . . . . .	7
1.1.4	Właściwości strukturalne modelu dynamiki . . . . .	11
1.2	Opis mechanizmów wielocłonowych w przestrzeni stanu. . . . .	12
1.3	Identyfikacja parametrów kinematycznych i dynamicznych robotów. . . . .	13
1.4	Zastosowanie kwaternionów. . . . .	13
1.5	Otwarte i zamknięte łańcuchy kinematyczne. . . . .	13
1.6	Układy niedosterowane. . . . .	13
1.7	Układy nieholonomiczne. . . . .	14
1.8	Kolizje. . . . .	14
<b>2</b>	<b>Standardowe modele układów wielocłonowych - Janek</b>	<b>15</b>
2.1	Wahadło odwrócone na wózku - Kuba . . . . .	15
2.1.1	Model nieliniowy w oparciu o równania fizyczne . . . . .	15
2.1.2	Model liniowy dla wybranego punktu pracy . . . . .	19
2.2	Acrobot . . . . .	19
2.3	Maszyny kroczące . . . . .	19
<b>3</b>	<b>Planowanie trajektorii ruchu robota - Janek</b>	<b>21</b>
3.1	Planowanie trajektorii ruchu robota w przestrzeni wewnętrznej, zewnętrznej i kartezjańskiej . . . . .	21
3.2	Koordinacja ruchu robotów w przestrzeni zadań . . . . .	21
3.3	Planowanie ruchu jako przeszukiwanie . . . . .	21
<b>4</b>	<b>Sterowanie w przestrzeni przegubów - Kuba</b>	<b>23</b>
4.1	Serwomechanizmy przegubów . . . . .	23
4.2	Sterowanie w przestrzeni przegubów . . . . .	23
4.2.1	Metoda odwrotnego modelu . . . . .	23
4.2.2	Transmitancja przegubu . . . . .	25
4.2.3	Regulator położenia przegubu . . . . .	25
4.2.4	Regulator momentu przegubu . . . . .	25
4.2.5	stabilność układu regulacji i wskaźniki jakości regulacji . . . . .	25
4.3	Sterowanie adaptacyjne . . . . .	25

<b>5 Sterowanie w przestrzeni stanu - Kuba</b>	<b>27</b>
5.1 Regulator dla robota o wielu stopniach swobody . . . . .	27
5.2 Linearyzacja sprzężenia zwrotnego . . . . .	27
5.3 Dobór funkcji Lapunova dla potrzeb sterowania mechanizmów wielocłonowych . . . . .	27
5.4 Sterowne pozycyjne i sterowanie nadążne . . . . .	27
5.5 Projektowanie sterowania jako zadanie optymalizacji . . . . .	27
<b>6 Sterowanie siłowe - Janek</b>	<b>29</b>
6.1 Sterowanie impedancyjne . . . . .	29
6.2 Sterowanie siłowe . . . . .	29
6.3 Sterowanie pozycyjno-siłowe . . . . .	29
6.4 Modele tarcia . . . . .	29
<b>7 Sterowanie optymalne i sterowanie odporne - Kuba</b>	<b>31</b>
7.1 Sterowanie o zmiennej strukturze mechanizmów wielocłonowych	31
7.2 Sterowanie predykcyjne mechanizmów wielocłonowych . . . . .	31
7.3 Sterowanie LQR/LQG mechanizmów wielocłonowych . . . . .	31
<b>8 Metody sztucznej inteligencji w sterowaniu mechanizmów wielocłonowych - Janek+Kuba</b>	<b>33</b>
8.1 Sieci neuronowe do modelowania mechanizmów wielocłonowych	33
8.1.1 Model neuronu . . . . .	33
8.1.2 Modele sztucznych sieci neuronowych . . . . .	35
8.1.3 Sieć jednowarstwowa jednokierunkowa . . . . .	35
8.1.4 Sieć wielowarstwowa jednokierunkowa . . . . .	36
8.1.5 Aproksymacja funkcji za pomocą sieci neuronowej . . . . .	38
8.2 Trenowanie sztucznej sieci neuronowej . . . . .	39
8.2.1 Adaptacja wag sieci neuronowej . . . . .	40
8.2.2 Zdolność uogólniania sieci neuronowej . . . . .	45
8.3 Regulator stanu mechanizmu wielocłowego oparty o sieci neuronowe . . . . .	46
8.4 Logika rozmyta w sterowaniu mechanizmów wielocłonowych . . . . .	46
8.5 Sterowanie ze wzmocnieniem mechanizmów wielocłonowych . . . . .	46

# Rozdział 1

## Opis mechanizmów wieloczłonowych - Janek+Kuba

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

### 1.1 Model geometrii, kinematyki i dynamiki mechanizmów wieloczłonowych.

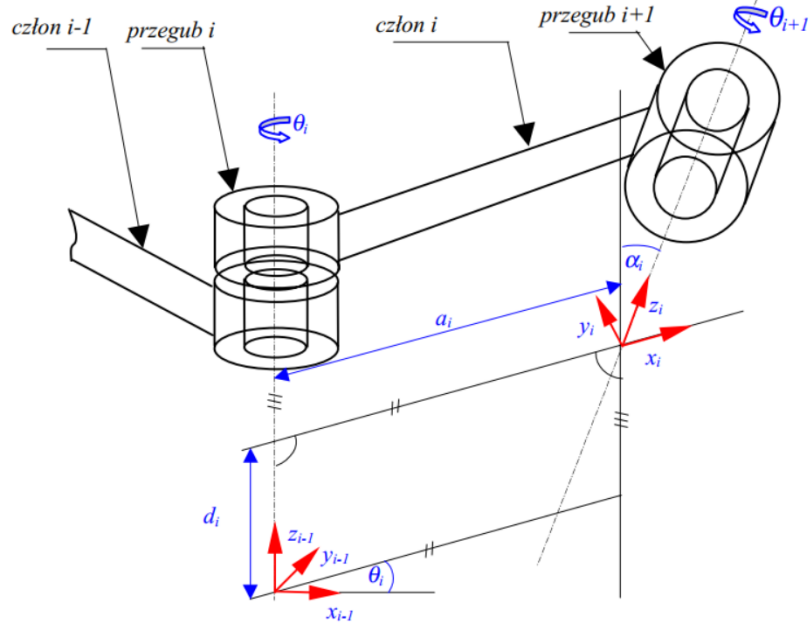
#### 1.1.1 Model geometrii - Janek

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 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 Denavit-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  $x_i$ ,  $\alpha_i$  - 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  $z_{i-1}$ ,  $\theta_i$  - kąt konfiguracji, kąt między osiami  $x_{i-1}$  i  $x_i$  mierzony wokół osi  $z_{i-1}$

$${}^{i-1}T_i = T_{z,\theta} T_{z,d} T_{x,a} T_{x,\alpha} \quad (1.1)$$

gdzie:  ${}^{i-1}T_i$  - 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  $\theta_i$ ,  $T_{z,d}$  - 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  $\alpha_i$ .

Macierze te wyrażone są następującymi zależnościami:

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

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

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

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

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]:

$$P_{Am} = {}^m T_k P_{Ak}, k > m \quad (1.6)$$

gdzie:

$${}^m T_k = {}^m T_{m+1} \quad {}^{m+1} T_{m+2} \dots {}^{k-1} T_k \quad (1.7)$$

$P_{Ak} = \begin{bmatrix} x_{Ak} \\ y_{Ak} \\ z_{Ak} \\ 1 \end{bmatrix}$  - 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 Lagrangea - Eulera o postaci [2]

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

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

$$L = K - P \quad (1.9)$$

$K$  - całkowita energia kinetyczna układu,  $P$  - całkowita energia potencjalna układu,  $q_i$  - uogólnione współrzędne układu w przegubie  $i$ ,  $\tau_i$  - uogólnione wymuszenia (siły lub momenty) działające w przegubie  $i$  <sup>1</sup>.

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

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

<sup>1</sup>W robocie o przegubach obrotowych, jako uogólnione współrzędne przyjmuje się położenia kątowe w przegubach, natomiast jako uogólnione wymuszenia przyjmuje się momenty napędowe w przegubach.

## 8ROZDZIAŁ 1. OPIS MECHANIZMÓW WIELOCZŁONOWYCH - JANEK+KUBA

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:

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

gdzie:

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

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

$J_i$  - jednorodna macierz inercji członu  $i$  wyrażona zależnością:

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

$I_{XXi}, I_{YYi}, I_{ZZi}$  - masowe momenty bezwładności członu  $i$ ,  $I_{XYi}, I_{XZi}, I_{YZi}$  - masowe momenty dewiacji członu  $i$ ,  $r_i = [x_i \ y_i \ z_i]^T$  - 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**

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

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

$$V_i = \dot{\theta}^T H_i \dot{\theta} \quad (1.16)$$

$$H_i \in R^{n \times n}, i = 1, 2, \dots, n \quad (1.17)$$

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



gdzie:

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

### Wektor $G(\theta)$ ma postać

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

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

gdzie:  $\bar{g} = [g_x \ g_y \ g_x \ 0]^T$  -jednorodny wektor grawitacji wyrażony w bazowym układzie współrzędnych,  $\sqrt{g\bar{g}^T} = 9.8062[m/s^2]$  - na powierzchni ziemi,  $m_i$  masa członu  $i$ .

Należy dodać, że istnieje szereg innych metod wyznaczania równań dynamiki robota, np. metoda Newtona-Eulera, uogólniona metoda d'Alemberta [2], metoda Christoffela-Lagrangea, metoda Walkera-Orina [3]. Ich powstanie wynika 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 syntezy układu sterowania.

### Model tarcia

W układach mechanicznych wyróżniamy różne rodzaje sił. Pierwsze z nich, to **siły zachowawcze**, dla których którejś pracy wykonanej przy przemieszczeniu ciała po torze zamkniętym o dowolnym kształcie równa jest zero. Siły nie spełniające tego warunku to tzw **siły dyssypatywne**, inaczej nazywane **siłami rozpraszającymi**.

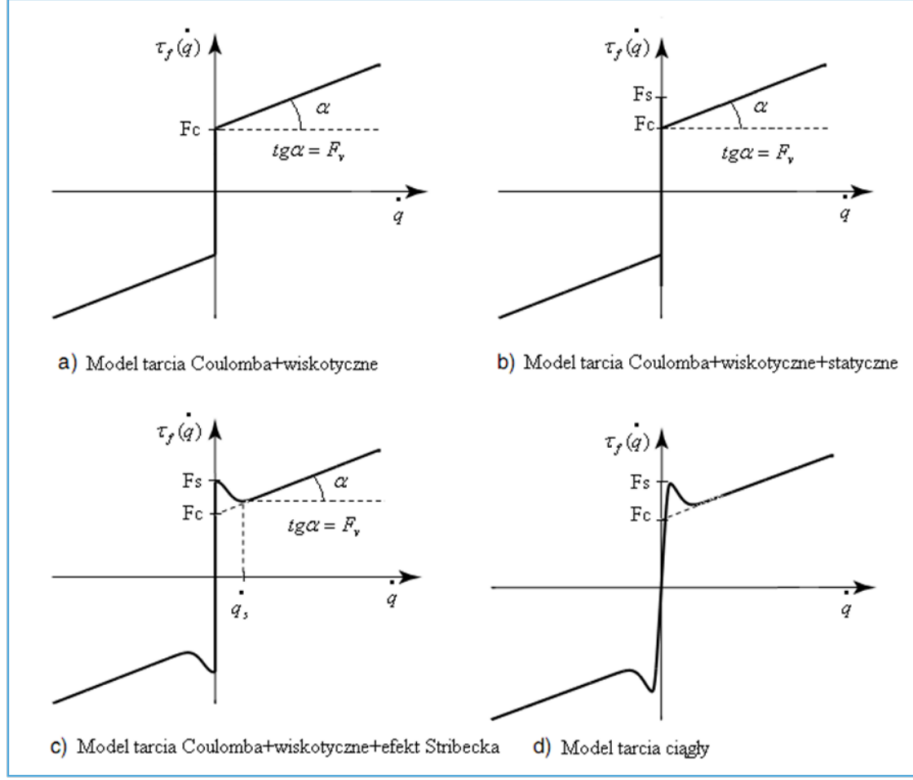
Siły tarcia, występujące w układach mechanicznych, możemy zaliczyć do sił dyssypatywnych. Występujące w układach mechanicznych tarcie jest nieliniowym i niestacjonarnym zjawiskiem o parametrach rozłożonych. W systemach mechatronicznych zmniejsza ono dokładność oraz pogarsza jakość przebiegów dynamicznych.

Modele tarcia najogólniej można podzielić na trzy grupy [21]:

- white-box - wykorzystujemy podstawy fizyczne badanego zjawiska i dzielimy je na statyczne oraz dynamiczne.
- black-box - modele parametryczne, bazujące na danych eksperymentalnych.
- grey-box - różne cechy modeli white-box i black-box.

W modelowaniu fizycznym przy modelowaniu mechanizmów wielocłonowych stosuje się zwykle statyczne modele typu white-box, które przy pomocy równań algebraicznych, opisują podstawowe właściwości tarcia. Wyróżnia się wśród nich

modele klasyczne uwzględniające różne kombinacje tarcia *Coulomba* (parametr  $F_c$ ), wiskotycznego (parametr  $F_v$ ), statycznego (parametr  $F_s$ ) oraz efektu *Stribeck*. Na rys. 1.1.3 przedstawiono różne modele tarcia statycznego.



Rysunek 1.2: Modele tarcia statycznego

Modele tarcia, przedstawione na rys. 1.1.3 opisywane są następującymi zależnościami:

- Model (a)

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

- Model (b)

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

- Model (c)

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

- Model (d) - w którym zdefiniowano niewielkie otoczenie, wewnątrz którego przyjmuje się zerową wartość prędkości. W tym przypadku tarcie dla jest

zależne od zewnętrznych sił ( $F_e$ ) utrzymujących układ w spoczynku, zaś poza przytłumionym otoczeniem jest zazwyczaj opisywane w funkcji prędkości (Model (a), (b), lub (c))

$$\tau_f(\dot{q}, F_e) = \begin{cases} \tau_f(\dot{q}) & \text{dla } |\dot{q}| \geq \beta \\ \tau_f(F_e) & \text{dla } |\dot{q}| < \beta \end{cases} \quad (1.24)$$

#### 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 [?, ?, ?, ?]

##### Własności strukturalne modelu robota (WMR)

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

$$M(q) = M^T(q) \quad (1.25)$$

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

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

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

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

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

1. Macierz  $M(q)$  jest nieosobliwa

$$\det[M(q)] \neq 0 \quad (1.28)$$

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

$$M(q) = P^T P \quad \det P \neq 0 \quad (1.29)$$

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

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

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

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

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

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

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

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

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

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*

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

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]:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \quad (1.36)$$

oraz przyjmując, że wejściami do układu są sygnały sterujące  $\tau$ , natomiast wyjściami położenia kątowne  $\theta$ , równanie dynamiki (1.1.3) można wyrazić w przestrzeni zmiennych stanu jako

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

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

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

gdzie:

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

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

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

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 wieloczęłonowych - 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 przykład układu niedosterowanego. Jako sygnał sterujący przyjmuje się moment silnika powodujący przemieszczenie wózka, natomiast sygnałami wyjściowymi jest położenie liniowe i kątowe wahadła.

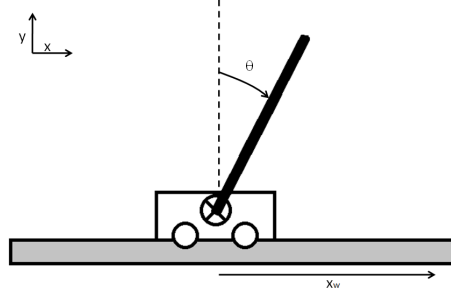
#### 2.1.1 Model nieliniowy w oparciu o równania fizyczne

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  $\theta$ . 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.

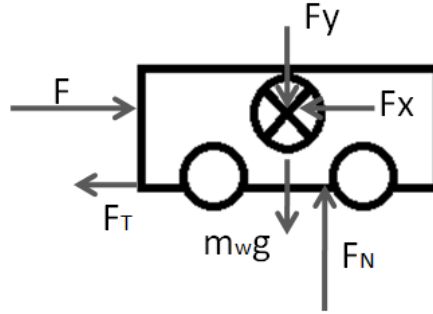
$$m_W \dot{x}_W = \sum F \quad (2.1)$$

$$m_W \dot{x}_W = F - F_T - F_x \quad (2.2)$$

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:

$$F_T = b\dot{x}_W \quad (2.3)$$

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:

$$F = \frac{M_s}{r} \quad (2.4)$$

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:

$$F_x = m_P \ddot{x}_P \quad (2.5)$$

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

$$x_P = x_W + l \sin(\theta) \quad (2.6)$$

gdzie:  $l$  jest odległością środka ciężkości pręta od wózka.



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

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

oraz

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

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

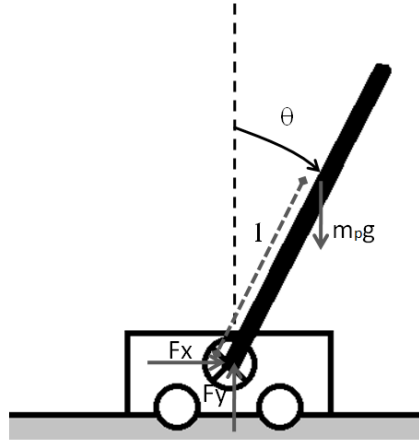
$$F_x = m_P \ddot{x}_W - m_P l \dot{\theta}^2 \sin(\theta) + m_P l \ddot{\theta} \cos(\theta) \quad (2.9)$$

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

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

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

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:

$$F_y - m_P g = m_P \ddot{y}_P \quad (2.12)$$

$$y_P = y_W + l \cos(\theta) \quad (2.13)$$

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:

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

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

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

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

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

$$I\ddot{\theta} = \sum \overline{M}, \quad (2.17)$$

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:

$$\overline{M} = \overline{F} \times \overline{r}, \quad (2.18)$$

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:

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

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

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

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

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  $\ddot{\theta}$ :

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

### 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

Jedną z najbardziej popularnych metod sterowania pozycyjnego w robotyce jest metoda odwrotnego modelu. Metoda modelu odwrotnego jest stosowana w robotyce metodą linearyzacji i dekompozycji modelu matematycznego manipulatora, dzięki której można sterować niezależnie wszystkimi ramionami robota z wykorzystaniem technik sterowania obiektami liniowymi. Metoda odwrotnego modelu ma ten zaletę w porównaniu z innymi metodami linearyzacji (np. rozwinięcie w szereg Taylora) modelu, że kompensuje nieliniowości w całym zakresie zmian współrzędnych złączowych, a nie tylko w pobliżu punktu, wokół którego linearyzujemy model.

Na rys. ?? przedstawiony jest ogólny schemat sterowania robota (mechanizmu wieloczołowego) z wykorzystaniem modelu odwrotnego.

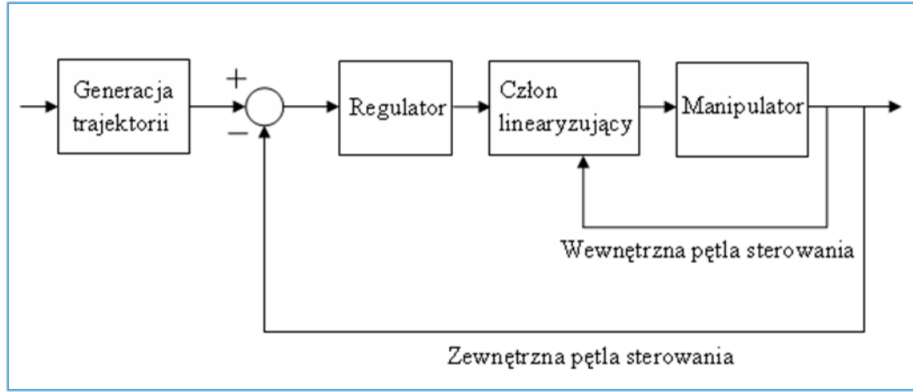
W strukturze sterowania możemy wyróżnić dwie płaszczyzny zwrotnego:

- Płaskość wewnętrzną linearyzujemy i rozdzielamy prawo sterowania na części modeli oraz sprzężeń
- Płaskość zewnętrzną sprzężenia zwrotnego pozwalającą na sterowanie manipulatora w dany sposób

Aby wyznaczyć odpowiednie sygnały sterujące, należy wszystkie układy związane z przegubami uniezależnić od siebie. Pozwoli to na uproszczenie problemu wyznaczenia sterowania do wyznaczenia sygnałów sterujących dla szeregu odprężających układów drugiego rzędu. Metody odprężania przegubów robota opisano m.in. w [2], [4].

Dany jest model robota opisany równaniem 1.1.3. Przyjmując postać wektora sygnałów sterujących:

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



Rysunek 4.1: Schemat sterowania robota z wykorzystaniem modelu odwrotnego

oraz przyjmując, że:

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

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

można rozdzielić układy związane z przegubami w równaniu (1.1.3). Macierze  $\hat{M}(x, t)$ ,  $\hat{V}(x, t)$ ,  $\hat{G}(x, t)$  są oszacowaniami (estymatami) macierzy  $M(x, t)$ ,  $V(x, t)$ ,  $G(x, t)$ , w równaniach Lagrange'a-Eulera, liczonymi według wzorów podanych w rozdziale 1, na podstawie oszacowanych parametrów robota.

Równanie opisujące dynamikę robota można zapisać, wykorzystując zależności (3.10) i (4.2.1), w postaci:

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

Zakładając, że oszacowane parametry robota mają te same wartości co parametry rzeczywiste, równanie (4.2.1) upraszcza się do postaci:

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

Zakładając, że macierz  $M(x)$  jest macierzą nieosobliwą, oraz  $M(x) = \hat{M}(x)$ , w wyniku odspzęgnienia uzyskuje się sześć niezależnych układów opisanych równaniami drugiego rzędu, postaci:

$$\ddot{q} = \hat{\tau} \quad (4.6)$$

Układ (4.2.1) można opisać równaniami w przestrzeni zmiennych stanu jako

$$\begin{cases} \dot{x}_i = Ax_i + B\hat{\tau}u \\ y_i = Cx_i \\ i = 1, \dots, n \end{cases} \quad (4.7)$$

gdzie

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



Głównym wymaganiem przy stosowaniu metody odwrotnego modelu do sterowania jest konieczność zapewnienia dokładnych oraz szybkich pomiarów współrzędnych zbieżnych. Jest to silne założenie, ponieważ pomiar parametrów robota jest zawsze obciążony jest pewnym błędem to nie wszystkie parametry modelu są dokładnie określone.

#### **4.2.2 Transmittancja 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

- 5.1 Regulator dla robota o wielu stopniach swobody
- 5.2 Linearyzacja sprzężenia zwrotnego
- 5.3 Dobór funkcji Lapunova dla potrzeb sterowania mechanizmów wieloczłonowych
- 5.4 Sterownie pozycyjne i sterowanie nadążne
- 5.5 Projektowanie sterowania jako zadanie optymalizacji



## 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
- 7.2 Sterowanie predycyjne 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* [?].

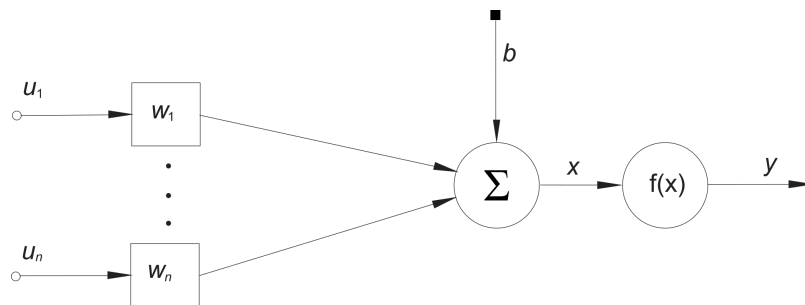
Model matematyczny pojedynczego neuronu jest opisany następującą zależnością

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

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)**

$$f(x) = a_{\text{ln}} x \quad (8.2)$$

gdzie  $a_{\text{ln}}$  - współczynnik nachylenia funkcji.

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

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

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

Funkcja sigmoidalna może mieć postać unipolarną

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

gdzie  $a_{\text{su}}$  - parametr funkcji sigmoidalnej unipolarnej.

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

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

Funkcja sigmoidalna może mieć również postać bipolarną

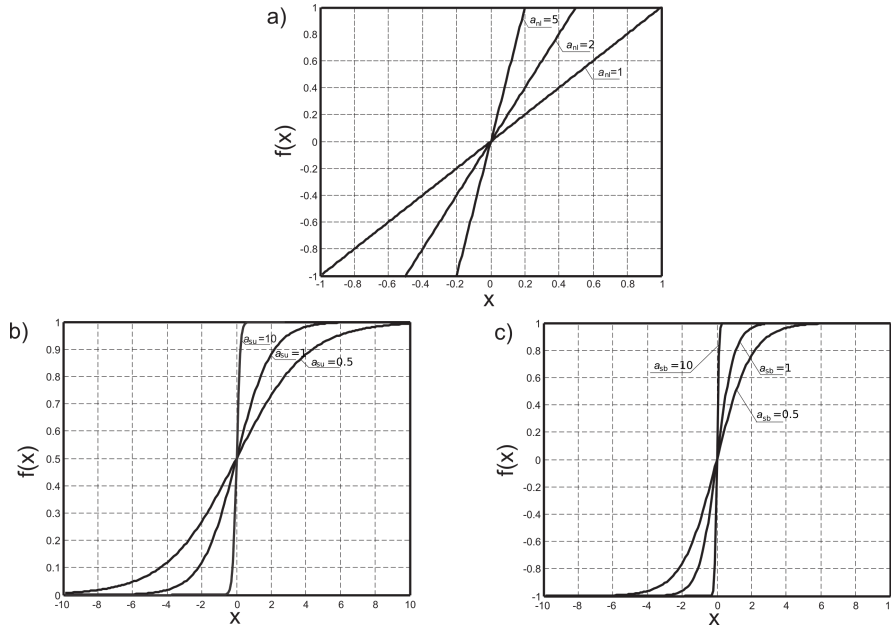
$$f(x) = \tanh(a_{\text{sb}} x) = \frac{e^{a_{\text{sb}} x} - e^{-a_{\text{sb}} x}}{e^{a_{\text{sb}} x} + e^{-a_{\text{sb}} x}} \quad (8.6)$$

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

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

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  $y = [y_j]_{m \times 1}$ .

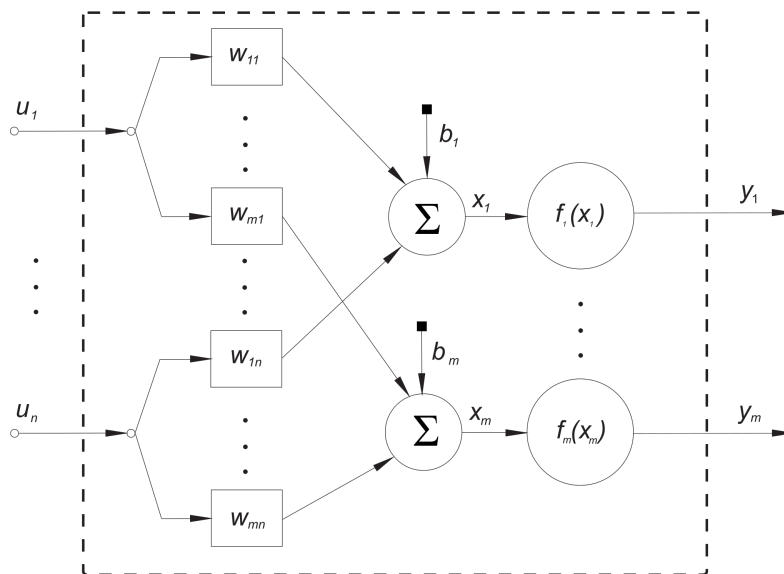
Przetwarzanie danych zachodzące w perceptronie jednowarstwowym opisane jest następująco

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

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

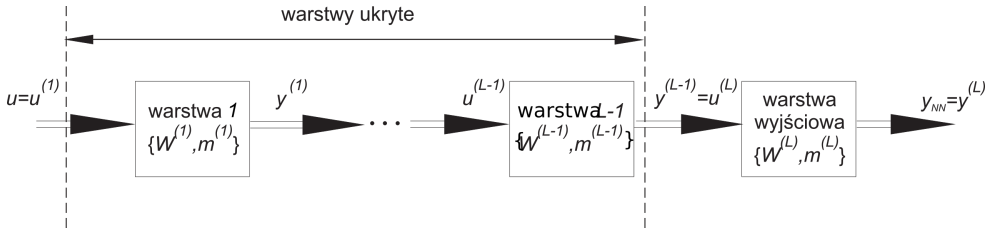
$$\begin{aligned} u^{(1)} &= [u_j^{(1)}]_{n^{(1)} \times 1} = u, \quad y^{(1)} = [y_j^{(1)}]_{m^{(1)} \times 1}, \quad n^{(1)} = n \\ u^{(i)} &= [u_j^{(i)}]_{n^{(i)} \times 1} = y^{(i-1)}, \quad y^{(i)} = [y_j^{(i)}]_{m^{(i)} \times 1}, \quad n^{(i)} = m^{(i-1)}, \quad i = 2, \dots, L \end{aligned} \quad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

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

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

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

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

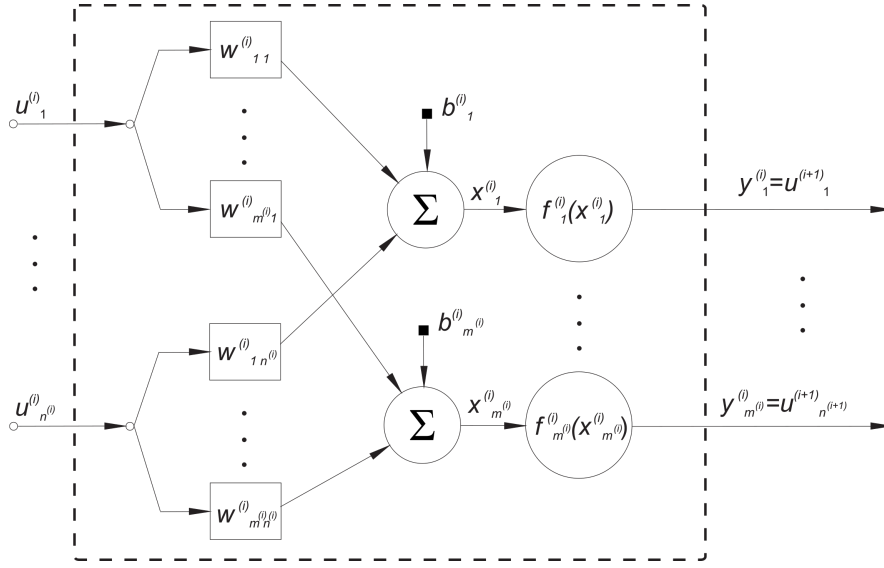
gdzie  $u^{(i)} = [u_j^{(i)}]_{n^{(i)} \times 1}$  - wektor sygnałów wejściowych do warstwy,  $x^{(i)} = [x_j^{(i)}]_{m^{(i)} \times 1}$  - wektor sygnałów wyjściowych sumatorów związanych z poszczególnymi neuronami warstwy,  $y^{(i)} = [y_j^{(i)}]_{m^{(i)} \times 1}$  - wektor sygnałów wyjściowych z 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

$$y_{NN} = F_{NN}(u), \quad F_{NN} : R^n \rightarrow R^m, \quad y_{NN} \in R^m, u \in R^n \quad (8.12)$$

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

$$y = F(u), \quad F : R^n \rightarrow R^m, \quad y \in R^m, u \in R^n \quad (8.13)$$

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

Aproksymacją w przypadku sieci neuronowej będziemy nazywali funkcję  $y_{\text{NN}} = F_{\text{NN}}(u)$  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

$$s(k) = \{u(k), y(k) : y(k) = F[u(k)]\}, \quad k = 1, \dots, N \quad (8.14)$$

gdzie  $k$  - numer próbki.

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

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

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

$$E = f[e(1), \dots, e(N)] \quad (8.16)$$

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

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

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

$$E = E(W) \quad (8.18)$$

gdzie

$$W = [b_1^{(1)}, w_{1,1}^{(1)}, \dots, w_{m^{(1)}, n^{(1)}}^{(1)}, b_1^{(2)}, \dots, w_{m^{(L)}, n^{(L)}}^{(L)}]^T = [w_i]_{n_W \times 1} = [w_1, \dots, w_{n_W}]^T \quad (8.19)$$

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

$$n_W = \sum_{l=1}^L m^{(l)}(n^{(l)} + 1) \quad (8.20)$$

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*),

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

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*) [?, ?, ?].

Zaletą sieci neuronowej jest możliwość realizacji sieci aproksymującej z zadaną dokładnością prawie każdą funkcję ciągłą określoną na zbiorze zwartym [?, ?, ?]. 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 [?]. W przypadku sieci neuronowej jednokierunkowej wielowarstwowej zadanie to polega na znalezieniu wektora wag  $\hat{W}$  spełniającego zależność

$$E(\hat{W}) = \min_W E(W) \quad (8.22)$$

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

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

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

Kierunek poprawy oraz współczynnik kroku są wyznaczane tak, aby

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

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* [?, ?, ?, ?].

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 brany 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

$$i \geq i_{\max} \quad (8.25)$$

Jeśli warunek (8.25) jest spełniony zakończyć trenowanie sieci.

4. Obliczyć wartość funkcji celu sieci neuronowej  $E(W_{(i)})$  a następnie sprawdzić warunek:

$$E(W_{(i)}) \leq E_0 \quad (8.26)$$

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

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

8. Wrócić do punktu 2.

□

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

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

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

$$\nabla E(W) = \frac{\partial E}{\partial W} \Big|_{W=W_{(i)}} = \left[ \begin{array}{c} \frac{\partial E(W)}{\partial w_1} \\ \vdots \\ \frac{\partial E(W)}{\partial w_{n_W}} \end{array} \right]_{n_W \times n_W} \Big|_{W=W_{(i)}} \quad (8.29)$$

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

$$H(W_{(i)}) = \frac{\partial^2 E}{\partial^2 W} \Big|_{W=W_{(i)}} = \left[ \begin{array}{ccc} \frac{\partial^2 E(W)}{\partial w_1 \partial w_1} & \cdots & \frac{\partial^2 E(W)}{\partial w_1 \partial w_{n_W}} \\ \vdots & & \vdots \\ \frac{\partial^2 E(W)}{\partial w_{n_W} \partial w_1} & \cdots & \frac{\partial^2 E(W)}{\partial w_{n_W} \partial w_{n_W}} \end{array} \right]_{n_W \times n_W} \Big|_{W=W_{(i)}} \quad (8.30)$$

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

### Metoda wstecznej propagacji błędów

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 błędów** (ang. *error backpropagation*) [?]. W algorytmie wstecznej propagacji błędów 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 błędów wymaga się, aby funkcja aktywacji każdego neuronu była różniczkowalna i znana była wartość jej pochodnej względem argumentu. Algorytm wstecznej propagacji błędów można zapisać następująco [?, ?].

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

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  $m^{(i)}, i = 1, \dots, L$ , sygnały wejściowe do sieci  $u_j^{(1)} = u_j, j = 1, \dots, n$ , macierze wag i wektory przesunięć sygnałów wyjściowych sumatorów warstw sieci  $W^{(i)}, b^{(i)}, i = 1, \dots, L$ , liczba sygnałów wyjściowych z sieci  $m^{(L)} = m$ , sygnały wyjściowe z sieci  $y_{NN} = [y_{NNi}]_{m^{(L)} \times 1}$
2. Obliczyć  $x^{(l)}, y^{(l)}$  wg (8.11) dla  $l = 1, \dots, L$ .
3. Przyjąć  $l=L$ .
4. Obliczyć wartość pochodnej funkcji celu sieci neuronowej względem sygnałów wyjściowych z sieci

$$E'_{y_{NNi}} = \frac{\partial E}{\partial y_{NNi}}, \quad i = 1, \dots, m^{(L)} \quad (8.31)$$

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

$$E'_{x_i^{(L)}} = \frac{\partial E}{\partial x_i^{(L)}} = \frac{\partial f_i^{(L)}(x_i^{(L)})}{\partial x_i^{(L)}} E'_{y_{NNi}}, \quad i = 1, \dots, m^{(L)}, \quad (8.32)$$

jeśli  $l < L$  wyznaczyć

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

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)

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

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

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

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

### Metoda największego spadku

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

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

natomiast wartość współczynnika kroku powinna spełniać zależność

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

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* [?, ?]. Stosowane są także inne modyfikacje algorytmu największego spadku, do których należą m.in. metoda największego spadku z momentem [?, ?, ?], metoda QUICKPROP (ang. *quick backpropagation*) [?], metoda RPROP (ang. *resilient backpropagation*) [?].

### Metoda gradientów sprzężonych

Metoda gradientów sprzężonych, w przypadku nieliniowego układu równań (np. sieci neuronowej), jest modyfikacją metody gradientów sprzężonych dla układu równań liniowych [?]. 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

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

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

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

oraz kierunek poprawy w iteracji  $i$

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

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

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 [?].

Algorytm gradientów sprzężonych charakteryzuje się skończoną liczbą iteracji  $n_{\text{iter}}$  [?, ?], która jest ograniczona przez liczbę modyfikowanych parametrów (wag sieci)  $n_W$

$$n_{\text{iter}} \leq n_W \quad (8.42)$$

### Metoda *Levenberga - Marquardta*

Metoda *Levenberga - Marquardta* [?, ?, ?, ?, ?] 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* [?] gdzie zmiany wag  $W$  sieci neuronowej w  $i$ -tej iteracji wyznacza się wg zależności

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

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

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

oraz

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

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

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

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

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

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

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

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

### 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 [?].

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.  $Z_{\text{train}}$ . Na jego podstawie można także określić jak dobrze układ estymuje dane wykorzystane podczas trenowania sieci. Wartość funkcji celu dla zbioru treningowego jest oznaczana  $E_{\text{train}}$  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*  $Z_{\text{test}}$  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  $E_{\text{test}}$ .

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

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 [?] 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 [?, ?]. 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 [?] przyjmuje się oszacowanie wartości VCdim jako liczbę wszystkich wag sieci ( $n_W$ ), natomiast minimalną liczbę próbek zbioru treningowego  $N_{\text{train}}$  jako

$$N_{\text{train}} \approx 2VCdim \approx 2n_W \quad (8.49)$$

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

$$N_{\text{train}} > 2n_W \quad (8.50)$$

- 8.3 Regulator stanu mechanizmu wieloczłowe-  
go oparty o sieci neuronowe**
- 8.4 Logika rozmyta w sterowaniu mechanizmów  
wieloczłonowych**
- 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 intelligence*. McGraw-Hill Book Company, USA, 1987.
- [3] W. Hejmo. *Sterowanie robotami i manipulatorami przemysłowymi: metody i modele matematyczne*. Wydawnictwo Politechniki Krakowskiej, Kraków, 1997.
- [4] Craig J. J. *Wprowadzenie do robotyki, mechanika i sterowanie*. WNT, Warszawa, 1995.