PROGRAM ROZWOJOWY
POLITECHNIKI WARSZAWSKIEJ

Maciej Ławryńczuk, Piotr Marusak

Sztuczna inteligencja w automatyce

preskrypt

Warszawa 2009–2018

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego

Publikacja dystrybuowana jest bezpłatnie

Spis treści

1. Wstęp

Niniejszy skrypt jest przeznaczony dla studentów studiów drugiego stopnia kursu „Soft computing w automatyce" oraz osób zainteresowanych użyciem technik soft computing do projektowania układów regulacji automatycznej. W związku z tym założono, że czytelnik opanował podstawy wiedzy z zakresu układów dynamicznych a także projektowania układów regulacji automatycznej (np. na wykładzie „Podstawy sterowania"). Zalecane jest także opanowanie materiału przedstawianego podczas wykładu „Sterowanie procesów".

Spośród technik soft computing, szczególnie przydatne w automatyce są sztuczne sieci neuronowe (ang. artificial neural networks), układy rozmyte (ang. fuzzy systems) oraz algorytmy genetyczne (ang. genetic algorithms). Dlatego na nich skoncentrowano się w niniejszym skrypcie. Zakres zastosowań wspomnianych technik soft computing jest szeroki jednak nas interesuje ich użycie przy projektowaniu układów sterowania.

Dzięki sztucznym sieciom neuronowym oraz modelom rozmytym można stosunkowo łatwo modelować układy dynamiczne. W przypadku sieci neuronowych ich dużą zaletą jest możliwość uczenia modelu na podstawie próbek pozyskanych z obiektu regulacji. Modele rozmyte z kolei umożliwiają, w stosunkowo prosty sposób, użycie podczas ich konstruowania wiedzy eksperta. Ponadto warto wspomnieć o podejściu łączącym zalety obydwu technik. Są to rozmyte sieci neuronowe. Dzięki nim rozmyty model obiektu może być zidentyfikowany (lub jedynie dostrojony) korzystając z mechanizmu uczenia oferowanego przez sieci neuronowe.

Układy rozmyte mają także zastosowanie przy projektowaniu regulatorów na podstawie wiedzy eksperta, np. operatora procesu. Wówczas, korzystając z reguł opisujących zachowanie człowieka – operatora, który sterował procesem, budowany jest regulator. Zauważmy, że zakłada się przy tym, że operator zna zachowanie obiektu regulacji (ma w swoim mózgu model obiektu), na podstawie którego podejmował decyzje. W związku z tym pomija się w tym przypadku etap identyfikacji modelu obiektu zakładając, że wiedza o tym modelu została zawarta i stosownie wykorzystana w regułach zachowania operatora – eksperta.

Algorytmy genetyczne są jedną z metod optymalizacji globalnej. Stosuje się je, gdy inne metody optymalizacji nie dają zadowalających wyników lub dany problem najprościej jest rozwiązać korzystając właśnie z tych algorytmów. Podczas procesu projektowania regulatora, algorytmy genetyczne można zastosować przy doborze parametrów regulatora dla danego obiektu regulacji wykorzystując symulację działania układu regulacji. Algorytmów genetycznych można także użyć w warstwie optymalizacji punktu pracy, gdzie model statyki procesu jest zwykle nieliniowy.

Kolejny rozdział zawiera krótkie przypomnienie zagadnień z zakresu projektowania algorytmów regulacji automatycznej, głównie w wersji dyskretnej, oraz układów sterowania z optymalizacją punktu pracy. W następnych rozdziałach przedstawione zostały:

— sztuczne sieci neuronowe (rozdz. 3),

— układy rozmyte (rozdz. 4),

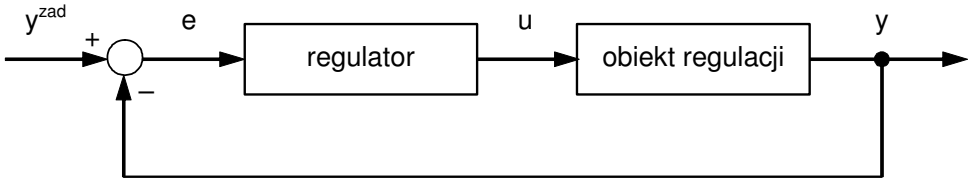
— algorytmy ewolucyjne (rozdz. 5)

w kontekście ich użycia do projektowania układów sterowania. Każdy z tych rozdziałów zawiera, w pierwszej swojej części, omówienie podstaw dotyczących danej techniki. Następnie, zaprezentowany jest sposób wykorzystania danego podejścia przy projektowaniu układów sterowania. Rozdziały 3 i 5 zostały napisane przez M. Ławryńczuka a rozdziały 2 i 4 – przez P. Marusaka.

2. Algorytmy regulacji automatycznej

2.1. Podstawowy układ regulacji

Rozważmy układ regulacji automatycznej przedstawiony na rys. 2.1. Zadaniem tego układu jest odtwarzanie wartości zadanej y^{zad} na wyjściu obiektu y tak dokładnie, jak to możliwe, czyli tak aby uchyb regulacji $e = y^{zad} - y$ był jak najmniejszy. Urządzeniem, które ma to zapewnić jest regulator generujący sygnał sterujący, którym wpływa w pożądany sposób na działanie obiektu. Podstawowym zadaniem projektanta układu regulacji jest wybór typu regulatora najlepiej spełniającego postawione zadanie (zależny w szczególności od obiektu regulacji) oraz jego dostrojenie do pracy z danym obiektem.



Rys. 2.1. Schemat układu regulacji automatycznej;
u – sygnał sterujący, y – wyjście, y^{zad} – wartość zadana, e – uchyb regulacji

W niniejszym rozdziale przypomniano kolejno:

— sposób badania stabilności układów dynamicznych korzystając z kryterium Hurwitza
— regulatory P, PI i PID,
— regulatory ze sprzężeniem od stanu,
— regulatory predykcyjne (szczególnie przydatne w przypadku obiektów o trudnej dynamice, w szczególności z opóźnieniami oraz gdy istotne są ograniczenia istniejące w układzie),
— warstwową strukturę sterowania.

Korzystano z kilku źródeł, w których zainteresowany czytelnik znajdzie rozwinięcie opisanych tu tematów. Dokładniejsze omówienie tematów zawartych w rozdz. 2.1, 2.2 oraz 2.3 znajduje się w publikacjach [2, 8, 13, 32, 35, 40], zaś tematów z rozdz. 2.4 i 2.5 – w książce [38].

2.1.1. Badanie stabilności układów ciągłych metodą Hurwitza

Istotną właściwością układu regulacji, którą musi zapewnić projektant jest stabilność. Przypomnijmy, że liniowy, ciągły układ dynamiczny jest stabilny asymptotycznie wtedy i tylko wtedy, gdy wszystkie bieguny transmitancji ten układ opisującej mają ujemne części rzeczywiste.

W celu badania stabilności asymptotycznej, znając wielomian charakterystyczny układu, można skorzystać z kryterium Hurwitza. Dzięki zastosowaniu tego kryterium można w stosunkowo łatwy sposób sprawdzić, czy wszystkie pierwiastki wielomianu charakterystycznego leżą w lewej półpłaszczyźnie zmiennej zespolonej s, bez konieczności znajdowania tych pierwiastków.

Załóżmy, że wielomian charakterystyczny ma następującą postać:

$$M(s) = a_n \cdot s^n + \ldots + a_1 \cdot s + a_0. \tag{2.1}$$

Kryterium Hurwitza mówi, że wszystkie pierwiastki równania charakterystycznego leżą w lewej półpłaszczyźnie zmiennej zespolonej s wtedy i tylko wtedy, gdy:

1. wszystkie współczynniki wielomianu są dodatnie (a_i>0, i = 0, 1,…, n),
2. wszystkie wiodące minory główne macierzy $\boldsymbol{A}$ skonstruowanej w następujący sposób:

$$\boldsymbol{A}=\begin{bmatrix} a_1 & a_0 & 0 & \cdots & 0 & 0 \\ a_3 & a_2 & a_1 & \cdots & 0 & 0 \\ a_5 & a_4 & a_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1} & a_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & a_n \end{bmatrix} \tag{2.2}$$

są dodatnie (Δ_i>0, i == 0, 1,…, n). Zauważmy, że pierwszy wiodący minor główny Δ_1=a_1 będzie dodatni, jeśli pierwszy zestaw warunków jest spełniony.

Przykład 2.1 (*Przypadek szczególny – układ drugiego rzędu*)

Rozpatrzmy układ drugiego rzędu. Wtedy wielomian charakterystyczny jest następujący:

$$M(s)=a_2\cdot s^2+a_1\cdot s+a_0. \tag{2.3}$$

Zauważmy, że w takim razie wystarczy sprawdzić jedynie pierwszy zestaw warunków (czy wszystkie współczynniki wielomianu charakterystycznego są dodatnie). Jest tak ponieważ, w przypadku spełnienia pierwszego zestawu warunków, drugi wiodący minor główny:

$$\Delta_2=\begin{vmatrix} a_1 & a_0 \\ 0 & a_2 \end{vmatrix}=a_1\cdot a_2 \tag{2.4}$$

jest dodatni. Zauważmy, że chociaż znajdowanie pierwiastków równania charakterystycznego układu drugiego rzędu nie jest skomplikowane, użycie kryterium Hurwitza do badania stabilności znacznie upraszcza to zadanie.

Przykład 2.2 (*Przypadek szczególny – układ trzeciego rzędu*)

Rozpatrzmy układ trzeciego rzędu. Wtedy wielomian charakterystyczny ma postać:

$$M(s)=a_3\cdot s^3+a_2\cdot s^2+a_1\cdot s+a_0. \tag{2.5}$$

Tym razem oprócz sprawdzenia dodatniości parametrów wielomianu charakterystycznego wystarczy sprawdzić czy drugi wiodący minor główny

$$\Delta_2=\begin{vmatrix} a_1 & a_0 \\ a_3 & a_2 \end{vmatrix}=a_1\cdot a_2-a_0\cdot a_3 \tag{2.6}$$

jest dodatni. Jest tak ponieważ trzeci wiodący minor główny ma postać:

$$\Delta_3=\begin{vmatrix} a_1 & a_0 & 0 \\ a_3 & a_2 & a_1 \\ 0 & 0 & a_3 \end{vmatrix}=a_3\cdot\Delta_2. \tag{2.7}$$

Jest on więc dodatni jeśli a_3>0 i Δ_2>0.

Zauważmy, że z postaci macierzy $\boldsymbol{A}$ wynika, że w przypadku układu dowolnego rzędu nie jest potrzebne sprawdzanie wyznacznika tej macierzy (Δ_n), ponieważ zachodzi równość:

$$\Delta_n=a_n\cdot\Delta_{n-1}. \tag{2.8}$$

Przykład 2.3

Rozpatrzmy układ dynamiczny opisany następującą transmitancją:

$$G(s) = \frac{s^3 + 2 \cdot s^2 + 2 \cdot s + 1}{8 \cdot s^5 + 6 \cdot s^4 + 6 \cdot s^3 + 4 \cdot s^2 + 2 \cdot s + 1}. \tag{2.9}$$

Naszym zadaniem jest zbadanie stabilności tego układu. W tym celu skorzystamy z kryterium Hurwitza.

Wszystkie parametry równania charakterystycznego są dodatnie. Pierwszy zestaw warunków jest więc spełniony. Należy więc sprawdzić kolejno wiodące minory główne.

$$\Delta_2 = \begin{vmatrix} a_1 & a_0 \\ a_3 & a_2 \end{vmatrix} = \begin{vmatrix} 2 & 1 \\ 6 & 4 \end{vmatrix} = 8 - 6 = 2 > 0,$$

$$\Delta_3 = \begin{vmatrix} a_1 & a_0 & 0 \\ a_3 & a_2 & a_1 \\ a_5 & a_4 & a_3 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 0 \\ 6 & 4 & 2 \\ 8 & 6 & 6 \end{vmatrix} = 48 + 16 - 24 - 36 = 4 > 0,$$

$$\Delta_4 = \begin{vmatrix} a_1 & a_0 & 0 & 0 \\ a_3 & a_2 & a_1 & a_0 \\ a_5 & a_4 & a_3 & a_2 \\ 0 & 0 & a_5 & a_4 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 0 & 0 \\ 6 & 4 & 2 & 1 \\ 8 & 6 & 6 & 4 \\ 0 & 0 & 8 & 6 \end{vmatrix} = 6 \cdot D_3 - 8 \cdot \begin{vmatrix} 2 & 1 & 0 \\ 6 & 4 & 1 \\ 8 & 6 & 4 \end{vmatrix} = 24 - 8 \cdot (32 + 8 - 12 - 24)$$

$$= 24 - 32 = -8 < 0.$$

Czwarty wiodący minor główny jest ujemny. W takim razie badany układ nie jest asymptotycznie stabilny. Faktycznie, oprócz trzech biegunów położonych w lewej półpłaszczyźnie zmiennej zespolonej s ($s_1 = -0{,}6251$ oraz $s_{2,3} = -0.2699 \pm 0.5915$j), układ ten ma dwa bieguny położone w prawej półpłaszczyźnie ($s_{4,5} = 0.2074 \pm 0.6558$j).

Przykład 2.4

Rozpatrzmy tym razem układ regulacji z rys. 2.1, przy czym załóżmy, że obiekt regulacji jest opisany transmitancją:

$$G(s) = \frac{1}{s^3 + 6 \cdot s^2 + 11 \cdot s + 6} \tag{2.10}$$

oraz że zastosowano regulator proporcjonalny ($R(s)=K$). Należy znaleźć taki zakres wartości wzmocnienia $K>0$, żeby układ regulacji był asymptotycznie stabilny.

W celu wykonania zadania użyjemy kryterium Hurwitza. Zauważmy, że transmitancja układu zamkniętego jest opisana wzorem:

$$G_z(s) = \frac{R(s) \cdot G(s)}{1 + R(s) \cdot G(s)}, \tag{2.11}$$

czyli w rozważanym przypadku:

$$G_z(s) = \frac{K}{s^3 + 6 \cdot s^2 + 11 \cdot s + 6 + K}.$$

Chcemy, aby bieguny układu zamkniętego były położone w lewej półpłaszczyźnie zmiennej zespolonej *s*. Zauważmy, że z pierwszego zestawu warunków wynika, że (pozostałe współczynniki równania charakterystycznego są dodatnie):

$$a_0 = K + 6 > 0\,. \tag{2.12}$$

Ponadto

$$\Delta_2 = \begin{vmatrix} a_1 & a_0 \\ a_3 & a_2 \end{vmatrix} = \begin{vmatrix} 11 & K+6 \\ 1 & 6 \end{vmatrix} = 66 - K - 6 = -K + 60 > 0\,. \tag{2.13}$$

Badany układ jest trzeciego rzędu, więc trzeci wiodący minor główny nie musi być sprawdzany (patrz przykład 2.2). Z ostatniego warunku wynika więc, że jeśli wartość wzmocnienia będzie liczbą dodatnią spełniającą warunek $K < 60$, to układ zamknięty będzie asymptotycznie stabilny.

2.1.2. Badanie stabilności układów dyskretnych metodą Hurwitza

Przypomnijmy, że liniowy, dyskretny układ dynamiczny jest stabilny asymptotycznie wtedy i tylko wtedy, gdy wszystkie bieguny transmitancji ten układ opisującej mają moduły mniejsze niż 1 (znajdują się wewnątrz okręgu jednostkowego).

W przypadku układów dyskretnych należy najpierw dokonać przekształcenia biliniowego transmitancji opisującej układ, tzn. w miejsce zmiennej zespolonej *z* podstawić:

$$z = \frac{w+1}{w-1}\,. \tag{2.14}$$

Przekształcenie to odwzorowuje bowiem koło jednostkowe na lewą półpłaszczyznę układu współrzędnych. Po zastosowaniu tego przekształcenia można skorzystać z kryterium Hurwitza.

Przykład 2.5

Rozpatrzmy układ dynamiczny opisany następującą transmitancją dyskretną:

$$G(z) = \frac{4 \cdot z^2 + z + 1}{z^4 + z^3 + 0{,}35 \cdot z^2 + 0{,}05 \cdot z + 0{,}0024}\,. \tag{2.15}$$

Należy zbadać stabilność tego układu używając w tym celu kryterium Hurwitza. Układ jest czwartego rzędu a jego wielomian charakterystyczny jest następujący:

$$M(z) = b_4 \cdot z^4 + b_3 \cdot z^3 + b_2 \cdot z^2 + b_1 \cdot z + b_0\,, \tag{2.16}$$

gdzie b_4=1, b_3=1, b_2=0,35, b_1=0,05, b_0=0,0024.

W celu skorzystania z kryterium Hurwitza należy najpierw dokonać przekształcenia biliniowego (2.14). Wtedy otrzymamy:

$$M(w) = b_4 \cdot \left(\frac{w+1}{w-1}\right)^4 + b_3 \cdot \left(\frac{w+1}{w-1}\right)^3 + b_2 \cdot \left(\frac{w+1}{w-1}\right)^2 + b_1 \cdot \left(\frac{w+1}{w-1}\right) + b_0$$

Równanie charakterystyczne badanego układu, w nowych współrzędnych ma postać:

$$b_4 \cdot (w+1)^4 + b_3 \cdot (w+1)^3 \cdot (w-1) + b_2 \cdot (w+1)^2 \cdot (w-1)^2 + b_1 \cdot (w+1) \cdot (w-1)^3 + b_0 \cdot (w-1)^4 = 0. \tag{2.17}$$

Po wykonaniu stosownych przekształceń, otrzyma się następujące równanie:

$$a_4 \cdot w^4 + a_3 \cdot w^3 + a_2 \cdot w^2 + a_1 \cdot w + a_0 = 0\,,$$

gdzie

$$a_4 = b_4 + b_3 + b_2 + b_1 + b_0 = 2{,}4024\,,$$

$$a_3 = 4 \cdot b_4 + 2 \cdot b_3 - 2 \cdot b_1 - 4 \cdot b_0 = 5{,}8904\,,$$

$$a_2 = 6 \cdot b_4 - 2 \cdot b_2 + 6 \cdot b_0 = 5{,}3144\,,$$

$$a_1 = 4 \cdot b_4 - 2 \cdot b_3 + 2 \cdot b_1 - 4 \cdot b_0 = 2{,}0904\,,$$

$$a_0 = b_4 - b_3 + b_2 - b_1 + b_0 = 0{,}3024\,.$$

Pierwszy zestaw warunków z kryterium Hurwitza jest spełniony, ponieważ wszystkie współczynniki a_i (i=0,…,4) przekształconego wielomianu charakterystycznego są dodatnie. Należy więc teraz sprawdzić wiodące minory główne.

$$\Delta_2 = \begin{vmatrix} a_1 & a_0 \\ a_3 & a_2 \end{vmatrix} = \begin{vmatrix} 2{,}0904 & 0{,}3024 \\ 5{,}8904 & 5{,}3144 \end{vmatrix} = 9{,}3280 > 0\,,$$

$$\Delta_3 = \begin{vmatrix} a_1 & a_0 & 0 \\ a_3 & a_2 & a_1 \\ 0 & a_4 & a_3 \end{vmatrix} = \begin{vmatrix} 2{,}0904 & 0{,}3024 & 0 \\ 5{,}8904 & 5{,}3144 & 2{,}0904 \\ 0 & 2{,}4024 & 5{,}8904 \end{vmatrix} = 44{,}4475 > 0\,.$$

W takim razie wszystkie pierwiastki równania charakterystycznego (2.17) leżą w lewej półpłaszczyźnie zmiennej zespolonej w. To oznacza z kolei, że wszystkie pierwiastki wielomianu charakterystycznego (2.16) leżą wewnątrz okręgu jednostkowego. W takim razie badany układ jest asymptotycznie stabilny. Faktycznie, bieguny badanego układu są następujące: $z_1 = -0{,}1$, $z_2 = -0{,}2$, $z_3 = -0{,}3$, $z_4 = -0{,}4$.

2.2. Regulatory ze sprzężeniem od wyjścia

Często stosowanym regulatorem jest regulator PID (proporcjonalno–całkowo–różniczkowy). Jest on złożony z trzech podstawowych członów. Wzór opisujący ogólną postać algorytmu PID w wersji ciągłej jest następujący:

$$u(t) = k_p \left[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{de(t)}{dt} \right], \tag{2.18}$$

gdzie k_p jest wzmocnieniem regulatora, T_i jest stałą czasową całkowania, zwaną też czasem zdwojenia, T_d jest stałą różniczkowania, zwaną również czasem wyprzedzenia. Transmitancja opisująca taki regulator jest następująca:

$$R(s) = k_p \left[1 + \frac{1}{T_i \cdot s} + T_d \cdot s \right]. \tag{2.19}$$

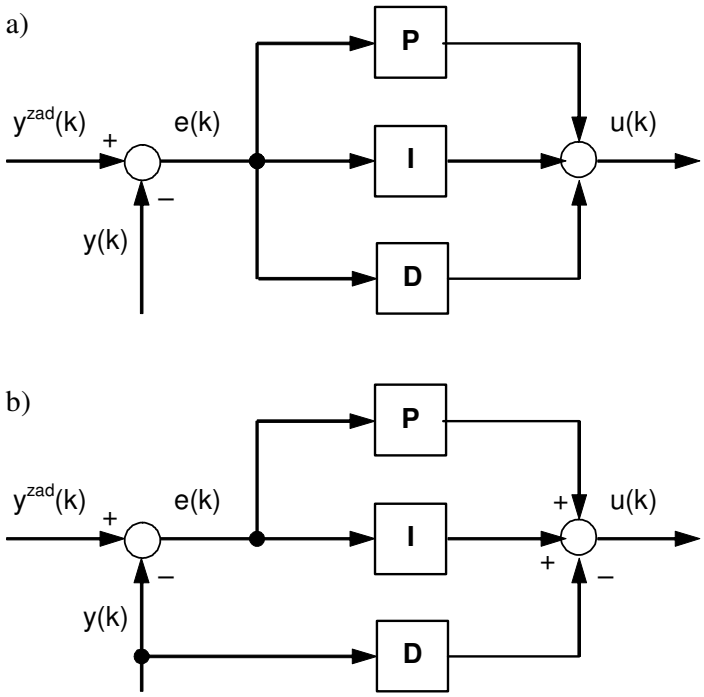
Przypomnijmy, że ze względu na niemożność praktycznego zrealizowania różniczkowania, w praktyce transmitancja regulatora PID jest dana wzorem:

$$R(s) = k_p\left[1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{\tau \cdot s + 1}\right], \tag{2.20}$$

gdzie τ zwykle spełnia warunek $\tau < 0{,}1 \cdot T_d$.

Warto zaznaczyć, że nie zawsze jest potrzebne stosowanie wszystkich członów. Możliwe jest więc stosowanie regulatorów: P (proporcjonalny), I (całkowy), PD (proporcjonalno–różniczkowy), PI (proporcjonalno– całkowy).

Podstawowa struktura regulatora PID została przedstawiona na rys. 2.2a. Na rys. 2.2b przedstawiono schemat regulatora nieróżniczkującego sygnału zadanego, który jest uodporniony na gwałtowne zmiany wartości zadanej.



Rys. 2.2. Struktury regulatora PID
a) różniczkującego uchyb regulacji, b) nieróżniczkującego sygnału zadanego

Przypomnijmy, że w przypadku regulatorów proporcjonalnego i proporcjonalno–różniczkowego zastosowanych do obiektu bez całkowania, występuje niezerowy uchyb ustalony. Jest on tym mniejszy im większe jest wzmocnienie regulatora. Niestety zwiększanie wzmocnienia powoduje zbliżanie się do granicy stabilności układu. Uchyb ustalony można wyeliminować stosując człon całkowy. Niestety działanie całkowe nie może być zbyt intensywne, ponieważ wpływa na zmniejszenie zapasu stabilności. Działanie różniczkowe wpływa stabilizująco na układ regulacji. Wpływ poszczególnych członów regulatora na działanie układu regulacji zostanie zademonstrowany w przykładzie 2.6.

2.2.1. Strojenie regulatorów metodą Zieglera–Nicholsa

Metoda została opracowana na podstawie badań eksperymentalnych w latach czterdziestych ubiegłego stulecia przez Johna G. Zieglera i Nathaniela B. Nicholsa [43]. Aby jej użyć, należy:

— W układzie regulacji z rys. 2.1 zastosować regulator P.
— Następnie wzmocnienie tego regulatora należy stopniowo zwiększać tak długo, aż otrzyma się niegasnące i nierosnące oscylacje.
— Należy zapamiętać wartość otrzymanego **wzmocnienia krytycznego** k_{kr} oraz odczytać **okres oscylacji** T_o.
— Otrzymane wartości wzmocnienia krytycznego k_{kr} i okresu oscylacji T_o służą do wyznaczenia wartości parametrów regulatora na podstawie następujących wzorów:
  – dla regulatora P:

$$k_p = 0{,}5 \cdot k_{kr}; \tag{2.21}$$

  – dla regulatora PI:

$$k_p = 0{,}45 \cdot k_{kr},\ T_i = 0{,}85 \cdot T_o; \tag{2.22}$$

  – dla regulatora PID:

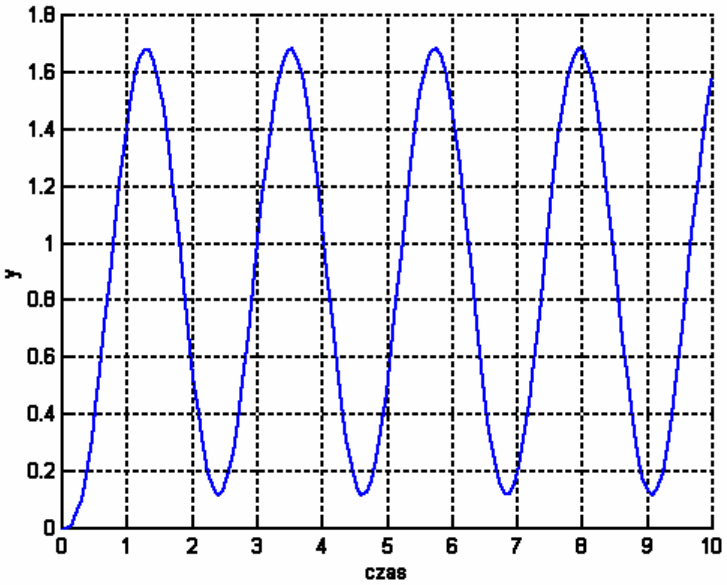
$$k_p = 0{,}6 \cdot k_{kr},\ T_i = 0{,}5 \cdot T_o,\ T_d = 0{,}25 \cdot T_i. \tag{2.23}$$

Parametry regulatora otrzymane w wyniku powyższej procedury należy w razie potrzeby dostroić.

Przykład 2.6

Załóżmy, że obiekt regulacji jest opisany następującą transmitancją:

$$G(s) = \frac{1}{s^3 + 5 \cdot s^2 + 8 \cdot s + 4}. \tag{2.24}$$



Rys. 2.3. Wynik eksperymentu Zieglera–Nicholsa dla przykładowego obiektu ($k_{kr} = 36$)

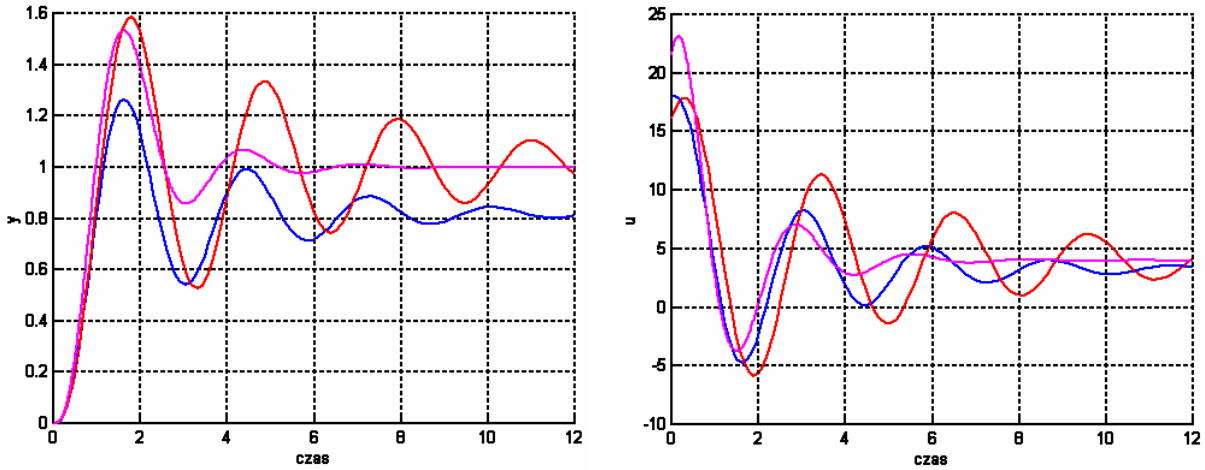
W celu przeprowadzenia strojenia regulatora metodą Zieglera–Nicholsa, układ regulacji zamodelowano w programie Simulink wykorzystując początkowo regulator proporcjonalny ($R(s) = k$). Niegasnące i nierosnące oscylacje uzyskano przy wzmocnieniu $k_{kr} = 36$ (rys. 2.3). Okres tych oscylacji wynosi około $T_o = 2{,}25$. W takim razie nastawy poszczególnych

regulatorów otrzymane ze wzorów (2.21), (2.22) i (2.23) mają wartości zestawione w tabl. 2.1.

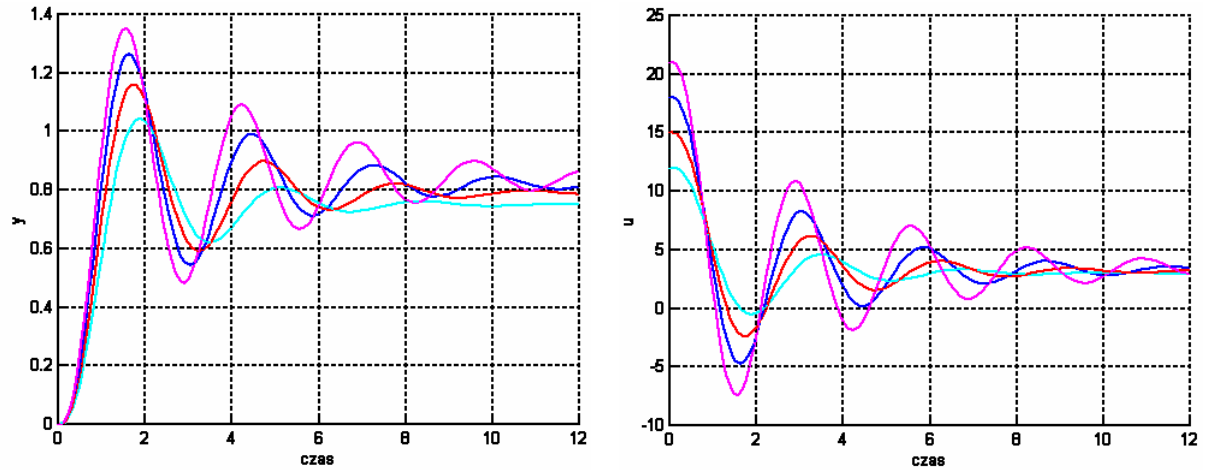
Tabl. 2.1. Nastawy regulatorów otrzymane metodą Zieglera–Nicholsa

	k_p	T_i	T_d
regulator P	18	—	—
regulator PI	16,2	1,9125	—
regulator PID	21,6	1,1250	0,28125

Następnie sprawdzono działanie tak nastrojonych regulatorów w układach regulacji przykładowego obiektu. Porównanie działania regulatorów przedstawiono na rys. 2.4. W przypadku regulatora proporcjonalnego utrzymuje się niezerowy uchyb ustalony (niebieskie przebiegi na rys. 2.4). We wszystkich układach regulacji występują oscylacje. Najdłużej utrzymują się one w układzie z regulatorem proporcjonalno–całkowym (czerwone przebiegi na rys. 2.4). Najlepiej działa regulator proporcjonalno–całkowo–różniczkowy (różowe przebiegi na rys. 2.4) – czas regulacji jest najkrótszy a dzięki części całkowej, uchyb ustalony jest zerowy.



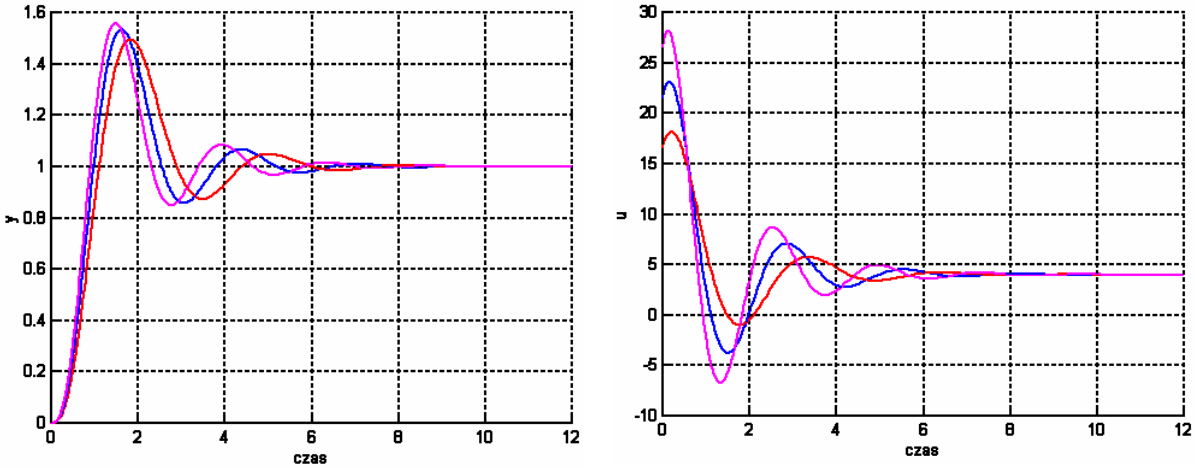
Rys. 2.4. Odpowiedzi układów regulacji z regulatorami P, PI oraz PID na skok wartości zadanej do 1; y – wyjście, u – sterowanie



Rys. 2.5. Odpowiedzi układu regulacji z regulatorem P na skok wartości zadanej do 1; różne wartości wzmocnienia: k_p=12, k_p=15, k_p=18, k_p=21; y – wyjście, u – sterowanie

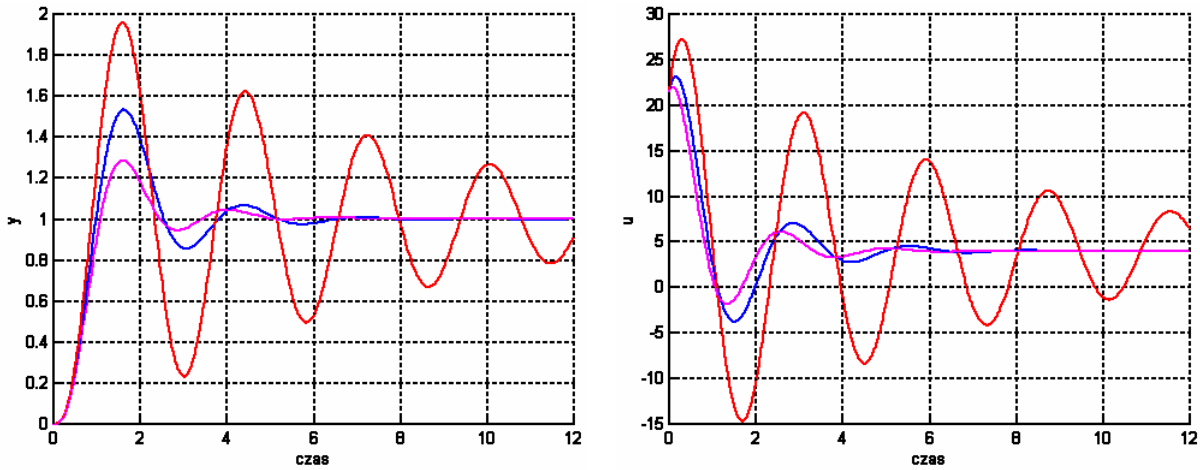
Wpływ zmian wzmocnienia na działanie układu regulacji z regulatorem P przedstawiono na rys. 2.5. Zauważmy, że wraz ze zmniejszeniem wzmocnienia zmniejszają się oscylacje, maleje także czas regulacji. Niestety z drugiej strony, wpływa to na zwiększenie uchybu ustalonego. Odwrotną tendencję, można zaobserwować przy zwiększaniu wzmocnienia.

Przetestowano także, jak zmiana poszczególnych parametrów regulatora PID w stosunku do nastaw z tabl. 2.1 wpływa na działanie układu regulacji. Wpływ zmian wzmocnienia k_p został pokazany na rys. 2.6, pozostałe parametry nie zostały zmienione. Zauważmy, że wzrost wzmocnienia przynosi przyspieszenie regulatora, ale niestety kosztem wzrostu przeregulowania.



Rys. 2.6. Odpowiedzi układu regulacji z regulatorem PID na skok wartości zadanej do 1; różne wartości wzmocnienia: k_p=16,6, k_p=21,6, k_p=26,6; y – wyjście, u – sterowanie

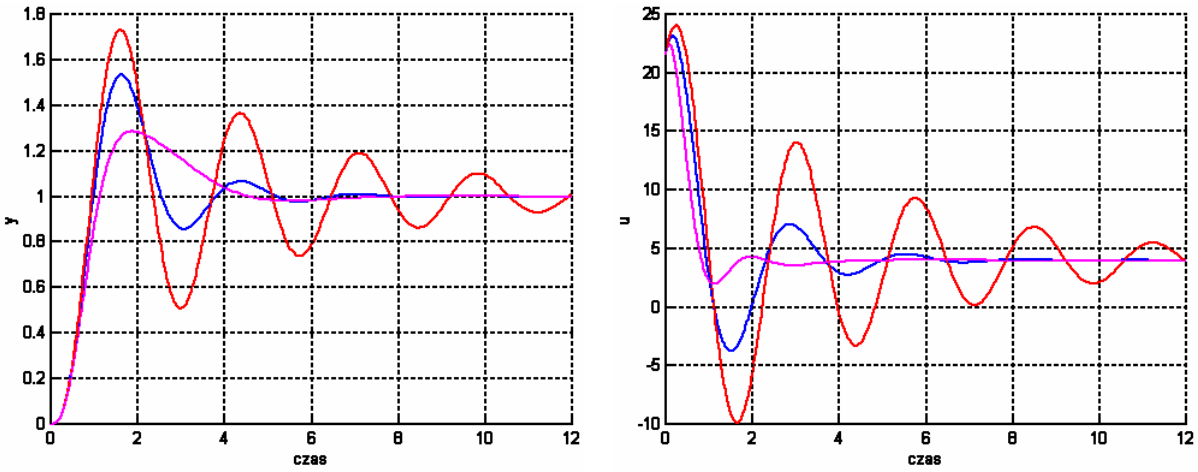
W ramach kolejnego eksperymentu zmieniany był czas zdwojenia T_i, pozostałe parametry regulatora miały wartości z tabl. 2.1. Otrzymane wyniki zostały pokazane na rys. 2.7. W przypadku dwukrotnego zmniejszenia czasu zdwojenia w stosunku do nastaw otrzymanych metodą Zieglera–Nicholsa (z tabl. 2.1) otrzymano silne oscylacje w układzie regulacji (czerwone przebiegi na rys. 2.7). Jest to dobra ilustracja destabilizującego wpływu całkowania. Dwukrotne zwiększenie czasu zdwojenia (różowe przebiegi na rys. 2.7) przyniosło zarówno zmniejszenie przeregulowania, jak i skrócenie czasu regulacji.



Rys. 2.7. Odpowiedzi układu regulacji z regulatorem PID na skok wartości zadanej do 1; różne wartości czasu zdwojenia: T_i=0,5625, T_i=1,1250, T_i=2,2500; y – wyjście, u – sterowanie

Sprawdzono także, co się stanie w wyniku zmian czasu wyprzedzenia T_d podczas, gdy pozostałe parametry regulatora pozostaną bez zmian. Otrzymane odpowiedzi pokazano na rys. 2.8. Dwukrotne zmniejszenie czasu wyprzedzenia przyniosło zwiększenie oscylacji w

układzie (czerwone przebiegi na rys. 2.8). Z kolei dwukrotne zwiększenie czasu wyprzedzenia poskutkowało zmniejszeniem przeregulowania oraz skróceniem (choć stosunkowo małym) czasu regulacji.



Rys. 2.8. Odpowiedzi układu regulacji z regulatorem PID na skok wartości zadanej do 1; różne wartości czasu wyprzedzenia: T_d=0,140625, T_d=0,28125, T_d=0,5625; y – wyjście, u – sterowanie

2.2.2. Dyskretna realizacja regulatora PID

Regulator PID w wersji dyskretnej, otrzymany w wyniku zastosowania całkowania metodą trapezów, jest opisany następującym wzorem [40]:

$$u(k) = u(k-1) + r_1 \cdot e(k) + r_2 \cdot e(k-1) + r_3 \cdot e(k-2), \tag{2.25}$$

gdzie $r_1 = k_p \cdot \left(1 + \frac{T_p}{2 \cdot T_i} + \frac{T_d}{T_p}\right)$, $r_2 = k_p \left(\frac{T_p}{2 \cdot T_i} - 2\frac{T_d}{T_p} - 1\right)$, $r_3 = k_p \frac{T_d}{T_p}$, T_p jest okresem próbkowania.

2.3. Regulator ze sprzężeniem od stanu

Załóżmy, że znany jest model obiektu w postaci dyskretnych, liniowych równań stanu:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \boldsymbol{A} \cdot \boldsymbol{x}(k) + \boldsymbol{B} \cdot \boldsymbol{u}(k), \\ \boldsymbol{y}(k) &= \boldsymbol{C} \cdot \boldsymbol{x}(k) + \boldsymbol{D} \cdot \boldsymbol{u}(k), \end{aligned} \tag{2.26}$$

gdzie $\boldsymbol{x}$ jest wektorem zmiennych stanu, $\boldsymbol{u}$ jest wektorem wejść, $\boldsymbol{y}$ jest wektorem wyjść. Na podstawie powyższych równań można określić transmitancję lub macierz transmitancji (w przypadku układów dynamicznych o wielu wejściach i wielu wyjściach) opisującą układ dynamiczny. Ma ona postać:

$$\boldsymbol{G}(z) = \boldsymbol{C} \cdot \left(z \cdot \boldsymbol{I} - \boldsymbol{A}\right)^{-1} \cdot \boldsymbol{B} + \boldsymbol{D}. \tag{2.27}$$

Przypomnijmy, że stan początkowy układu nazywamy sterowalnym do zera w chwili t_0, jeśli istnieje takie sterowanie dopuszczalne, że w skończonym czasie stan zostanie sprowadzony do zera. Jeśli każdy stan jest sterowalny do zera w chwili t_0, to rozpatrywany układ dynamiczny jest **sterowalny**. Jeśli warunek sterowalności jest spełniony dla dowolnej chwili t_0, to układ jest **całkowicie sterowalny**.

W przypadku liniowych, stacjonarnych układów dynamicznych wystarczy sprawdzić, czy rząd macierzy sterowalności jest równy rzędowi układu. Macierz ta ma postać:

$$\boldsymbol{S} = \left[\boldsymbol{B} \quad \boldsymbol{A}\cdot\boldsymbol{B} \quad \ldots \quad \boldsymbol{A}^{n-1}\cdot\boldsymbol{B}\right], \tag{2.28}$$

gdzie n jest rzędem układu dynamicznego.

Przypomnijmy, że stan układu jest obserwowalny w chwili t_0, jeśli istnieje taka chwila $t_1 \geq t_0$, że znajomość przebiegu zmiennych wyjściowych $\boldsymbol{y}(t)$ oraz zmiennych sterujących $\boldsymbol{u}(t)$ w przedziale $(t_0, t_1]$ umożliwia jednoznaczne określenie stanu $\boldsymbol{x}(t_0)$. Jeśli każdy stan $\boldsymbol{x}(t_0)$ jest obserwowalny w chwili t_0, to mówimy, że dany układ dynamiczny jest **obserwowalny** w chwili t_0. Jeśli układ jest obserwowalny w dowolnej chwili t_0, to mówimy, że jest **całkowicie obserwowalny**.

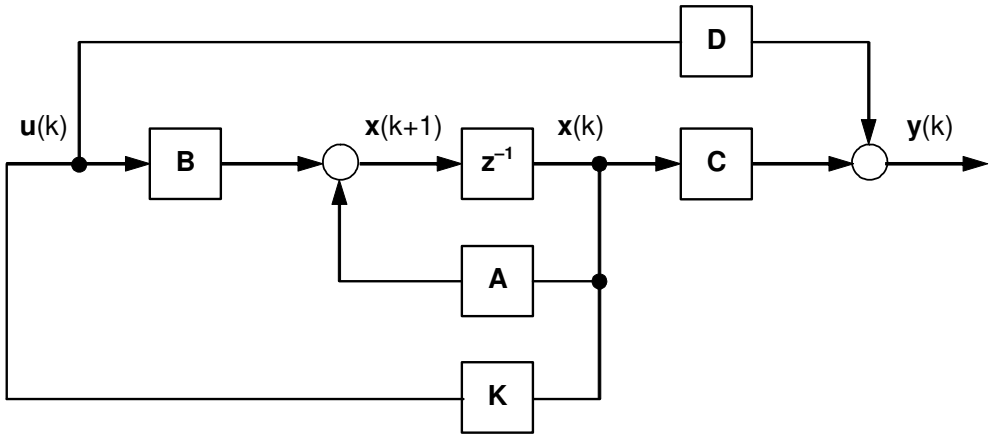
W przypadku liniowych, stacjonarnych układów dynamicznych wystarczy sprawdzić, czy rząd macierzy obserwowalności jest równy rzędowi układu. Macierz ta ma postać:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C}\cdot\boldsymbol{A} \\ \vdots \\ \boldsymbol{C}\cdot\boldsymbol{A}^{n-1} \end{bmatrix}. \tag{2.29}$$

Załóżmy, że obiekt jest sterowalny. Wówczas, można dla niego dobrać regulator ze sprzężeniem od stanu, który ma postać:

$$\boldsymbol{u}(k) = \boldsymbol{K}\cdot\boldsymbol{x}(k), \tag{2.30}$$

gdzie $\boldsymbol{K}$ jest macierzą wzmocnień, wartości elementów której dobiera się tak, aby układ zamknięty miał bieguny ulokowane w wybranych położeniach. Schemat układu z regulatorem ze sprzężeniem od stanu został przedstawiony na rys. 2.9.



Rys. 2.9. Schemat układu regulacji ze sprzężeniem od stanu;
$\boldsymbol{x}$ – wektor stanu, $\boldsymbol{y}$ – wektor wyjść, $\boldsymbol{u}$ – wektor wejść

Zauważmy, że po wstawieniu (2.30) do (2.26), otrzymamy:

$$\begin{aligned} \boldsymbol{x}(k+1) &= \left(\boldsymbol{A} + \boldsymbol{B}\cdot\boldsymbol{K}\right)\cdot\boldsymbol{x}(k), \\ \boldsymbol{y}(k) &= \left(\boldsymbol{C} + \boldsymbol{D}\cdot\boldsymbol{K}\right)\cdot\boldsymbol{x}(k). \end{aligned} \tag{2.31}$$

W takim razie, równanie charakterystyczne układu zamkniętego ma postać:

$$\left|z\cdot\boldsymbol{I} - \left(\boldsymbol{A} + \boldsymbol{B}\cdot\boldsymbol{K}\right)\right| = 0. \tag{2.32}$$

Zauważmy więc, że odpowiednio dobierając elementy macierzy $\boldsymbol{K}$, można ulokować bieguny układu zamkniętego (pierwiastki równania charakterystycznego) w wybranych położeniach.

Przykład 2.7

Rozważmy układ dynamiczny opisany następującymi równaniami:

$$\begin{aligned} x(k+1) &= \begin{bmatrix} -3 & 2 \\ -2 & 2 \end{bmatrix} \cdot x(k) + \begin{bmatrix} 3 \\ 2 \end{bmatrix} \cdot u(k) \\ y(k) &= \begin{bmatrix} 2 & 4 \end{bmatrix} \cdot x(k) \end{aligned} \tag{2.33}$$

Jest to więc układ o jednym wejściu i jednym wyjściu. Zauważmy także, że układ ten jest niestabilny (miejsca zerowe równania charakterystycznego to: z_1=–2 oraz z_2=1). Najpierw zbadajmy sterowalność tego układu. W tym celu konstruujemy macierz sterowalności:

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{A} \cdot \boldsymbol{B} \end{bmatrix} = \begin{bmatrix} 3 & -5 \\ 2 & -2 \end{bmatrix}. \tag{2.34}$$

Macierz sterowalności jest nieosobliwa ($\det(\boldsymbol{S}) = 4$). W takim razie badany układ jest sterowalny. Do tego układu dobierzemy regulator ze sprzężeniem od stanu. Wielomian charakterystyczny układu z regulatorem ze sprzężeniem od stanu jest następujący:

$$\begin{aligned} \left| z \cdot \boldsymbol{I} - (\boldsymbol{A} + \boldsymbol{B} \cdot \boldsymbol{K}) \right| &= \left| \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \left(\begin{bmatrix} -3 & 2 \\ -2 & 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 2 \end{bmatrix} \cdot \begin{bmatrix} k_1 & k_2 \end{bmatrix} \right) \right| = \\ &= \begin{vmatrix} z+3-3 \cdot k_1 & -2-3 \cdot k_2 \\ 2-2 \cdot k_1 & z-2-2 \cdot k_2 \end{vmatrix} = \\ &= z^2 + (1-3 \cdot k_1 - 2 \cdot k_2) \cdot z + (-2+2 \cdot k_1). \end{aligned} \tag{2.35}$$

Załóżmy, że układ zamknięty powinien mieć bieguny z_1 i z_2. W takim razie żądamy, żeby wielomian charakterystyczny miał postać:

$$(z - z_1) \cdot (z - z_2) = z^2 + (-z_1 - z_2) \cdot z + z_1 \cdot z_2 \,. \tag{2.36}$$

Po przyrównaniu (2.35) do (2.36), otrzymujemy następujące warunki:

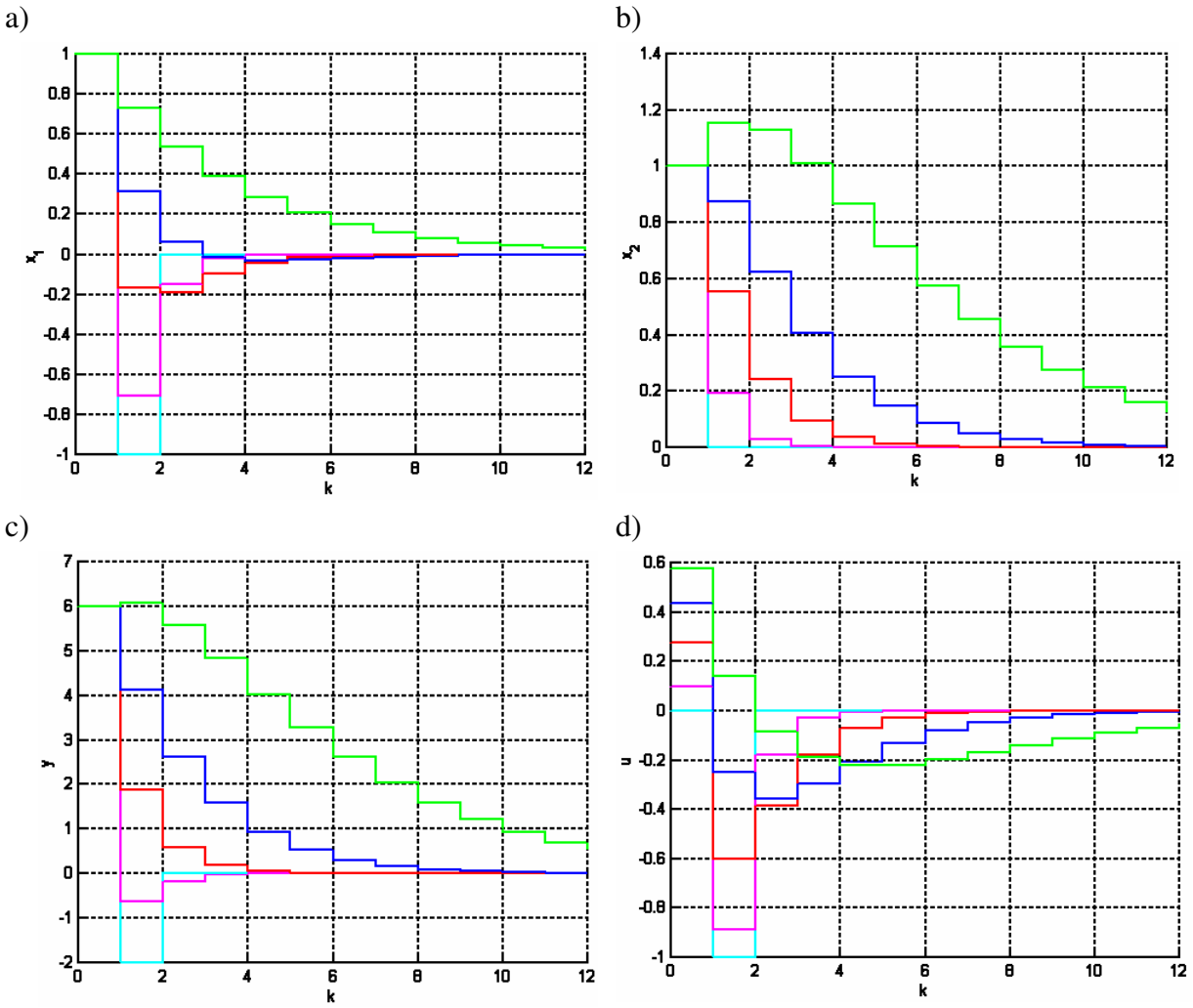
$$\begin{aligned} (-z_1 - z_2) &= (1 - 3 \cdot k_1 - 2 \cdot k_2), \\ z_1 \cdot z_2 &= (-2 + 2 \cdot k_1), \end{aligned} \tag{2.37}$$

a po przekształceniach:

$$\begin{aligned} k_1 &= \frac{1}{2} \cdot z_1 \cdot z_2 + 1, \\ k_2 &= \frac{1}{2} \cdot z_1 + \frac{1}{2} \cdot z_2 - \frac{3}{4} \cdot z_1 \cdot z_2 - 1. \end{aligned} \tag{2.38}$$

Następnie, zasymulowano działanie układów regulacji dla różnie ulokowanych biegunów układu zamkniętego. Otrzymane wyniki przedstawiono na rys. 2.10. Wybierano bieguny rzeczywiste, dodatnie, obydwa położone w tym samym miejscu, kolejno w: 0 (kolor jasno niebieski na rys. 2.10), 0,1 (kolor różowy), 0,3 (kolor czerwony), 0,5 (kolor niebieski), 0,7 (kolor zielony). Przypomnijmy, że aby układ zamknięty był stabilny, należy wybierać bieguny leżące w kole jednostkowym. Zauważmy, że im bliżej początku układu współrzędnych położone są bieguny, tym szybciej działa układ regulacji. Odbywa się to jednak kosztem dużych zmian sterowania. Z drugiej strony, wraz z oddalaniem się od początku układu współrzędnych, otrzymywane przebiegi stają się coraz wolniejsze, co

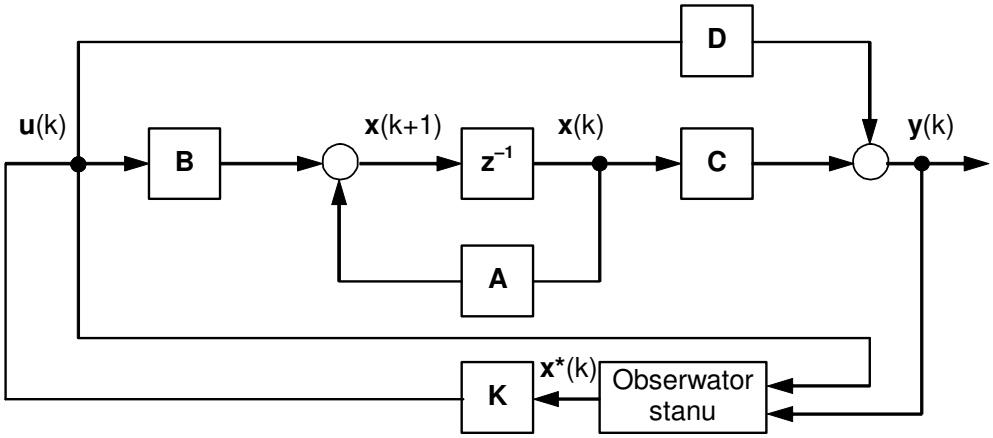
szczególnie dobrze widać w przypadku, gdy bieguny są położone w punkcie 0,7 (zielone przebiegi na rys. 2.10). Warto zauważyć, że pomimo tego, iż układ otwarty był niestabilny, regulator ustabilizował go. Co więcej, zmieniając położenie biegunów łatwo można wpływać na właściwości układu zamkniętego.



Rys. 2.10. Odpowiedzi układów regulacji ze sprzężeniem od stanu na niezerowy warunek początkowy $\boldsymbol{x}_0=[1\ 1]^T$; przebiegi: a) zmiennej stanu x_1, b) zmiennej stanu x_2, c) wyjścia y, d) sterowania u; bieguny układu zamkniętego: $z_1=z_2=0$, $z_1=z_2=0{,}1$, $z_1=z_2=0{,}3$, $z_1=z_2=0{,}5$, $z_1=z_2=0{,}7$

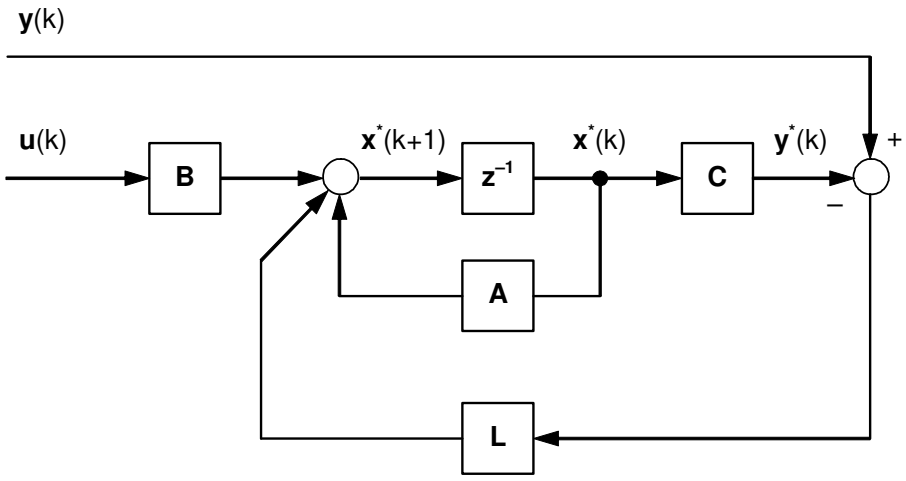
2.3.1. Obserwator stanu

Sprzężenie od stanu można zrealizować w sposób opisany powyżej pod warunkiem, że jest możliwy pomiar wszystkich zmiennych stanu. Jeśli warunek ten nie jest spełniony, wówczas należy opracować tzw. obserwator stanu. Układ regulacji przyjmie wówczas postać jak na rys.2.11.



Rys. 2.11. Schemat układu regulacji ze sprzężeniem od stanu i obserwatorem stanu; $\boldsymbol{x}$ – wektor stanu, $\boldsymbol{y}$ – wektor wyjść, $\boldsymbol{u}$ – wektor wejść, $\boldsymbol{x}^*$ – wektor stanu estymowanego

Opracowanie obserwatora stanu jest możliwe pod warunkiem, że obiekt jest obserwowalny. Załóżmy, że obiekt jest obserwowalny. Załóżmy także, że znamy macierze opisujące układ dynamiczny a także, dla uproszczenia lecz bez utraty ogólności rozważań, że nie ma bezpośredniej zależności między wejściem a wyjściem ($\boldsymbol{D}=\boldsymbol{0}$). Wówczas mamy do czynienia z sytuacją, jak na rys. 2.12.



Rys. 2.12. Struktura obserwatora stanu

Wejściami obserwatora są więc wektory sygnałów wejściowych i wyjściowych układu $\boldsymbol{u}(k)$ i $\boldsymbol{y}(k)$ a uzyskuje się estymatę stanu $\boldsymbol{x}^*(k)$. Zauważmy, że w obserwatorze wykorzystuje się sprzężenie korekcyjne mające zniwelować wpływ błędów modelowania na pracę obserwatora. Zadaniem projektanta jest odpowiednie dobranie parametrów tego sprzężenia (wartości elementów wektora $\boldsymbol{L}$). Przypomnijmy, że na mocy zasady separowalności obserwator można projektować niezależnie od regulatora, któremu dostarczana jest estymata stanu. Równanie opisujące obserwator jest następujące:

$$\begin{aligned}\boldsymbol{x}^*(k+1) &= \boldsymbol{A}\cdot\boldsymbol{x}^*(k)+\boldsymbol{B}\cdot\boldsymbol{u}(k)+\boldsymbol{L}\cdot\left(\boldsymbol{y}(k)-\boldsymbol{y}^*(k)\right)= \\ &= \boldsymbol{A}\cdot\boldsymbol{x}^*(k)+\boldsymbol{B}\cdot\boldsymbol{u}(k)+\boldsymbol{L}\cdot\left(\boldsymbol{y}(k)-\boldsymbol{C}\cdot\boldsymbol{x}^*(k)\right)= \\ &= \left(\boldsymbol{A}-\boldsymbol{L}\cdot\boldsymbol{C}\right)\cdot\boldsymbol{x}^*(k)+\boldsymbol{B}\cdot\boldsymbol{u}(k)+\boldsymbol{L}\cdot\boldsymbol{y}(k)\end{aligned} \tag{2.39}$$

W takim razie otrzymujemy następujące równanie charakterystyczne obserwatora:

$$\left|z \cdot \boldsymbol{I} - \left(\boldsymbol{A} - \boldsymbol{L} \cdot \boldsymbol{C}\right)\right| = 0. \tag{2.40}$$

Podobnie, jak w przypadku regulatora ze sprzężeniem od stanu, elementy wektora $\boldsymbol{L}$ dobiera się w taki sposób, aby miejsca zerowe jego równania charakterystycznego przyjmowały zadane wartości.

Przykład 2.8

Rozważmy układ dynamiczny z przykładu 2.7 opisany równaniami (2.33). Najpierw zbadajmy obserwowalność tego układu. W tym celu konstruujemy macierz obserwowalności:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{C} \cdot \boldsymbol{A} \end{bmatrix} = \begin{bmatrix} 2 & 4 \\ -14 & 12 \end{bmatrix}. \tag{2.41}$$

Macierz obserwowalności jest nieosobliwa ($\det(\boldsymbol{Q}) = 80$). W takim razie badany układ jest obserwowalny. Dla tego układu dobierzemy obserwator. Wielomian charakterystyczny obserwatora jest następujący:

$$\begin{aligned} \left|z \cdot \boldsymbol{I} - \left(\boldsymbol{A} - \boldsymbol{L} \cdot \boldsymbol{C}\right)\right| &= \left|\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \left(\begin{bmatrix} -3 & 2 \\ -2 & 2 \end{bmatrix} - \begin{bmatrix} l_1 \\ l_2 \end{bmatrix} \cdot \begin{bmatrix} 2 & 4 \end{bmatrix}\right)\right| = \\ &= \begin{vmatrix} z+3+2 \cdot l_1 & -2+4 \cdot l_1 \\ 2+2 \cdot l_2 & z-2+4 \cdot l_2 \end{vmatrix} = \\ &= z^2 + \left(2 \cdot l_1 + 4 \cdot l_2 + 1\right) \cdot z + \left(-12 \cdot l_1 + 16 \cdot l_2 - 2\right). \end{aligned} \tag{2.42}$$

Załóżmy, że równanie charakterystyczne obserwatora powinno mieć bieguny z_1 i z_2. W takim razie żądamy, żeby wielomian charakterystyczny miał postać (2.36), jak w przykładzie 2.7.

Po przyrównaniu (2.42) do (2.36), otrzymujemy następujące warunki:

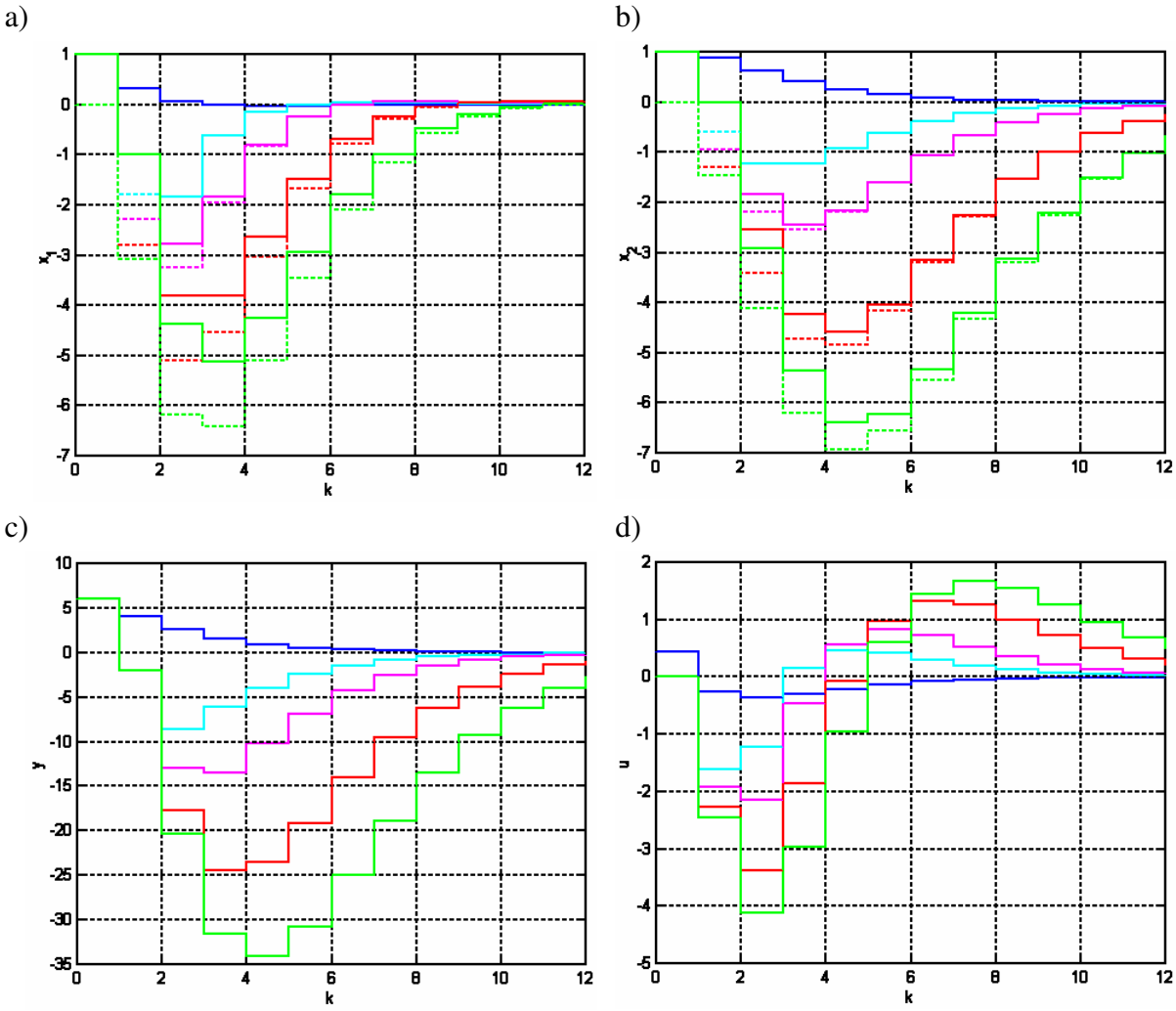
$$\begin{aligned} \left(-z_1 - z_2\right) &= \left(2 \cdot l_1 + 4 \cdot l_2 + 1\right), \\ z_1 \cdot z_2 &= \left(-12 \cdot l_1 + 16 \cdot l_2 - 2\right), \end{aligned} \tag{2.43}$$

a po dalszych przekształceniach:

$$\begin{aligned} l_1 &= \frac{4 \cdot \left(z_1 + z_2\right) + z_1 \cdot z_2 + 6}{-20}, \\ l_2 &= \frac{-6 \cdot \left(z_1 + z_2\right) + z_1 \cdot z_2 - 4}{40}. \end{aligned} \tag{2.44}$$

Zasymulowano działanie układów regulacji z obserwatorem stanu o różnie ulokowanych biegunach (rys. 2.13). Bieguny układu zamkniętego z regulatorem ze sprzężeniem od stanu były ulokowane w z_1=z_2=0,5. Bieguny obserwatora stanu, obydwa ulokowane w tym samym miejscu, były rzeczywiste, dodatnie i umiejscowione w: 0 (kolor jasno niebieski na rys. 2.10), 0,2 (kolor różowy), 0,4 (kolor czerwony), 0,5 (kolor zielony). Oprócz tego, dla porównania zamieszczono przebiegi otrzymane w układzie regulacji z bezpośrednim sprzężeniem od stanu (kolor niebieski na rys. 2.6). Na rys. 2.6a i 2.6b liniami przerywanymi przedstawiono estymaty stanu. W przypadku obserwatora, im bliżej początku układu współrzędnych położone są bieguny, tym szybciej stan estymowany (linie przerywane) osiąga stan rzeczywisty (linie ciągłe). Ponadto, jakość regulacji też jest lepsza (przebiegi bliższe tym

otrzymanym w układzie z bezpośrednim sprzężeniem od stanu). Oddalanie biegunów obserwatora od początku układu współrzędnych, wpływa na dłuższe utrzymywanie się stosunkowo dużego błędu estymacji, a co za tym idzie – pogorszenie jakości regulacji.



Rys. 2.13. Odpowiedzi układów regulacji ze sprzężeniem od stanu i obserwatorem stanu na niezerowy warunek początkowy $\boldsymbol{x}_0=[1\ 1]^T$; przebiegi: a) zmiennej stanu x_1, b) zmiennej stanu x_2, c) wyjścia y, d) sterowania u; bezpośrednie sprzężenie od stanu, sprzężenie od stanu obserwowanego, bieguny obserwatora: $z_1=z_2=0$, $z_1=z_2=0{,}2$, $z_1=z_2=0{,}4$, $z_1=z_2=0{,}5$; stan estymowany – linia przerywana

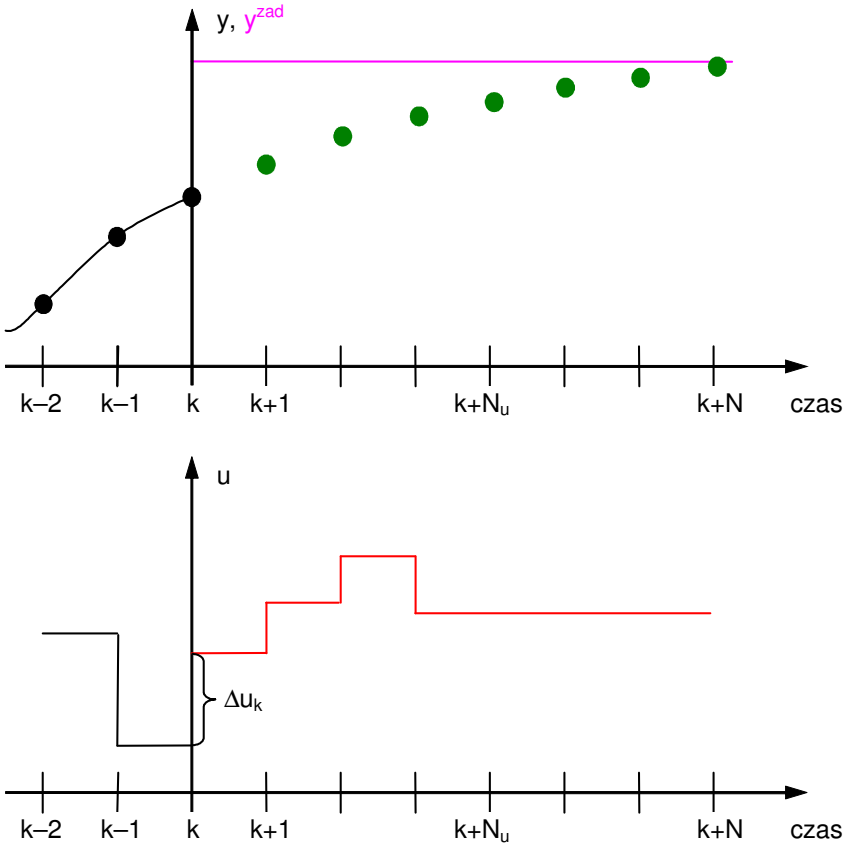
2.4. Algorytmy regulacji predykcyjnej

Pierwsze algorytmy regulacji predykcyjnej z przesuwanym horyzontem zostały opracowane pod koniec lat siedemdziesiątych ubiegłego stulecia. Od tego czasu zdobyły niekwestionowaną pozycję w przemyśle z udokumentowaną dużą liczbą praktycznych zastosowań. Stało się tak ze względu na zalety oferowane przez tego typu algorytmy. Do najważniejszych z tych zalet należą:

- stosunkowo łatwe uwzględnianie ograniczeń istniejących w układzie regulacji nałożonych zarówno na sygnały sterujące, jak i wyjściowe;
- dobra jakość regulacji także w przypadku procesów o trudnej dynamice (opóźnienie, odpowiedź odwrotna);

– stosunkowo łatwe projektowanie regulatorów dla obiektów o wielu wejściach i wielu wyjściach (ang. Multiple Input Multiple Output – MIMO), dla których regulatory predykcyjne oferują niejako „automatyczne” odsprzęganie dzięki wykorzystaniu modelu obiektu regulacji.

Zalety algorytmów predykcyjnych wynikają ze sposobu ich formułowania. W algorytmach tych, zgodnie z ich nazwą, podczas generowania sterowań bierze się pod uwagę nie tylko bieżące informacje o pracy obiektu ale także, korzystając z modelu, przewiduje się zachowanie obiektu wiele chwil do przodu. Ideę działania algorytmów predykcyjnych przedstawia rys. 2.14. W algorytmach predykcyjnych przyjmuje się, na ile chwil do przodu przeprowadza się predykcję; liczba tych chwil jest nazywana horyzontem predykcji N. Zakłada się także, ile zmian sygnału sterującego może nastąpić na horyzoncie predykcji; liczba tych zmian jest nazywana horyzontem sterowania N_u.



Rys. 2.14. Idea działania regulatorów predykcyjnych;
N –horyzont predykcji, N_u – horyzont sterowania, Δu_k – bieżący przyrost sterowania;
przyszłe sterowanie, przewidywane wartości wyjścia, wartość zadana

Sygnał sterujący jest generowany w taki sposób, aby przyszłe zachowanie układu regulacji spełniało przyjęte przez projektanta kryteria. Najczęściej, przyszłe sterowania są wyznaczane w wyniku rozwiązania problemu optymalizacji z następującym wskaźnikiem jakości:

$$\min_{\Delta \boldsymbol{u}} \left\{ \sum_{i=1}^{N} \left(y_{k+ilk}^{zad} - y_{k+ilk} \right)^2 + \sum_{i=0}^{N_u-1} \lambda \cdot \left(\Delta u_{k+ilk} \right)^2 \right\}, \tag{2.45}$$

gdzie y_{k+ilk}^{zad} to trajektoria zadana, y_{k+ilk} to przewidywane wartości wyjścia obiektu dla przyszłych chwil ($k+i$, $i = 1,\ldots, N$) wyznaczane na podstawie modelu obiektu regulacji w

bieżącej chwili k, $\Delta u_{k+i|k}$ ($i = 1,\ldots, N_u-1$) to przyszłe (szukane) przyrosty sterowania, $\lambda \geq 0$ jest parametrem dostrajalnym i określa karę za zmienność sygnału sterującego, $\Delta \boldsymbol{u} = \left[\Delta u_{k|k}, \ldots, \Delta u_{k+N_u-1|k}\right]^T$ jest wektorem zmiennych decyzyjnych.

Zauważmy, że wskaźnik jakości ze wzoru (2.45) oznacza, że regulator powinien generować taki sygnał sterujący, aby przyszłe uchyby regulacji były jak najmniejsze (pierwszy składnik wskaźnika jakości) a jednocześnie sterowanie nie zmieniało się zanadto (drugi składnik wskaźnika jakości). Zwróćmy także uwagę na trajektorię zadaną. We wskaźniku jakości z zadania (2.45) przyjęto, że w każdej chwili z horyzontu predykcji można podać różne wartości na trajektorii zadanej. Często jednak przyjmuje się stałą wartość zadaną dla całego horyzontu predykcji ($y^{zad}_{k+i|k} = y^{zad}_k$). Tak założono w dalszej części niniejszego rozdziału.

Predykcja $y_{k+i|k}$ jest otrzymywana na podstawie modelu obiektu regulacji. Można zastosować różnego rodzaju modele. Różnice między algorytmami predykcyjnymi wynikają głównie z tego, na jakich modelach bazują dane algorytmy. Ze względu na prostotę sformułowania wyznaczanie predykcji oraz sposób formułowania algorytmów predykcyjnych zostanie szczegółowo opisany na przykładzie algorytmu DMC (Dynamic Matrix Control) – jednego z najwcześniej opracowanych i najbardziej rozpowszechnionych.

2.4.1. Algorytm DMC

W algorytmie DMC do predykcji zachowania obiektu jest wykorzystywany model w postaci odpowiedzi skokowej obiektu:

$$y^M_k = \sum_{i=1}^{D-1} s_i \cdot \Delta u_{k-i} + s_D \cdot u_{k-D}, \tag{2.46}$$

gdzie y^M_k to wyjście modelu obiektu regulacji w chwili k, s_i ($i = 1,\ldots, D$) to rzędne odpowiedzi skokowej obiektu, D to horyzont dynamiki obiektu, równy liczbie okresów próbkowania, po upływie których można uznać odpowiedź skokową obiektu za ustaloną, u_{k-D} to wartość sterowania z chwili $k–D$.

Przewidywane wartości wyjścia są obliczane ze wzoru:

$$y_{k+i|k} = \sum_{n=1}^{i} s_n \cdot \Delta u_{k-n+i} + \sum_{n=i+1}^{D-1} s_n \cdot \Delta u_{k-n+i} + s_D \cdot u_{k-D+i} + d_k, \tag{2.47}$$

gdzie $d_k = y_k - y^M_k$ i zakłócenie to przyjmuje się takie samo dla całego horyzontu predykcji (jest to model zakłócenia typu DMC). Zauważmy, że model zakłócenia zawiera zarówno wpływ zakłóceń niemierzalnych, jak i błędów modelowania.

Wzór (2.47) można więc zapisać w następującej postaci:

$$y_{k+i|k} = y_k + \sum_{n=i+1}^{D-1} s_n \cdot \Delta u_{k-n+i} + s_D \cdot \sum_{n=D}^{D+i-1} \Delta u_{k-n+i} - \sum_{n=1}^{D-1} s_n \cdot \Delta u_{k-n} + \sum_{n=1}^{i} s_n \cdot \Delta u_{k-n+i|k}. \tag{2.48}$$

We wzorze (2.48) jedynie ostatni składnik zależy od przyszłych (wyznaczanych przez algorytm) przyrostów sterowania.

Wprowadźmy wektor zawierający przewidywane wartości wyjścia obiektu:

$$\boldsymbol{y} = \left[y_{k+1|k}, \ldots, y_{k+N|k}\right]^T.$$

W takim razie, predykcję wartości wyjścia obiektu można zapisać jako następującą sumę:

$$\boldsymbol{y} = \boldsymbol{y}^0 + \boldsymbol{M} \cdot \Delta\boldsymbol{u}, \tag{2.49}$$

gdzie $\boldsymbol{M} \cdot \Delta\boldsymbol{u}$ jest *odpowiedzią wymuszoną* obiektu regulacji (zależną od przyszłych przyrostów sygnału sterującego), $\boldsymbol{y}^0 = \left[y^0_{k+1|k}, \ldots, y^0_{k+N|k}\right]^{\mathrm{T}}$ jest wektorem nazywanym *odpowiedzią swobodną* obiektu regulacji. Jest tak, ponieważ zawiera on wartości wyjścia, które zostałyby otrzymane, gdyby sygnał sterujący pozostał niezmieniony na całym horyzoncie predykcji. Innymi słowy, zawiera wpływ jedynie przeszłych wartości sygnału sterującego na wyjście obiektu i jest dany następującym wzorem:

$$\boldsymbol{y}^0 = \boldsymbol{y}_k + \boldsymbol{M}^P \cdot \Delta\boldsymbol{u}^P, \tag{2.50}$$

gdzie $\boldsymbol{y}_k = \left[y_k, \ldots, y_k\right]^{\mathrm{T}}$ jest wektorem N–elementowym, $\Delta\boldsymbol{u}^P = \left[\Delta u_{k-1}, \ldots, \Delta u_{k-D+1}\right]^{\mathrm{T}}$ oraz

$$\boldsymbol{M}^P = \begin{bmatrix} s_2 - s_1 & s_3 - s_2 & \cdots & s_{D-1} - s_{D-2} & s_D - s_{D-1} \\ s_3 - s_1 & s_4 - s_2 & \cdots & s_D - s_{D-2} & s_D - s_{D-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{N+1} - s_1 & s_{N+2} - s_2 & \cdots & s_D - s_{D-2} & s_D - s_{D-1} \end{bmatrix}; \tag{2.51}$$

$\boldsymbol{M}$ jest nazywana macierzą dynamiczną (ang. Dynamic Matrix; stąd pochodzi nazwa algorytmu DMC) złożoną z parametrów odpowiedzi skokowej obiektu:

$$\boldsymbol{M} = \begin{bmatrix} s_1 & 0 & \cdots & 0 & 0 \\ s_2 & s_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_N & s_{N-1} & \cdots & s_{N-N_u+2} & s_{N-N_u+1} \end{bmatrix}. \tag{2.52}$$

Algorytm w wersji analitycznej

Jeśli rozwiązuje się problem optymalizacji (2.45) bez ograniczeń, wówczas otrzyma się rozwiązanie analityczne, dane wzorem:

$$\Delta\boldsymbol{u} = \left(\boldsymbol{M}^T \cdot \boldsymbol{M} + \lambda \cdot \boldsymbol{I}\right)^{-1} \cdot \boldsymbol{M}^T \cdot \left(\boldsymbol{y}^{zad} - \boldsymbol{y}^0\right), \tag{2.53}$$

gdzie $\boldsymbol{I}$ jest macierzą jednostkową, $\boldsymbol{y}^{zad} = \left[y_k^{zad}, \ldots, y_k^{zad}\right]^{\mathrm{T}}$ jest wektorem o długości N. W takim razie, wartość pierwszego elementu z ciągu przyszłych sterowań, jest opisana zależnością:

$$\Delta u_{k|k} = \boldsymbol{K}_1 \cdot \left(\boldsymbol{y}^{zad} - \boldsymbol{y}^0\right), \tag{2.54}$$

gdzie $\boldsymbol{K}_1 = [K_{11}, \ldots, K_{1N}]$ jest pierwszym wierszem macierzy $\boldsymbol{K} = \left(\boldsymbol{M}^T \cdot \boldsymbol{M} + \lambda \cdot \boldsymbol{I}\right)^{-1} \cdot \boldsymbol{M}^T$. Zauważmy, że w takim razie jedynie pierwszy wiersz macierzy $\boldsymbol{K}$ musi zostać obliczony. Co więcej, można otrzymać prawo regulacji, które w przypadku algorytmu DMC ma postać:

$$\Delta u_{k|k} = k^e \cdot e_k - \sum_{i=1}^{D-1} k_i^u \cdot \Delta u_{k-i} , \tag{2.55}$$

gdzie $e_k = y_k^{zad} - y_k$ jest uchybem regulacji w bieżącej chwili k, a parametry regulatora DMC można obliczyć ze wzorów:

$$k^e = \sum_{j=1}^{N} K_{1j} , \tag{2.56}$$

$$\left[k_1^u, \ldots, k_{D-1}^u\right] = \boldsymbol{K}_1 \cdot \boldsymbol{M}^P . \tag{2.57}$$

Uwzględnianie ograniczeń w algorytmie analitycznym

Zignorowanie ograniczeń wartości lub przyrostów sterowania istniejących w układzie regulacji, wpływa zwykle bardzo niekorzystnie na jakość działania układu regulacji. W przypadku analitycznych algorytmów regulacji predykcyjnej, można skorzystać z mechanizmu rzutowania sterowań na zbiór ograniczeń. Zaletą tego mechanizmu jest duża prostota, polega on bowiem na odpowiednim modyfikowaniu przyrostów sterowań generowanych przez regulator predykcyjny. Modyfikacje te są opisane następującymi regułami:

dla przyrostów sterowania:

– jeśli $\Delta u_{k|k} < \Delta u_{\min}$, to $\Delta u_{k|k} = \Delta u_{\min}$,

– jeśli $\Delta u_{k|k} > \Delta u_{\max}$, to $\Delta u_{k|k} = \Delta u_{\max}$;

oraz dla wartości sterowania:

– jeśli $u_{k-1} + \Delta u_{k|k} < u_{\min}$, to $\Delta u_{k|k} = u_{\min} - u_{k-1}$,

– jeśli $u_{k-1} + \Delta u_{k|k} > u_{\max}$, to $\Delta u_{k|k} = u_{\max} - u_{k-1}$.

Algorytm w wersji numerycznej

Wskaźnik jakości ze wzoru (2.45) może być także minimalizowany numerycznie, w każdej iteracji algorytmu, przy uwzględnieniu ograniczeń nałożonych na przyszłe wartości: przyrostów sterowania, sterowania i wyjścia obiektu. Wówczas, problem optymalizacji zapisany w postaci macierzowo–wektorowej będzie miał postać:

$$\min_{\Delta \boldsymbol{u}} \left\{ \left(\boldsymbol{y}^{zad} - \boldsymbol{y}\right)^T \cdot \left(\boldsymbol{y}^{zad} - \boldsymbol{y}\right) + \Lambda \cdot \Delta \boldsymbol{u}^T \cdot \Delta \boldsymbol{u} \right\}, \tag{2.58}$$

przy ograniczeniach:

$$\Delta \boldsymbol{u}_{\min} \le \Delta \boldsymbol{u} \le \Delta \boldsymbol{u}_{\max}, \tag{2.59}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u} \le \boldsymbol{u}_{\max}, \tag{2.60}$$

$$\boldsymbol{y}_{\min} \le \boldsymbol{y} \le \boldsymbol{y}_{\max}, \tag{2.61}$$

gdzie $\Delta \boldsymbol{u}_{min}$, $\Delta \boldsymbol{u}_{max}$, $\boldsymbol{u}_{min}$, $\boldsymbol{u}_{max}$, $\boldsymbol{y}_{min}$, $\boldsymbol{y}_{max}$ są wektorami dolnych i górnych ograniczeń wartości odpowiednio: przyrostów sterowania, sterowania i wyjścia obiektu; $\Lambda = \lambda \cdot \boldsymbol{I}$, $\boldsymbol{u} = \left[u_{k|k}, \ldots, u_{k+N_u-1|k}\right]^T = \boldsymbol{u}_{k-1} + \boldsymbol{J} \cdot \Delta \boldsymbol{u}$, $\boldsymbol{u}_{k-1} = \left[u_{k-1}, \ldots, u_{k-1}\right]$ jest wektorem N_u elementowym,

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}. \tag{2.62}$$

Po wykorzystaniu, w zadaniu optymalizacji, predykcji danej wzorem (2.49) otrzymuje się standardowe zadanie optymalizacji kwadratowej:

$$\min_{\Delta \boldsymbol{u}} \left\{ \left(\boldsymbol{y}^{zad} - \boldsymbol{y}^0 - \boldsymbol{M} \cdot \Delta \boldsymbol{u} \right)^T \cdot \left(\boldsymbol{y}^{zad} - \boldsymbol{y}^0 - \boldsymbol{M} \cdot \Delta \boldsymbol{u} \right) + \Lambda \cdot \Delta \boldsymbol{u}^T \cdot \Delta \boldsymbol{u} \right\} , \tag{2.63}$$

przy ograniczeniach:

$$\Delta \boldsymbol{u}_{\min} \le \Delta \boldsymbol{u} \le \Delta \boldsymbol{u}_{\max}, \tag{2.64}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}_{k-1} + \boldsymbol{J} \cdot \Delta \boldsymbol{u} \le \boldsymbol{u}_{\max}, \tag{2.65}$$

$$\boldsymbol{y}_{\min} \le \boldsymbol{y}^0 + \boldsymbol{M} \cdot \Delta \boldsymbol{u} \le \boldsymbol{y}_{\max}, \tag{2.66}$$

W wyniku rozwiązania powyższego zadania, otrzymuje się wektor przyszłych zmian sterowania $\Delta \boldsymbol{u}$, z którego jedynie pierwszy element $\Delta u_{k|k}$ jest używany a następnie optymalizacja jest powtarzana w kolejnej chwili próbkowania.

Przykład 2.9

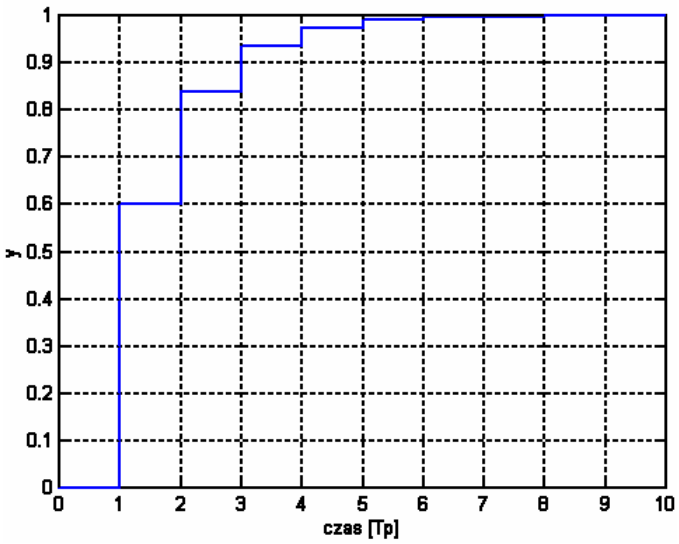
W celu prześledzenia sposobu syntezy algorytmu DMC rozpatrzmy następujący przykład. Załóżmy, że obiekt regulacji jest opisany następującym równaniem różnicowym:

$$y_k^M = 0{,}4 \cdot y_{k-1} + 0{,}6 \cdot u_{k-1} \tag{2.67}$$

Pierwszym krokiem prowadzącym do opracowania algorytmu DMC jest otrzymanie odpowiedzi skokowej obiektu (rys. 2.15). Zauważmy, że odpowiedź ta już w szóstej chwili próbkowania osiąga wartość bliską wartości wzmocnienia obiektu (równego w rozważanym przypadku 1). W związku z tym przyjęto horyzont dynamiki D=6. Rzędne odpowiedzi skokowej, użyte następnie do syntezy regulatora DMC zostały zestawione w tabl. 2.2. Założono także, że horyzont predykcji N=D=6 a horyzont sterowania N_u=3.

Tabl. 2.2. Rzędne odpowiedzi skokowej przykładowego obiektu

s_1	s_2	s_3	s_4	s_5	s_6
0,6000	0,8400	0,9360	0,9744	0,9898	0,9959



Rys. 2.15. Odpowiedź skokowa przykładowego obiektu

W takim razie, macierz wykorzystywana do wyznaczenia odpowiedzi swobodnej ma następującą postać:

$$\boldsymbol{M}^{P} = \begin{bmatrix} 0{,}2400 & 0{,}0960 & 0{,}0384 & 0{,}0154 & 0{,}0061 \\ 0{,}3360 & 0{,}1344 & 0{,}0538 & 0{,}0215 & 0{,}0061 \\ 0{,}3744 & 0{,}1498 & 0{,}0599 & 0{,}0215 & 0{,}0061 \\ 0{,}3898 & 0{,}1559 & 0{,}0599 & 0{,}0215 & 0{,}0061 \\ 0{,}3959 & 0{,}1559 & 0{,}0599 & 0{,}0215 & 0{,}0061 \\ 0{,}3959 & 0{,}1559 & 0{,}0599 & 0{,}0215 & 0{,}0061 \end{bmatrix}. \tag{2.68}$$

Natomiast macierz dynamiczna:

$$\boldsymbol{M} = \begin{bmatrix} 0{,}6000 & 0 & 0 \\ 0{,}8400 & 0{,}6000 & 0 \\ 0{,}9360 & 0{,}8400 & 0{,}6000 \\ 0{,}9744 & 0{,}9360 & 0{,}8400 \\ 0{,}9898 & 0{,}9744 & 0{,}9360 \\ 0{,}9959 & 0{,}9898 & 0{,}9744 \end{bmatrix}. \tag{2.69}$$

Powyższe macierze, mogą następnie posłużyć do sformułowania problemu optymalizacji z ograniczeniami (2.63)–(2.66) w przypadku algorytmu w wersji numerycznej. Można także wyznaczyć parametry analitycznego algorytmu regulacji DMC opisanego wzorem (2.55). Zróbmy to dla parametru λ=1. Zauważmy, że pierwszy wiersz macierzy $\boldsymbol{K}$ ze wzoru (2.54) jest następujący:

$$\boldsymbol{K}_1 = \begin{bmatrix} 0{,}2803 & 0{,}2109 & 0{,}0948 & 0{,}0484 & 0{,}0298 & 0{,}0223 \end{bmatrix}. \tag{2.70}$$

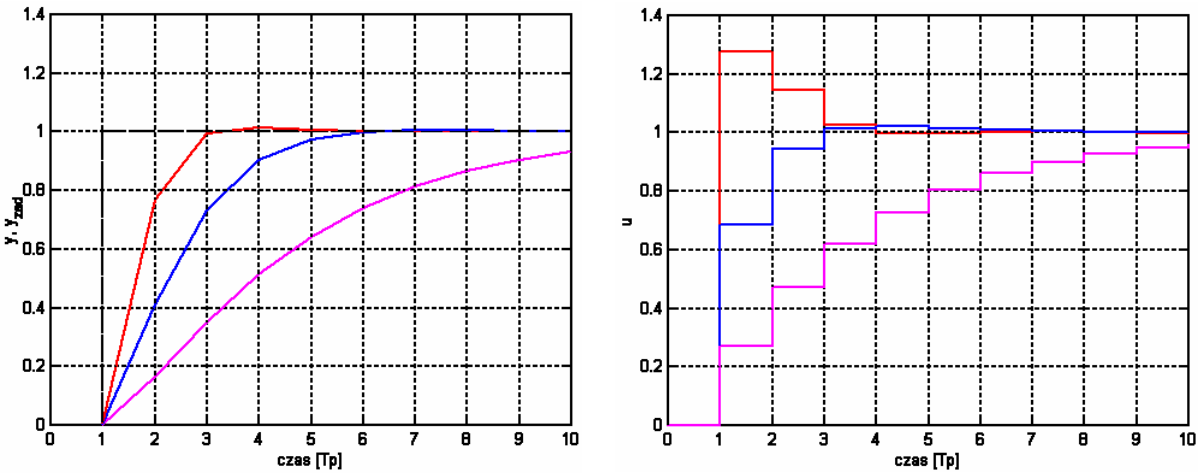
Uwaga: Zauważmy, że w celu wyznaczenia parametrów algorytmu DMC w wersji analitycznej wystarczy obliczyć jedynie elementy pierwszego wiersza macierzy $\boldsymbol{K}$.

Parametry algorytmu DMC w wersji analitycznej można obliczyć, korzystając ze wzorów (2.56) i (2.57). W naszym przykładzie mają one następujące wartości:

$$k^e = 0{,}6864\,, \tag{2.71}$$

$$\left[k_1^u, k_2^u, k_3^u, k_4^u, k_5^u\right] = \left[0{,}2131 \quad 0{,}0851 \quad 0{,}0338 \quad 0{,}0130 \quad 0{,}0042\right]. \tag{2.72}$$

Sprawdźmy, jak na odpowiedzi układu regulacji z regulatorem analitycznym DMC na skok wartości zadanej w chwili próbkowania T_p=1 wpływają różne wartości parametru dostrajalnego λ; odpowiedzi te zostały przedstawione na rys. 2.16. Zauważmy, że im wartość parametru λ jest większa, tym wolniejsze są otrzymane przebiegi. Z drugiej strony, im mniejsza jest wartość tego parametru, tym gwałtowniejsze są zmiany sygnału sterującego oraz tym większe wartości osiąga sygnał sterujący.

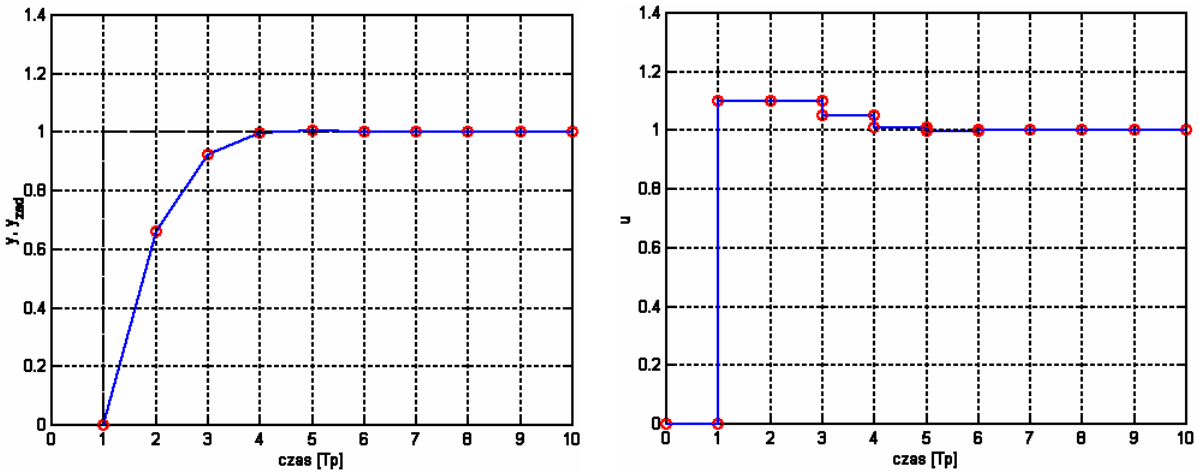


Rys. 2.16. Odpowiedzi układu regulacji z regulatorem DMC na skok wartości zadanej do 1; różne wartości parametru λ: λ=0,1, λ=1, λ=10; *y* – wyjście, *u* – sterowanie

Przetestujmy teraz mechanizm uwzględniania ograniczeń nałożonych na wartości sygnału sterującego w obydwu wersjach algorytmu DMC (analitycznej i numerycznej). Załóżmy, że wartości sterowania są ograniczone od góry:

$$\boldsymbol{u} \le 1{,}1\,. \tag{2.73}$$

Ze względu na charakter otrzymanego przebiegu testy wykonano dla parametru λ=0,1. Otrzymane odpowiedzi przedstawiono na rys. 2.17. Zauważmy, że w przypadku obydwu wersji algorytmu DMC, ograniczenie jest zachowane. Co więcej, wyniki otrzymane z różnymi wersjami algorytmu DMC są takie same (odpowiedzi nałożyły się na siebie). Świadczy to o zadowalającej pracy mechanizmu rzutowania sterowań na zbiór ograniczeń. Co zrozumiałe, ze względu na ograniczenie sygnału sterującego, otrzymany przebieg jest wolniejszy od tego, który został uzyskany przy założeniu, że wartość sterowania nie jest ograniczona od góry.



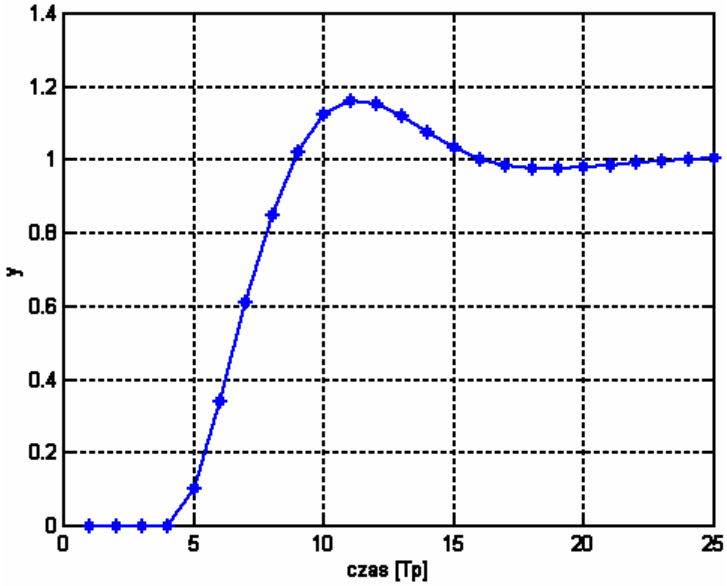
Rys. 2.17. Odpowiedzi układu regulacji z regulatorami DMC w wersji analitycznej i numerycznej na skok wartości zadanej do 1; λ=0,1; *y* – wyjście, *u* – sterowanie

Przykład 2.10

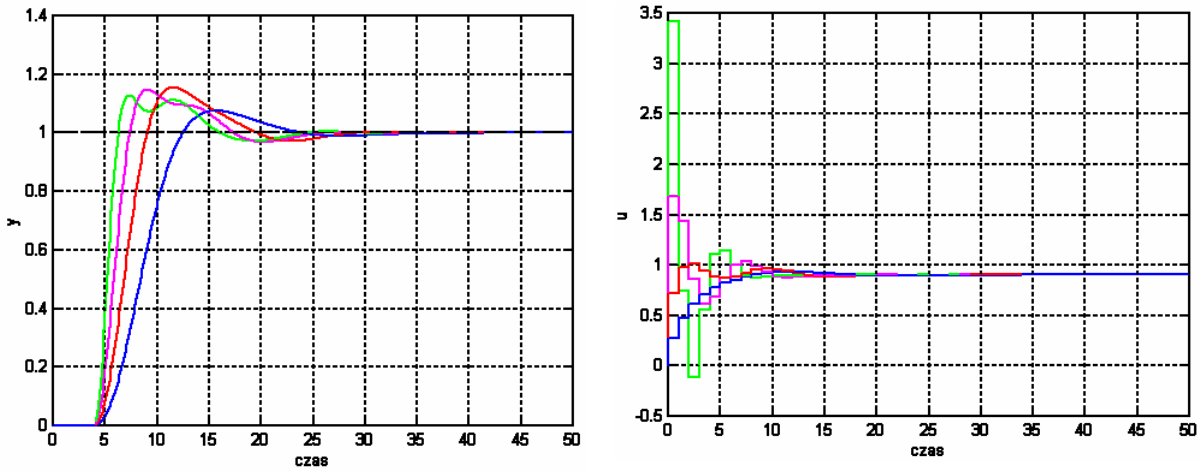
Załóżmy, że obiekt regulacji jest opisany następującą transmitancją:

$$G(s) = \frac{e^{-4s}}{4 \cdot s^2 + 2 \cdot s + 1}. \tag{2.74}$$

Obiekt jest oscylacyjny i ma opóźnienie. Jego odpowiedź skokowa jest przedstawiona na rys. 2.18. Na podstawie tej odpowiedzi został zaprojektowany algorytm DMC. Przyjęto przy tym wartość horyzontu dynamiki D=25. W takiej sytuacji, początkowe wartości horyzontów predykcji i sterowania przyjęto równe odpowiednio: N=25 (równy horyzontowi dynamiki) oraz N_u=12 (równy około połowie horyzontu predykcji).

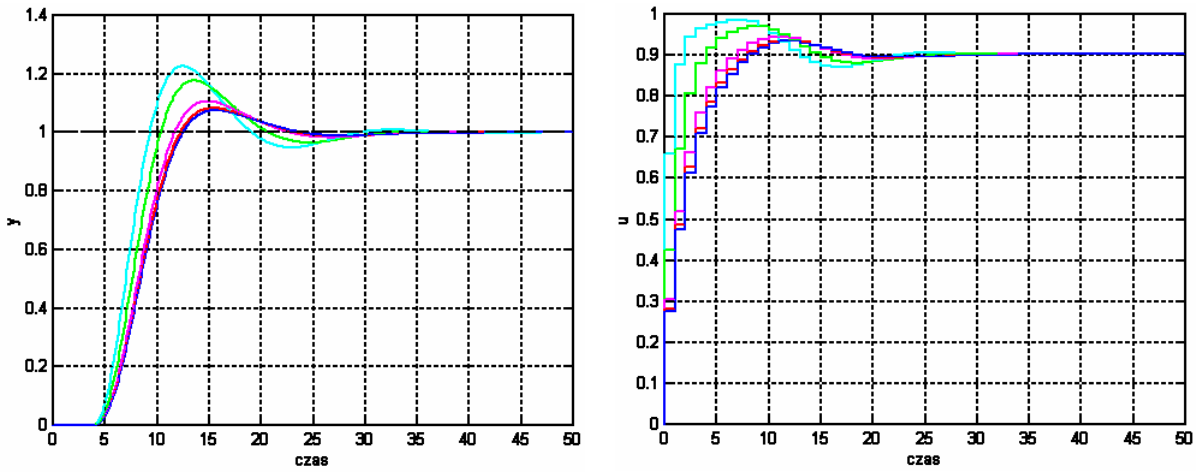


Rys. 2.18. Odpowiedź skokowa przykładowego obiektu



Rys. 2.19. Odpowiedzi układu regulacji z regulatorem DMC na skok wartości zadanej do 1; N=25, N_u=12; różne wartości parametru λ: λ=0,01, λ=0,1, λ=1, λ=10; y – wyjście, u – sterowanie

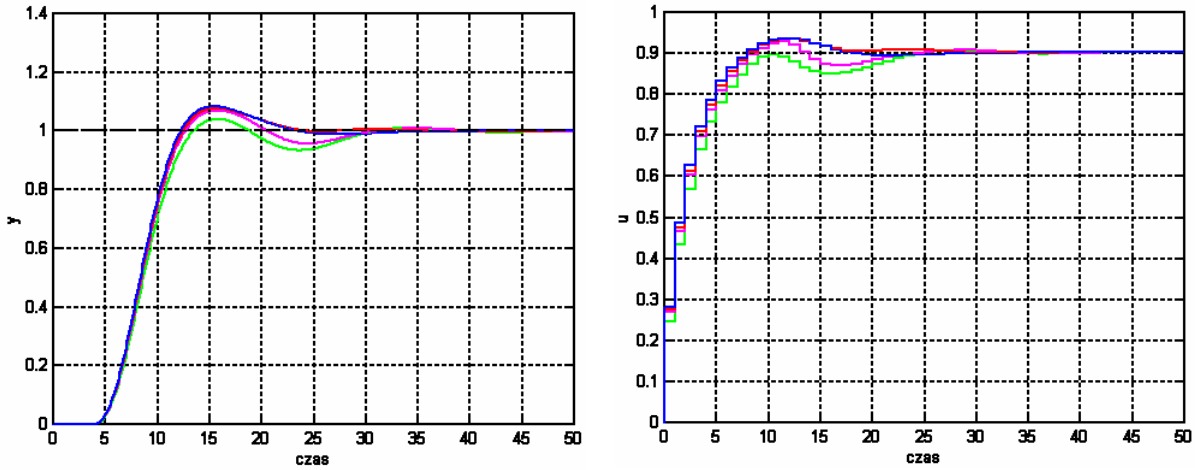
Podobnie, jak w poprzednim przykładzie najpierw przetestowano wpływ różnych wartości parametru dostrajalnego λ na charakter otrzymanych odpowiedzi (rys. 2.19). Tak, jak poprzednio im większa jest wartość parametru, tym wolniejsze przebiegi są otrzymywane przy łagodniejszym przebiegu sterowania. Największa zmienność sygnału sterującego pojawiła się przy najmniejszej wartości parametru λ=0,01 (zielone przebiegi na rys. 2.19). W rozpatrywanym przykładzie można także zaobserwować, że przeregulowanie jest najmniejsze dla największej wartości parametru λ=10 (niebieskie przebiegi na rys. 2.19).



Rys. 2.20. Odpowiedzi układu regulacji z regulatorem DMC na skok wartości zadanej do 1; N=25, λ=10, różne wartości horyzontu sterowania: N_u=1, N_u=2, N_u=4, N_u=6, N_u=12; y – wyjście, u – sterowanie

Kolejne testy miały na celu sprawdzenie, jak na działanie układu regulacji wpływa zmiana horyzontu sterowania; przykładowe odpowiedzi są przedstawione na rys. 2.20. Zauważmy, że w dużym zakresie wartości, przebiegi zmieniają się niewiele. Te otrzymane dla N_u=6 (czerwone przebiegi na rys. 2.20) oraz dla N_u=12 (niebieskie przebiegi na rys. 2.20) są bardzo podobne. Większą różnicę można zaobserwować przy N_u=4 (różowe przebiegi na rys. 2.20). Przeregulowanie rośnie znacząco dla N_u=1 (jasnoniebieskie przebiegi na rys. 2.20) i N_u=2 (zielone przebiegi na rys. 2.20) i w pierwszym przypadku osiąga największą wartość. Sygnał sterujący ma wtedy także najbardziej agresywny charakter.

Warto zaznaczyć, że im krótszy jest horyzont sterowania, tym mniej złożony problem optymalizacji (z ograniczeniami lub bez) musi być rozwiązany, ze względu na mniejszą liczbę zmiennych decyzyjnych. Dlatego warto stosować jak najmniejszą wartość horyzontu sterowania, która jeszcze nie wpływa negatywnie na jakość regulacji.



Rys. 2.21. Odpowiedzi układu regulacji z regulatorem DMC na skok wartości zadanej do 1; N_u=6, λ=10, różne wartości horyzontu predykcji: N=10, N=12, N=16, N=25; y – wyjście, u – sterowanie

Sprawdzono także, jak na jakość sterowania wpływają zmiany horyzontu predykcji; otrzymane przykładowe odpowiedzi zostały przedstawione na rys. 2.21. W tym przypadku dla szerokiego przedziału wartości horyzontu predykcji otrzymywane odpowiedzi są bardzo zbliżone. Jako przykład mogą posłużyć te otrzymane dla N=25 (niebieskie przebiegi na rys. 2.21) oraz N=16 (czerwone przebiegi na rys. 2.21). Większe różnice, choć nie spektakularnie duże pojawiły się dla krótszych wartości horyzontu predykcji N=10 (zielone przebiegi na rys. 2.21) i N=12 (różowe przebiegi na rys. 2.21).

Uwzględnianie zakłócenia mierzalnego

Istnieje możliwość stosunkowo łatwego uwzględniania zakłócenia mierzalnego w algorytmie DMC. Przypomnijmy, że predykcję można zdekomponować na odpowiedź wymuszoną i odpowiedź swobodną. Odpowiedź swobodna zależy od aktualnej wartości wyjścia oraz od składników określających wpływ poprzednich przyrostów sterowania i można powiedzieć, że zawiera ona wszystko to, co już wiemy o przyszłym zachowaniu obiektu. Jeśli chcemy uwzględnić wpływ zakłócenia, które możemy mierzyć bądź oszacować, na rozważany obiekt, to wystarczy do odpowiedzi swobodnej dodać składniki opisujące ten wpływ.

W takim razie przyjmijmy, że model obiektu regulacji (2.46), rozszerzony o dodatkowe składniki, zależne od przyrostów zakłócenia ma postać:

$$y_k^M = \sum_{i=1}^{D-1} s_i \cdot \Delta u_{k-i} + s_D \cdot u_{k-D} + \sum_{i=1}^{D_z-1} s_i^z \cdot \Delta z_{k-i} + s_{D_z}^z \cdot z_{k-D_z}, \qquad (2.75)$$

gdzie s_i^z $(i = 1,\ldots, D_z)$ to rzędne odpowiedzi skokowej obiektu na skok zakłócenia, D_z jest horyzontem dynamiki zakłócenia. Przy założeniu, że nie dysponujemy prognozą przyszłych wartości zakłócenia, przewidywane wartości wyjścia będą opisane wzorem:

$$y_{k+i|k} = \sum_{n=1}^{i} s_n \cdot \Delta u_{k-n+i} + \sum_{n=i+1}^{D-1} s_n \cdot \Delta u_{k-n+i} + s_D \cdot u_{k-D+i} + \\ + s_i^z \cdot \Delta z_k + \sum_{n=i+1}^{D_z-1} s_n^z \cdot \Delta z_{k-n+i} + s_{D_z}^z \cdot z_{k-D_z+i} + d_k \,, \tag{2.76}$$

gdzie składnik $s_i^z \cdot \Delta z_k$ opisuje wpływ bieżącej zmiany zakłócenia mierzalnego. Po rozwinięciu składnika d_k, otrzymamy więc:

$$y_{k+i|k} = y_k + \sum_{n=i+1}^{D-1} s_n \cdot \Delta u_{k-n+i} + s_D \cdot \sum_{n=D}^{D+i-1} \Delta u_{k-n+i} - \sum_{n=1}^{D-1} s_n \cdot \Delta u_{k-n} + \sum_{n=1}^{i} s_n \cdot \Delta u_{k-n+i|k} + \\ + \sum_{n=i+1}^{D_z-1} s_n^z \cdot \Delta z_{k-n+i} + s_{D_z}^z \cdot \sum_{n=D_z}^{D_z+i-1} \Delta z_{k-n+i} - \sum_{n=1}^{D_z-1} s_n^z \cdot \Delta z_{k-n} + s_i^z \cdot \Delta z_k \,. \tag{2.77}$$

Zauważmy, że w powyższym wzorze składnik zależny od przyszłych przyrostów sterowania nie uległ zmianie. W takim razie odpowiedź wymuszona pozostanie bez zmian tak samo, jak macierz dynamiczna. Ponieważ zakładamy, że znamy wartości składników zależnych od zakłócenia mierzalnego, znajdą się one w odpowiedzi swobodnej, która po prostu ulegnie rozszerzeniu. Dzięki skorzystaniu z opisu z użyciem wektorów i macierzy, jest to dobrze widoczne. Odpowiedź swobodna jest teraz bowiem opisana następującym wzorem:

$$\boldsymbol{y}^0 = \boldsymbol{y}_k + \boldsymbol{M}^P \cdot \Delta \boldsymbol{u}^P + \boldsymbol{M}^{zP} \cdot \Delta \boldsymbol{z}^P \,, \tag{2.78}$$

gdzie $\Delta \boldsymbol{z}^P = \left[\Delta z_k, \Delta z_{k-1}, \ldots, \Delta z_{k-D+1}\right]^{\mathrm{T}}$ oraz

$$\boldsymbol{M}^{zP} = \begin{bmatrix} s_1^z & s_2^z - s_1^z & s_3^z - s_2^z & \cdots & s_{D-1}^z - s_{D-2}^z & s_D^z - s_{D-1}^z \\ s_2^z & s_3^z - s_1^z & s_4^z - s_2^z & \cdots & s_D^z - s_{D-2}^z & s_D^z - s_{D-1}^z \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ s_N^z & s_{N+1}^z - s_1^z & s_{N+2}^z - s_2^z & \cdots & s_D^z - s_{D-2}^z & s_D^z - s_{D-1}^z \end{bmatrix}, \tag{2.79}$$

gdzie pierwsza kolumna macierzy $\boldsymbol{M}^{zP}$ opisuje wpływ bieżącego przyrostu zakłócenia (przy założeniu natychmiastowego pomiaru i uwzględnienia zakłócenia).

Dysponując rozszerzoną odpowiedzią swobodną (2.78) można sformułować zadanie optymalizacji dla algorytmu w wersji numerycznej lub skorzystać z analitycznego wzoru służącego do wyznaczania wartości sterowania (w przypadku problemu optymalizacji bez ograniczeń). Zauważmy, że wówczas prawo regulacji analitycznego regulatora DMC ulegnie rozszerzeniu o składnik zależny od zmian zakłócenia:

$$\Delta u_{k|k} = k^e \cdot e_k - \sum_{i=1}^{D-1} k_i^u \cdot \Delta u_{k-i} - \sum_{i=0}^{D_z-1} k_i^z \cdot \Delta z_{k-i} \,, \tag{2.80}$$

gdzie

$$\left[k_0^z, k_1^z, \ldots, k_{D_z-1}^z\right] = \boldsymbol{K}_1 \cdot \boldsymbol{M}^{zP} \,. \tag{2.81}$$

2.4.2. Algorytm GPC

Algorytm GPC jest oparty na modelu obiektu w postaci równania różnicowego:

$$y_k^M = b_1 \cdot y_{k-1} + b_2 \cdot y_{k-2} + \ldots + b_n \cdot y_{k-n} + c_1 \cdot u_{k-1} + c_2 \cdot u_{k-2} + \ldots + c_m \cdot u_{k-m}, \quad (2.82)$$

gdzie b_i (i=1,…,n), c_i (i=1,…,m) są parametrami modelu, y_{k-i} jest wartością wyjścia w chwili k–i, u_{k-i} jest wartością sterowania w chwili k–i. Predykcję przy użyciu tego modelu można przeprowadzić analogicznie, jak w przypadku algorytmu DMC. Przy założeniu modelu zakłóceń typu DMC otrzyma się następujące wyrażenie:

$$\begin{aligned} y_{k+i|k} &= \tilde{b}_1^{i-1} \cdot y_k + \tilde{b}_2^{i-1} \cdot y_{k-1} + \ldots + \tilde{b}_n^{i-1} \cdot y_{k-n+1} + \tilde{b}_{n+1}^{i-1} \cdot y_{k-n} + \\ &+ c_2^{i-1} \cdot \Delta u_{k-1} + \ldots + c_{m-1}^{i-1} \cdot \Delta u_{k-m+2} + c_m^{i-1} \cdot \Delta u_{k-m+1} + \sum_{l=0}^{i-1} c_1^l \cdot \Delta u_{k-l+i-1|k}, \end{aligned} \quad (2.83)$$

gdzie

$$\tilde{b}_j^i = \begin{cases} \tilde{b}_1^{i-1} \cdot \tilde{b}_j + \tilde{b}_{j+1}^{i-1}; & j \geq 1, j \leq n \\ \tilde{b}_1^{i-1} \cdot \tilde{b}_j; & j = n+1 \end{cases}, i = 1,\ldots, N, \; \tilde{b}_j = \begin{cases} b_1 + 1; & j = 1 \\ b_j - b_{j-1}; & j \geq 2, j \leq n, \\ -b_n; & j = n+1 \end{cases}$$

$$c_j^i = \begin{cases} \tilde{b}_1^{i-1} \cdot c_j + c_{j+1}^{i-1}; & j \geq 1, j \leq m-1 \\ \tilde{b}_1^{i-1} \cdot c_j; & j = m \end{cases}, i = 1,\ldots, N,$$

$\tilde{b}_j^0 = \tilde{b}_j$, $c_j^0 = c_j$. Zauważmy, że w powyższym wzorze jedynie ostatni składnik zależy od przyszłych przyrostów sterowania. Odpowiedź swobodna będzie więc opisana następującą zależnością:

$$y_{k+i|k}^0 = \tilde{b}_1^{i-1} \cdot y_k + \tilde{b}_2^{i-1} \cdot y_{k-1} + \ldots + \tilde{b}_{n+1}^{i-1} \cdot y_{k-n} + c_2^{i-1} \cdot \Delta u_{k-1} + \ldots + c_m^{i-1} \cdot \Delta u_{k-m+1}. \quad (2.84)$$

W takim razie wektor przewidywanych wartości wyjścia opisuje wzór (2.49), a więc ten sam, co w przypadku algorytmu DMC. Należy jednak podkreślić, że tym razem odpowiedź swobodna jawnie zależy od przeszłych wartości wyjścia. Macierz dynamiczna będzie zaś opisana wzorem:

$$\boldsymbol{M} = \begin{bmatrix} c_1^0 & 0 & \cdots & 0 & 0 \\ c_1^1 & c_1^0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_1^{N-1} & c_1^{N-2} & \cdots & c_1^{N-N_u+1} & c_1^{N-N_u} \end{bmatrix}. \quad (2.85)$$

Przy czym zauważmy, że współczynniki c_1^l; l = 0,…, N–1 są równe rzędnym odpowiedzi skokowej obiektu regulacji. W takim razie macierz ta jest taka sama, jak w algorytmie DMC.

Algorytm w wersji analitycznej

Jeśli rozpatrywany jest problem optymalizacji (2.45) bez ograniczeń, wówczas jego rozwiązanie jest opisane wzorem (2.53) tak, jak w przypadku algorytmu DMC, zaś wartość pierwszego elementu z ciągu przyszłych sterowań, jest opisana zależnością (2.54). Tym razem jednak, ze względu na różnicę w odpowiedzi swobodnej, prawo regulacji jest opisane zależnością:

$$\Delta u_{k|k} = k^e \cdot e_k + \sum_{i=1}^{m-1} k_i^u \cdot \Delta u_{k-i} + \sum_{j=1}^{n+1} k_j^y \cdot y_{k-j+1}\,, \tag{2.86}$$

gdzie parametry regulatora k^e, k_i^u $(i = 1,\ldots,m-1)$, k_j^y $(j = 1,\ldots,n+1)$ można obliczyć, korzystając z odpowiednio przekształconego wzoru (2.54) oraz z odpowiedzi swobodnej wyznaczanej na podstawie wzoru (2.84).

W przypadku zaistnienia potrzeby uwzględnienia ograniczeń sygnału sterującego, istniejących w układzie regulacji, można skorzystać z tego samego mechanizmu rzutowania sterowań na zbiór ograniczeń, co w przypadku algorytmu DMC w wersji analitycznej.

Algorytm w wersji numerycznej

Analogicznie, jak w przypadku algorytmu DMC jest możliwe sformułowanie zadania optymalizacji z ograniczeniami (2.59)–(2.61). Ze względu na postać predykcji, otrzymanej na podstawie modelu liniowego, zadanie to będzie standardowym zadaniem optymalizacji kwadratowej, które należy rozwiązywać w każdej iteracji algorytmu. Zauważmy także, że odpowiedź swobodna potrzebna do sformułowania tego zadania może zostać obliczona nie tylko na podstawie wzoru (2.84) ale także iteracyjnie, przy użyciu modelu (2.82).

Przykład 2.11

Rozpatrzmy ten sam obiekt, co w przykładzie 2.10. W celu zaprojektowania algorytmu GPC, na podstawie transmitancji (2.74), otrzymano następujący model w postaci równania różnicowego (założono okres próbkowania T_p=1):

$$y_k^M = 1{,}414 \cdot y_{k-1} - 0{,}6065 \cdot y_{k-2} + 0{,}1044 \cdot u_{k-5} + 0{,}08828 \cdot u_{k-6}\,. \tag{2.87}$$

Macierz dynamiczna jest taka sama, jak w przypadku algorytmu DMC. Pozostaje więc wyznaczyć wartości odpowiedzi swobodnej dla kolejnych chwil z horyzontu predykcji. Można przy tym skorzystać ze wzoru (2.84) lub zastosować metodę iteracyjną. Prześledźmy teraz użycie tej drugiej metody. Zauważmy, że zakłócenie typu DMC jest dane wzorem:

$$d_k = y_k - y_k^M = y_k - 1{,}414 \cdot y_{k-1} + 0{,}6065 \cdot y_{k-2} - 0{,}1044 \cdot u_{k-5} - 0{,}08828 \cdot u_{k-6}\,. \tag{2.88}$$

Następnie zakłócenie to jest wykorzystywane podczas obliczania wartości elementów odpowiedzi swobodnej. Dla pierwszej chwili z horyzontu predykcji otrzyma się więc:

$$\begin{aligned} y_{k+1|k}^0 = y_{k+1}^M + d_k = {} & 1{,}414 \cdot y_k - 0{,}6065 \cdot y_{k-1} + 0{,}1044 \cdot u_{k-4} + 0{,}08828 \cdot u_{k-5} + \\ & + y_k - 1{,}414 \cdot y_{k-1} + 0{,}6065 \cdot y_{k-2} - 0{,}1044 \cdot u_{k-5} - 0{,}08828 \cdot u_{k-6}. \end{aligned} \tag{2.89}$$

W takim razie, ostatecznie otrzyma się:

$$y_{k+1|k}^0 = 2{,}414 \cdot y_k - 2{,}0205 \cdot y_{k-1} + 0{,}6065 \cdot y_{k-2} + 0{,}1044 \cdot u_{k-4} - 0{,}01612 \cdot u_{k-5} - 0{,}08828 \cdot u_{k-6}. \tag{2.90}$$

Dla kolejnej chwili z horyzontu predykcji:

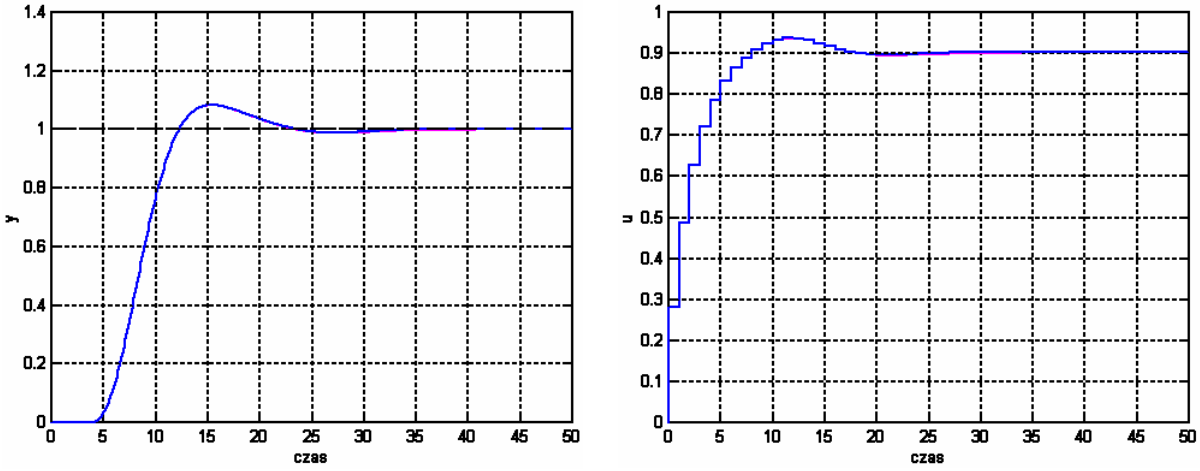
$$\begin{aligned} y_{k+2|k}^0 = y_{k+2}^M + d_k = {} & 1{,}414 \cdot y_{k+1|k}^0 - 0{,}6065 \cdot y_k + 0{,}1044 \cdot u_{k-3} + 0{,}08828 \cdot u_{k-4} + \\ & y_k - 1{,}414 \cdot y_{k-1} + 0{,}6065 \cdot y_{k-2} - 0{,}1044 \cdot u_{k-5} - 0{,}08828 \cdot u_{k-6}. \end{aligned} \tag{2.91}$$

Zauważmy, że w celu obliczenia wartości odpowiedzi swobodnej dla chwili k+2 z horyzontu predykcji jest potrzebna znajomość wartości wyjścia dla chwili k+1. Zamiast tej wartości, we wzorze (2.91) użyto już wyznaczonego elementu odpowiedzi swobodnej $y_{k+1|k}^0$. W takim razie:

$$y^0_{k+2|k} = 1{,}414 \cdot y^0_{k+1|k} + 0{,}3935 \cdot y_k - 1{,}414 \cdot y_{k-1} + 0{,}6065 \cdot y_{k-2} + 0{,}1044 \cdot u_{k-3} + 0{,}08828 \cdot u_{k-4} - 0{,}1044 \cdot u_{k-5} - 0{,}08828 \cdot u_{k-6}. \quad (2.92)$$

Wyżej opisana procedura jest stosowana także dla dalszych elementów odpowiedzi swobodnej, dla kolejnych chwil z horyzontu predykcji. Zauważmy, że procedura ta jest łatwa do zaimplementowania.

Macierz dynamiczna i otrzymana odpowiedź swobodna zostały użyte do sformułowania algorytmu GPC. Odpowiedź układu regulacji z algorytmem GPC została porównana z odpowiedzią otrzymaną w układzie regulacji z algorytmem DMC. Założono następujące wartości parametrów: horyzont predykcji N=25, horyzont sterowania N_u=6, parametr dostrajalny λ=10. Otrzymana odpowiedź układu regulacji z algorytmem GPC praktycznie pokryła się z odpowiedzią otrzymaną w układzie regulacji z algorytmem DMC (rys 2.22). Małą różnicę można dostrzec w przebiegach wyjścia między 35 a 40 chwilą symulacji oraz w przebiegach sterowania w okolicy 25 chwili symulacji. Taki rezultat jest jak najbardziej uzasadniony i świadczy o użyciu właściwego modelu w postaci odpowiedzi skokowej obiektu regulacji podczas projektowania algorytmu DMC.



Rys. 2.22. Porównanie odpowiedzi układów regulacji z regulatorami GPC i DMC na skok wartości zadanej do 1; N=25, N_u=6, λ=10; y – wyjście, u – sterowanie

Uwzględnianie zakłócenia mierzalnego

Podobnie, jak w przypadku algorytmu DMC, w algorytmie GPC można w stosunkowo łatwy sposób uwzględnić wpływ zakłócenia mierzalnego na obiekt regulacji. W tym celu model obiektu w postaci równania różnicowego rozszerza się otrzymując:

$$y^M_k = b_1 \cdot y_{k-1} + b_2 \cdot y_{k-2} + \ldots + b_n \cdot y_{k-n} + c_1 \cdot u_{k-1} + c_2 \cdot u_{k-2} + \ldots + c_m \cdot u_{k-m} + e_1 \cdot z_{k-1} + e_2 \cdot z_{k-2} + \ldots + e_o \cdot z_{k-o}, \quad (2.93)$$

gdzie e_i $(i = 1,\ldots, o)$, są parametrami modelu opisującymi wpływ zakłócenia na wyjście obiektu, z_{k-i} są wartościami zakłócenia mierzalnego. Zauważmy, że w szczególnie łatwy sposób nowy model można wykorzystać w iteracyjnej metodzie wyznaczania odpowiedzi swobodnej.

Uwaga: Dzięki skorzystaniu z analitycznych wzorów opisujących odpowiedź swobodną, wpływ zakłócenia można także uwzględnić w podobny sposób, jak w algorytmie DMC, rozszerzając odpowiedź swobodną o składniki opisujące wpływ zakłócenia na obiekt.

2.4.3. Wykorzystanie modeli nieliniowych w algorytmach predykcyjnych

W algorytmie predykcyjnym jest możliwe bezpośrednie skorzystanie z nieliniowego modelu procesu do wyznaczania predykcji w zadaniu optymalizacji (2.58)–(2.61). W predykcji tej jednak zależność przewidywanych wartości wyjścia od przyszłych sterowań (zmiennych decyzyjnych) jest nieliniowa. W związku z tym otrzymany problem optymalizacji, który musi być rozwiązywany w każdej iteracji algorytmu, nie jest już problemem kwadratowym i w ogólności nie jest wypukły. Wiąże się to z kolei z możliwością pojawienia się problemów podczas szukania rozwiązania przez procedurę optymalizacji, w skrajnym przypadku prowadzących do problemu ze znalezieniem rozwiązania. Ponadto, czas potrzebny na znalezienie sterowania nie może zostać zawczasu określony.

W celu uniknięcia komplikacji algorytmu, wynikającej z konieczności rozwiązywania zadania optymalizacji nieliniowej w każdej iteracji, opracowano algorytmy wykorzystujące linearyzację modelu nieliniowego przeprowadzaną w każdej iteracji i prowadzącą do otrzymania predykcji zależnej liniowo od zmiennych decyzyjnych. Następnie, w każdej iteracji algorytmu formułuje się problem optymalizacji (2.58)–(2.61), który dzięki użyciu predykcji zależnej liniowo od zmiennych decyzyjnych jest problemem optymalizacji kwadratowej z liniowymi ograniczeniami, łatwym do rozwiązania przy użyciu powszechnie dostępnych procedur optymalizacji. Algorytmów wykorzystujących linearyzację modelu nieliniowego jest kilka. Różnica między nimi wynika z różnych sposobów przeprowadzania predykcji zachowania procesu; w dalszej części rozdziału zostały przedstawione dwa najczęściej stosowane spośród omawianych algorytmów. Realizacje tych algorytmów oparte na modelach rozmytych i neuronowych zostały natomiast omówione dokładniej w dalszych rozdziałach.

Algorytmy typu MPC–SL

Najprostsze spośród algorytmów wykorzystujących linearyzację nieliniowego modelu procesu są algorytmy typu MPC–SL (z sukcesywną linearyzacją). W algorytmach tych, do prognozowania wpływu zarówno przeszłych (odpowiedź swobodna) jak i przyszłych (odpowiedź wymuszona) wartości sterowania na zachowanie obiektu w przyszłości, stosuje się w danej iteracji algorytmu ten sam model liniowy, otrzymany w aktualnym punkcie pracy.

W każdej iteracji algorytmu MPC–SL jest powtarzana następująca sekwencja czynności:

1. Na podstawie modelu nieliniowego jest wyznaczany model liniowy dla bieżących wartości wyjścia obiektu i sterowania. Następne kroki algorytmu są w praktyce takie same, jak w algorytmach bazujących na liniowych modelach obiektu, tzn.

2. Otrzymany zlinearyzowany model obiektu służy do wyznaczenia zestawu współczynników odpowiedzi skokowych, na podstawie których jest generowana macierz dynamiczna.

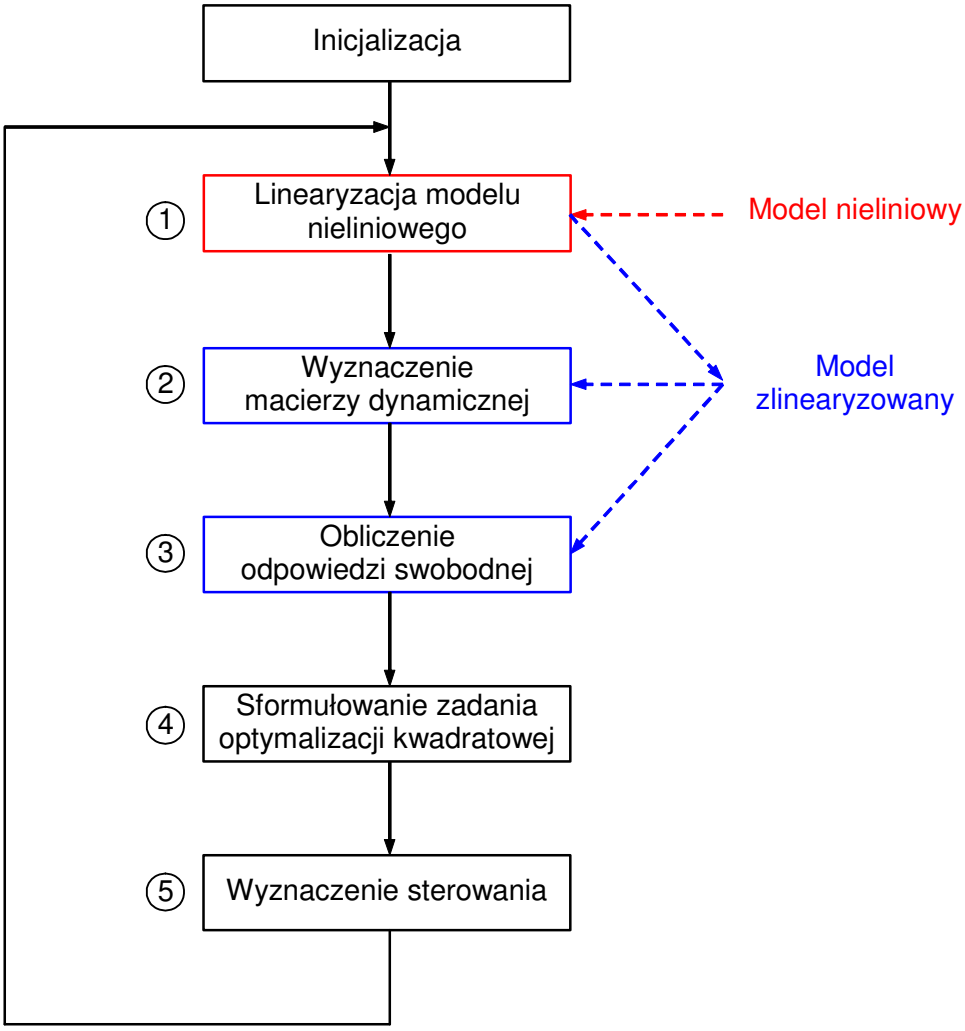
3. Zlinearyzowany model obiektu jest wykorzystywany do otrzymania odpowiedzi swobodnej obiektu.

4. Uzyskana odpowiedź swobodna wraz z macierzą dynamiczną są używane do sformułowania zadania optymalizacji kwadratowej.

5. Zadanie optymalizacji jest rozwiązywane i na podstawie otrzymanego wyniku jest generowane sterowanie. Następnie algorytm wraca do punktu 1 w kolejnej iteracji.

Powyższy algorytm został zaprezentowany na rys. 2.23 z zaznaczeniem modeli obiektu wykorzystywanych podczas jego działania. Jakość regulacji oferowana przez algorytm typu MPC–SL, zastosowany do obiektu nieliniowego, jest zwykle znacznie lepsza od jakości oferowanej przez algorytmy bazujące na modelach liniowych. Należy jednak zauważyć, że jeśli obiekt regulacji jest silnie nieliniowy, wówczas niedokładność modeli otrzymywanych w

wyniku linearyzacji może być znacząca. Dlatego opracowane zostały algorytmy, w których predykcja jest otrzymywana w inny sposób niż w algorytmach typu MPC–SL. Zabieg ten przynosi zwykle zauważalną poprawę działania układu regulacji.

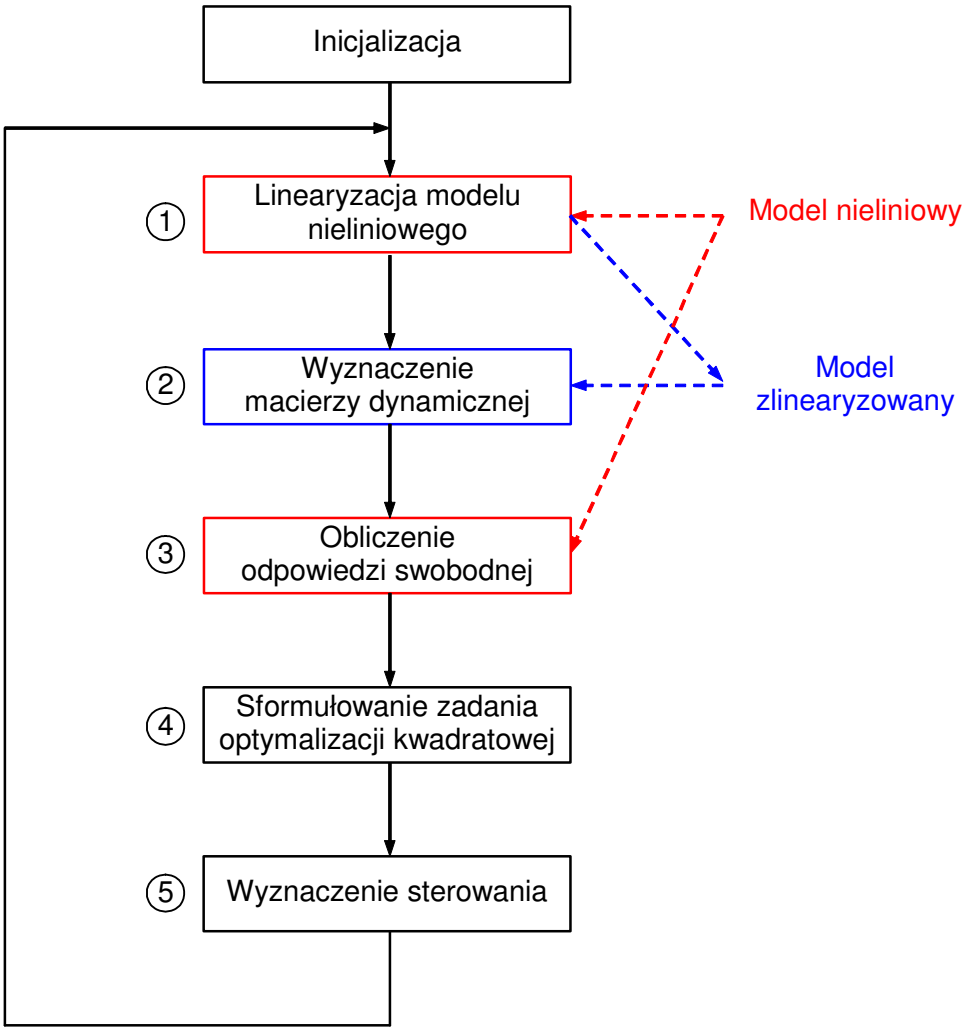


Rys. 2.23. Algorytm typu MPC–SL

Algorytmy typu MPC–NPL

Algorytmy typu MPC–NPL (z nieliniową predykcją i linearyzacją) są praktycznie nieco zmodyfikowanymi algorytmami typu MPC–SL. Modyfikacja ta polega na użyciu nieliniowego modelu obiektu do obliczania odpowiedzi swobodnej. Dzięki takiemu podejściu, jedynie wpływ przyszłych przyrostów sterowań (zmiennych decyzyjnych) jest liczony na podstawie macierzy dynamicznej otrzymanej przy użyciu modelu zlinearyzowanego. W takim razie punkty: 1, 2, 4 i 5 algorytmu pozostają takie same, jak w algorytmie MPC–SL. Zmiana następuje natomiast w punkcie 3, który jest w algorytmie MPC–NPL następujący:

3. Odpowiedź swobodna obiektu jest generowana przy użyciu nieliniowego modelu obiektu. Obliczenia tej odpowiedzi dla chwili k+1 i dalszych dokonuje się, używając dostępnych pomiarów zmiennych wyjściowych oraz informacji o przeszłych wartościach sterowania, rekurencyjnie, tzn. jako wartości wyjść obiektu w przyszłości, używane są już wyznaczone elementy odpowiedzi swobodnej.

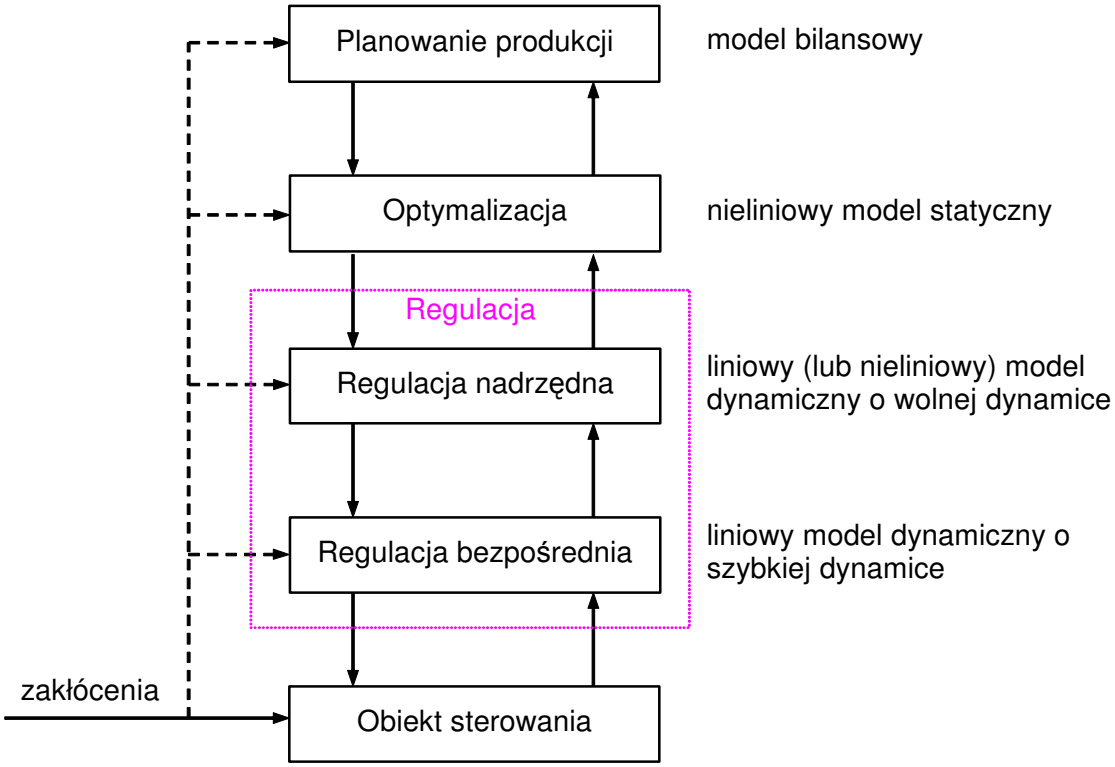


Rys. 2.24. Algorytm typu MPC–NPL

Algorytm typu MPC–NPL został zaprezentowany na rys. 2.24. Warto przy tym nadmienić, że algorytm ten przynosi zwykle zauważalną poprawę jakości działania układu regulacji w stosunku do przypadku, gdy korzysta się z algorytmu typu MPC–SL. Jest tak, ponieważ, jak to zaznaczono na rys. 2.24 część predykcji (odpowiedź swobodna) jest wyznaczana na podstawie modelu nieliniowego. Zauważmy, że w takim razie im mniejsze będą przyszłe przyrosty sterowania, tym dokładniejsza predykcja zostanie otrzymana. Z tego spostrzeżenia korzystają wspomniane wcześniej, inne algorytmy wykorzystujące linearyzację nieliniowego modelu obiektu. Są one jednak bardziej skomplikowane niż algorytm MPC–NPL a zwykle nie przynoszą spektakularnej poprawy jakości regulacji w stosunku do algorytmu MPC–NPL. Zainteresowany czytelnik więcej informacji na ten temat znajdzie w książce [38].

2.5. Hierarchiczna struktura sterowania

Ze względu na dużą złożoność obiektów przemysłowych, do ich sterowania stosuje się hierarchiczną (warstwową) strukturę sterowania (rys. 2.25). Dzięki temu podstawowy problem sterowania jest dekomponowany na szereg powiązanych ze sobą, mniej skomplikowanych zadań cząstkowych. Dekompozycja ta upraszcza projektowanie układu sterowania, jego realizację oraz nadzorowanie. W hierarchicznej strukturze sterowania można wyodrębnić następujące cztery warstwy: planowania produkcji, optymalizacji, regulacji nadrzędnej oraz regulacji bezpośredniej.



Rys. 2.25. Hierarchiczna struktura sterowania

Warstwa planowania produkcji ma za zadanie ustalić cele, parametry i ograniczenia produkcji dla warstwy optymalizacji. Okres interwencji tej warstwy jest rzędu doby (do kilku dni). Celem jej działania jest maksymalizacja efektów ekonomicznych w skali całego przedsiębiorstwa, którego jednym z elementów jest obiekt sterowania. W warstwie tej stosuje się modele bilansowe, zwykle liniowe. Najczęściej planowanie jest wykonywane przez człowieka (osobę odpowiedzialną za jego wykonanie). Podczas tego procesu uwzględniane są ekonomiczne uwarunkowania towarzyszące pracy obiektu. Planowanie wykonuje się więc biorąc pod uwagę m.in.: zmiany cen produktów oraz surowców potrzebnych do produkcji, wielkość produkcji, jakość produktów.

Warstwa optymalizacji jest odpowiedzialna za wyznaczanie pożądanych wartości zadanych dla regulatorów pracujących w niższych warstwach. W warstwie tej stosuje się modele statyczne procesu, zwykle nieliniowe. Okres interwencji jest rzędu godzin, choć może ona także zostać uruchomiona w przypadku wykrycia istotnych zmian zakłóceń. Warstwa optymalizacji pracuje pod nadzorem operatora a wartości zadane są otrzymywane w wyniku rozwiązania problemu optymalizacji, mającego zwykle następującą postać:

$$\min_{\boldsymbol{y}^{zad},\boldsymbol{u}^{zad}} Q\left(\boldsymbol{y}^{zad},\boldsymbol{u}^{zad}\right), \tag{2.94}$$

przy ograniczeniach:

$$\boldsymbol{u}_{\min}^{zad} \le \boldsymbol{u}^{zad} \le \boldsymbol{u}_{\max}^{zad}, \tag{2.95}$$

$$\boldsymbol{y}_{\min}^{zad} \le \boldsymbol{y}^{zad} \le \boldsymbol{y}_{\max}^{zad}, \tag{2.96}$$

$$\boldsymbol{y}^{zad} = F\left(\boldsymbol{u}^{zad},\tilde{\boldsymbol{w}}\right), \tag{2.97}$$

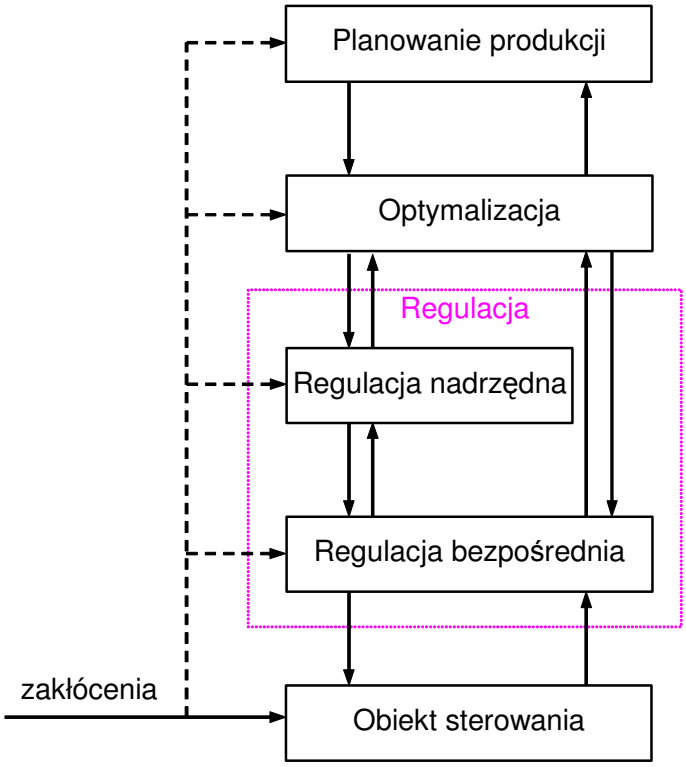
gdzie $F : R^{n_u} \times R^{n_w} \to R^{n_y}$ jest (zwykle nieliniowym) modelem statycznym procesu, n_u jest liczbą zmiennych sterujących, n_w jest liczbą zmiennych zakłócających, n_y jest liczbą zmiennych wyjściowych, $\boldsymbol{y}^{zad}$ jest wektorem wartości zadanych dla regulatorów, $\boldsymbol{u}^{zad}$ jest wektorem odpowiadających im wartości sterowań, wyznaczonych na podstawie modelu statycznego obiektu $\boldsymbol{y}^{zad} = F\left(\boldsymbol{u}^{zad}, \tilde{\boldsymbol{w}}\right)$, $\tilde{\boldsymbol{w}}$ jest estymatą zakłóceń, $\boldsymbol{u}_{\min}^{zad}, \boldsymbol{u}_{\max}^{zad}$ są wektorami odpowiednio: dolnych i górnych ograniczeń sygnałów sterujących, $\boldsymbol{y}_{\min}^{zad}, \boldsymbol{y}_{\max}^{zad}$ są wektorami odpowiednio: dolnych i górnych ograniczeń wyjść obiektu. $Q\left(\boldsymbol{y}^{zad}, \boldsymbol{u}^{zad}\right)$ jest wskaźnikiem jakości, zwykle natury ekonomicznej i przeważnie liniowym względem zmiennych wyjściowych i sterujących. W wyniku rozwiązania problemu optymalizacji (2.94–2.97), otrzymuje się wartości zadane przekazywane regulatorom w niższych warstwach.

Model statyczny procesu może mieć różną postać. Warto jednak zauważyć, że w przypadku zastosowania w tej roli sieci neuronowych lub modeli rozmytych z jednej strony zyskuje się możliwość stosunkowo łatwej identyfikacji obiektu a z drugiej strony, problem optymalizacji (2.94–2.97) może zostać uproszczony. Co więcej, dzięki zapewnieniu różniczkowalności tych modeli, co nie jest trudne, do rozwiązania powyższego problemu optymalizacji można użyć metod gradientowych.

Warstwa regulacji nadrzędnej (nazywana także warstwą regulacji zaawansowanej – ang. advanced control lub regulacji z ograniczeniami – ang. constraint control) ma za zadanie sterowanie wolniejszych zmiennych decydujących o jakości produkcji, generując wartości zadane dla regulatorów warstwy regulacji bezpośredniej. Zwykle w warstwie tej jest realizowana zaawansowana regulacja kluczowych zmiennych procesu w niewielkiej odległości od ograniczeń. Jest tak, ponieważ wartości zadane wyznaczane przez warstwę optymalizacji leżą najczęściej na ograniczeniach (punkt optymalny jest zwykle położony na ograniczeniach). Czyli, im bliżej ograniczeń będzie prowadzony proces, tym większe będą korzyści ekonomiczne. Należy jednak podkreślić, że złamanie ograniczeń może prowadzić do strat ponieważ produkt nie będzie spełniał podstawowych wymagań dotyczących jego jakości i w konsekwencji nie zostanie sprzedany.

Względy opisane wyżej sprawiły, że regulatory warstwy regulacji nadrzędnej umożliwiają uwzględnianie ograniczeń nałożonych zarówno na zmienne sterujące jak i wyjściowe. W warstwie regulacji nadrzędnej stosuje się więc zwykle algorytmy regulacji predykcyjnej w wersjach numerycznych. W algorytmach tych używa się modeli dynamicznych liniowych lub nieliniowych, zwykle o wolniejszej dynamice niż w przypadku regulatorów warstwy regulacji bezpośredniej. Okres interwencji warstwy regulacji nadrzędnej jest rzędu minut.

Warstwa regulacji nadrzędnej została wyodrębniona stosunkowo niedawno. Stało się to wtedy, gdy moc obliczeniowa urządzeń sterujących umożliwiła implementację bardziej skomplikowanych algorytmów regulacji, jak omówione w rozdz. 2.4 algorytmy regulacji predykcyjnej w wersjach numerycznych (z zadaniem optymalizacji liniowo–kwadratowej rozwiązywanym w każdej iteracji algorytmu). Warstwa regulacji nadrzędnej nie musi jednak występować w każdym układzie sterowania. Może tej warstwy nie być w ogóle, a może także generować wartości zadane tylko dla części regulatorów z warstwy regulacji bezpośredniej, zaś pozostałe wartości zadane mogą być przekazywane bezpośrednio z warstwy optymalizacji, jak to zostało pokazane na rys. 2.26.



Rys. 2.26. Hierarchiczna struktura sterowania ze zmniejszonym zasięgiem warstwy regulacji nadrzędnej

Warstwa regulacji bezpośredniej (i zabezpieczeń) ma za zadanie przede wszystkim zapewnienie bezpieczeństwa procesu. W przypadku obiektów niestabilnych, regulatory tej warstwy stabilizują go. Okres interwencji warstwy regulacji bezpośredniej jest rzędu ułamków sekund. Warstwa ta generuje sygnały sterujące, przekazywane bezpośrednio do obiektu sterowania (stąd nazwa tej warstwy). Wykorzystywane modele dynamiczne są zwykle liniowe (rzadziej nieliniowe) o szybkiej dynamice. W warstwie regulacji bezpośredniej, ze względu na swoją niezawodność i prostotę, dominują regulatory typu PID w różnych odmianach zależnych od potrzeb, np. uwzględniające pomiar zakłócenia (ang. feedforward) czy z nieliniowo zmiennym wzmocnieniem (ang. gain scheduling). Mogą zostać także zastosowane regulatory rozmyte, jak te opisane w rozdz. 4. Innymi regulatorami, stosowanymi w szczególności w przypadku obiektów z dużymi opóźnieniami są algorytmy typu IMC (ang. Internal Model Control) oraz algorytmy regulacji predykcyjnej.

3. Sztuczne sieci neuronowe

3.1. Neuron i jego model

Inspiracją do opracowania sztucznych sieci neuronowych, zwanych krótko sieciami neuronowymi, były badania systemów nerwowych istot żywych. Transmisja sygnałów wewnątrz systemu nerwowego jest bardzo skomplikowanym procesem chemiczno-elektrycznym [29]. W skrócie, przekazywanie impulsów elektrycznych między komórkami nerwowymi polega na wydzielaniu pod wpływem nadchodzących bodźców specjalnych substancji chemicznych, zwanych neuromediatorami. Oddziałują one następnie na błonę komórki, powodując zmianę jej potencjału elektrycznego. Wielkość zmiany potencjału zależy od ilości neuromediatora. Sygnały wejściowe doprowadzone są do komórek za pomocą synaps, przy czym poszczególne synapsy różnią się wielkością oraz możliwościami gromadzenia neuromediatora. Dlatego też ten sam impuls docierający do komórki za pośrednictwem różnych synaps może powodować jej silniejsze lub słabsze pobudzenie. Stopień pobudzenia komórki zależy od sumarycznej ilości neuromediatora wydzielonego we wszystkich synapsach.

Jak wynika z powyższego opisu, poszczególnym wejściom komórki można przypisać współczynniki liczbowe odpowiadające ilości neuromediatora. W modelu matematycznym neuronu sygnały wejściowe muszą być mnożone przez te współczynniki. Dzięki temu można uwzględnić wpływ poszczególnych sygnałów wejściowych. Wspomniane współczynniki mogą być zarówno dodatnie jak i ujemne. Pierwsze z nich działają pobudzająco, natomiast drugie hamująco, co powoduje utrudnienie pobudzenia komórki przez inne sygnały.

Na skutek uwolnienia odpowiedniej ilości neuromediatora następuje pobudzenie komórki. Jeżeli pobudzenie przekracza pewien próg (próg uaktywnienia komórki nerwowej), to sygnał wyjściowy gwałtownie rośnie, jest on przesyłany do innych neuronów połączonych z daną komórką. Sygnał ten jest niezależny od stopnia przekroczenia progu.

Funkcje pełnione przez poszczególne neurony są bardzo proste (mnożenie, sumowanie, generacja lub nie impulsu wyjściowego), ale należy przypomnieć, że są one połączone w sieci – sieci neuronowe. Sieci te są bardzo skomplikowane. Przyjmuje się, że ludzki mózg liczy około 10^{11} neuronów [36]. Ogromna jest również liczba połączeń między neuronami. Sieci neuronowe organizmów żywych mają dwie ważne zalety: są odporne na błędy pojedynczych neuronów oraz charakteryzują się dużą szybkością działania, istniejące komórki nerwowe przetwarzają informację równolegle. Dzięki temu, codzienne czynności, takie jak na przykład rozpoznawanie przez człowieka obrazu i mowy lub podejmowanie decyzji odbywa się w ciągu milisekund. W chwili obecnej, pomimo ciągłego postępu elektroniki, nie udało się opracować równie efektywnych urządzeń.

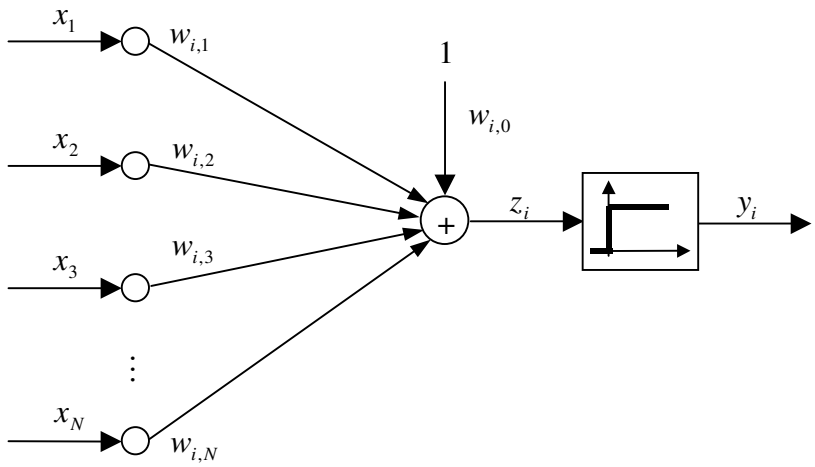
Kierując się zasadą działania komórki nerwowej opracowano wiele różnych modeli matematycznych zachodzących zjawisk. W 1943 roku W. S. McCulloch i W. H. Pitts opracowali model komórki nerwowej przedstawiony na rys. 3.1. Model ten do dzisiaj stanowi podstawę wielu sieci neuronowych. Zgodnie z opisaną zasadą działania komórki nerwowej przyjęto, że sygnały wejściowe $x_1,\ldots,x_N$ sumowane są z odpowiednimi współczynnikami $w_{i,j}$, zwanymi wagami, a następnie poddane działaniu pewnej nieliniowej funkcji aktywacji φ. Sygnał wyjściowy *i*-tego neuronu obliczany jest ze wzoru

$$y_i = \varphi(z_i) = \varphi\left(\sum_{j=1}^{N} w_{i,j}x_j + w_{i,0}\right) \tag{3.1}$$

przy czym z_i oznacza sumę ważoną sygnałów wejściowych. Ponieważ komórka nerwowa może być uaktywniona (generować na wyjściu impuls) lub nie, w modelu McCullocha-Pittsa przyjęto funkcję aktywacji w postaci skoku jednostkowego.

$$\varphi(z_i) = \begin{cases} 1 & \text{dla} \quad z_i > 0 \\ 0 & \text{dla} \quad z_i \le 0 \end{cases} \tag{3.2}$$

Współczynniki $w_{i,j}$ reprezentują połączenia synaptyczne. Gdy $w_{i,j} > 0$ synapsa działa pobudzająco, gdy $w_{i,j} < 0$ synapsa działa hamująco, natomiast zerowa wartość wagi świadczy o braku połączenia. Waga $w_{i,0}$ jest tzw. polaryzacją neuronu, skojarzona jest ona ze stałym wejściem, na które podaje się sygnał jednostkowy.



Rys. 3.1. Model neuronu McCullocha-Pittsa

Warto podkreślić, że choć przedstawiony powyżej model komórki nerwowej jest stosunkowo prosty, to jednak stanowi on podstawę większości współcześnie stosowanych sieci neuronowych. W pochodzącej z 1962 roku pracy F. Rosenblatta model neuronu z binarną funkcją aktywacji wykorzystano do przedstawienia teorii dynamicznych systemów neuronowych modelujących mózg.

Dobór funkcji aktywacji neuronów determinuje wybór algorytmów uczących sieci, dzięki którym można dobrać wartości wag do aktualnie rozwiązywanego problemu, np. aproksymacji. Funkcja aktywacji zastosowana w modelu neuronu McCullocha-Pittsa nie jest ciągła. Zwykle wykorzystuje się funkcje ciągłe, dzięki czemu do uczenia sieci można zastosować bardzo skuteczne gradientowe metody optymalizacji. Struktura najczęściej wykorzystywanego neuronu sigmoidalnego jest właściwie analogiczna jak modelu McCullocha-Pittsa, ale jego funkcja aktywacji jest ciągła. Można zastosować sigmoidalną funkcję unipolarną

$$\varphi(z_i) = \frac{1}{1 + e^{-\beta z_i}} \tag{3.3}$$

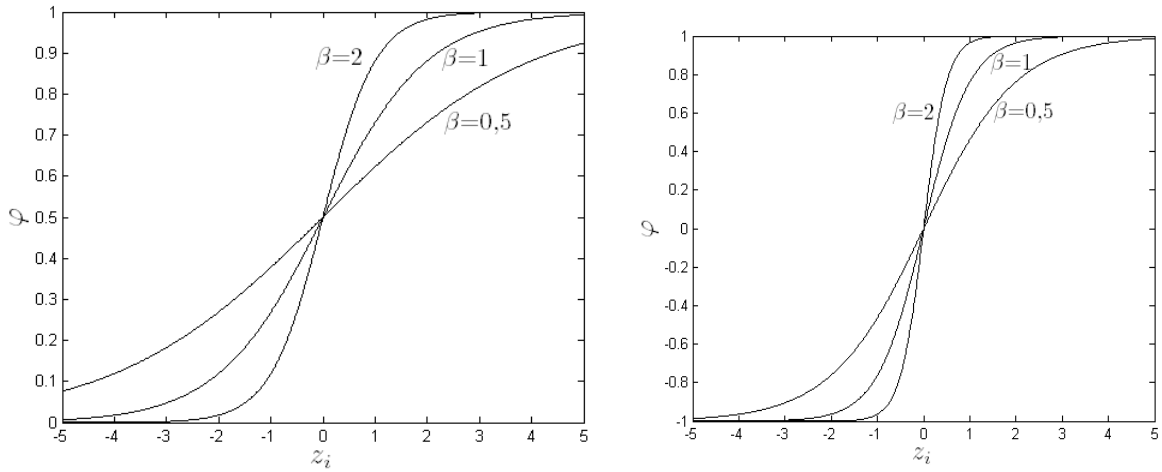
lub funkcję bipolarną

$$\varphi(z_i) = \operatorname{tgh}(\beta z_i) \tag{3.4}$$

Wartość współczynnika β jest parametrem, który wpływa na kształt funkcji aktywacji. Na rys. 3.2 przedstawiono wpływ tego parametru na kształt unipolarnej i bipolarnej funkcji aktywacji.

W praktyce zwykle przyjmuje się β=1.

a) b)



Rys. 3.2. Sigmoidalne funkcje aktywacji: a) unipolarna, b) bipolarna

3.2. Zastosowania sieci neuronowych

Wzajemnie ze sobą połączone neurony tworzą sieć neuronową. Mają one zdolność uczenia się na przykładach i adaptacji do zmieniających się warunków, posiadają zdolność uogólniania nabytej wiedzy. Obecnie tematyka sztucznych sieci neuronowych stanowi intensywnie rozwijającą się interdyscyplinarną dziedzinę wiedzy. Sieci neuronowe znajdują zastosowanie w wielu dziedzinach, nie tylko w technice, lecz także w fizyce, medycynie, statystyce lub nawet naukach ekonomicznych.

Najbardziej rozpowszechnionym zastosowaniem sieci neuronowych jest aproksymacja nieliniowych funkcji wielu zmiennych. Udowodniono, że przy wystarczająco dużej liczbie neuronów sieć neuronowa jest uniwersalnym aproksymatorem nieliniowych funkcji wielu zmiennych [11]. Bardzo wiele zadań modelowania, identyfikacji oraz przetwarzania sygnałów można sprowadzić do zadania aproksymacji [29].

W praktyce wykorzystuje się zarówno modele statyczne i dynamiczne procesów. W porównaniu z innymi klasami modeli zaletą sieci neuronowych jest nie tylko duża dokładność modelowania, lecz także fakt, że ich struktura jest bardzo prosta, regularna. Modele neuronowe mają stosunkowo mało parametrów, nie występuje tzw. zjawisko „przekleństwa wymiarowości", polegające na gwałtownym zwiększaniu liczby parametrów (wag) przy wzroście wymiarowości problemu. Modele neuronowe, co zostanie omówione w dalszej części pracy, mogą być efektywnie zastosowane do modelowania rzeczywistych procesów.

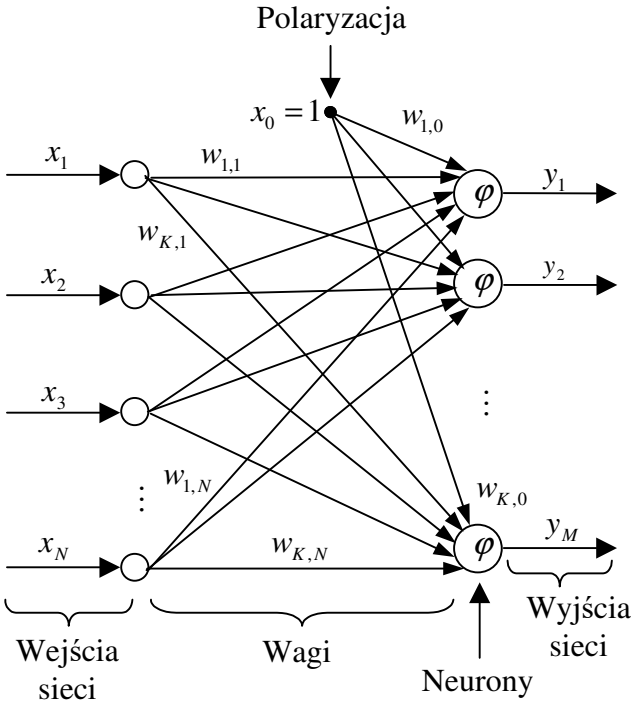
W automatyce neuronowe modele dynamiczne znajdują zastosowanie w wielu algorytmach regulacji (między innymi w algorytmie IMC i w algorytmach regulacji predykcyjnej). Neuronowe modele statyczne mogą być natomiast wykorzystane do optymalizacji ekonomicznej punktu pracy. Oprócz aproksymacji, modelowania i prognozowania, sieci neuronowe znajdują zastosowanie w klasyfikacji oraz rozpoznawaniu, również mowy i obrazów. Sieci neuronowe wykorzystuje się do kompresji danych, przetwarzania sygnałów (np. filtracji, separacji), diagnostyce, optymalizacji.

Wieloletnie badania zaowocowały opracowaniem różnych struktur sieci neuronowych. Rozwojowi struktur sieci neuronowych towarzyszyło opracowanie licznych algorytmów uczenia oraz syntezy optymalnej architektury, umożliwiających wyznaczenie argumentów modelu oraz liczby neuronów ukrytych. O użyteczności sieci neuronowych świadczą liczne zastosowania praktyczne.

3.3. Jednokierunkowe sigmoidalne sieci neuronowe

W zależności od wzajemnego połączenia neuronów można wyróżnić sieci jednokierunkowe lub rekurencyjne. W tym drugim przypadku występuje sprzężenie zwrotne między niektórymi neuronami, np. między wejściem a wyjściem sieci. Choć opracowano bardzo wiele różnych typów sieci [29], największym zainteresowaniem cieszy się sieć jednokierunkowa z neuronami o sigmoidalnej funkcji aktywacji. Sieć taka nazywana jest często perceptronem wielowarstwowym (ang. Multi Layer Perceptron, w skrócie MLP). Sieci tego typu są najczęściej stosowane w praktyce do rozwiązywania najróżniejszych zadań, w tym oczywiście do aproksymacji, modelowania i predykcji.

W sieciach jednokierunkowych przepływ sygnałów odbywa się tylko w jednym kierunku, od wejścia do wyjścia. Na rys. 3.3 przedstawiono najprostszą sieć neuronową, zawierającą tylko jedną warstwę neuronów. Wszystkie neurony działają niezależnie od siebie. Sieć ma N wejść, M wyjść oraz K neuronów. Każdy neuron ma sygnał polaryzacji (wejście ze stałym sygnałem $x_0 = 1$). Wagi oznaczone są symbolem $w_{i,j}$, przy czym indeks i wskazuje neuron ($i = 1,\ldots,K$), natomiast j indeksuje wejścia sieci ($j = 0,\ldots,N$).



Rys. 3.3. Struktura jednokierunkowej jednowarstwowej sieci neuronowej

Sieć neuronowa realizuje odwzorowanie funkcyjne

$$\begin{aligned} y_1 &= f_1(x_1,\ldots,x_N) \\ &\vdots \\ y_M &= f_M(x_1,\ldots,x_N) \end{aligned} \tag{3.5}$$

Korzystając z topologii sieci przedstawionej na rys. 3.3, jej sygnały wyjściowe można obliczyć z zależności

$$\begin{aligned} y_1 &= \varphi(w_{1,0} + w_{1,1}x_1 \ldots + w_{1,N}x_N) \\ &\vdots \\ y_M &= \varphi(w_{M,0} + w_{M,1}x_1 \ldots + w_{M,N}x_N) \end{aligned} \tag{3.6}$$

co można zapisać w sposób zwarty jako

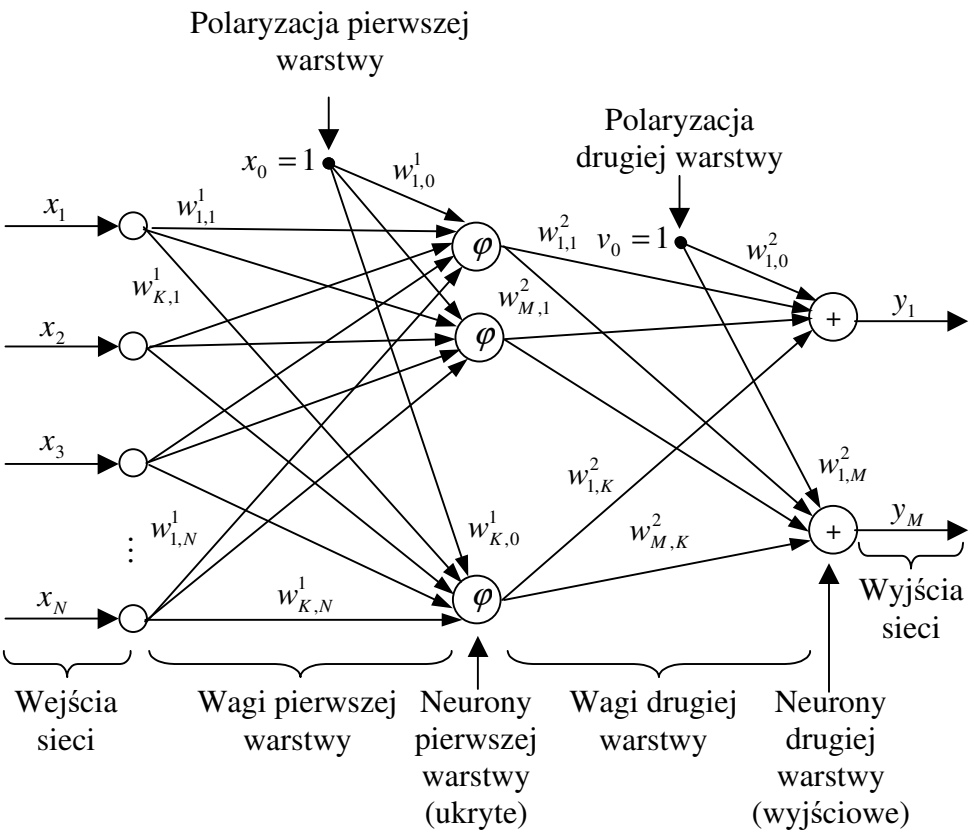
$$y_m = \varphi\left(w_{m,0} + \sum_{j=1}^{N} w_{m,j} x_j \right) \tag{3.7}$$

gdzie m=1,…,M wskazuje wyjście modelu.

Sieć o jednej warstwie neuronów ma ograniczone zdolności aproksymacji. Znacznie lepszym rozwiązaniem jest zastosowanie sieci o dwóch warstwach ukrytych. Analogicznie jak sieć jednowarstwowa, również sieć dwuwarstwowa realizuje odwzorowanie funkcyjne (3.5), ale w nieco inny sposób.

Strukturę sieci dwuwarstwowej przedstawiono na rys. 3.4. Analogicznie jak w przypadku sieci jednowarstwowej, sieć ma N wejść oraz M wyjść. Liczba neuronów pierwszej warstwy (neuronów ukrytych) wynosi K. Liczba neuronów drugiej warstwy (neuronów wyjściowych) jest równa liczbie wyjść sieci M. Do wszystkich neuronów doprowadzone są sygnały polaryzacji. W przypadku warstwy ukrytej jest to po prostu dodatkowe stałe wejście sieci $x_0 = 1$. W przypadku warstwy wyjściowej jest to stały sygnał $v_0 = 1$. Wagi pierwszej warstwy oznaczone są symbolem $w^1_{i,j}$, przy czym indeks i wskazuje neuron ukryty ($i = 1,\ldots,K$), natomiast j indeksuje wejścia sieci ($j = 0,\ldots,N$). Wagi drugiej warstwy oznaczone są jako $w^2_{i,j}$, przy czym indeks i wskazuje wyjście sieci ($i = 1,\ldots,M$), natomiast j indeksuje wejścia drugiej warstwy, czyli sygnał polaryzacji v_0 oraz neurony ukryte ($j = 0,\ldots,K$).

W warstwie ukrytej sieci zastosowano neurony o nieliniowej, na przykład sigmoidalnej funkcji aktywacji, natomiast neurony wyjściowe są liniowe (sumatory). Sieć tego typu jest najczęściej stosowana, ale w ogólności można również zastosować nieliniowe neurony wyjściowe.



Rys. 3.4. Struktura jednokierunkowej dwuwarstwowej sieci neuronowej

Korzystając z topologii sieci pokazanej na rys. 3.4 można wyprowadzić jej równania. Symbolem z_i oznaczono sumę sygnałów wejściowych *i*-tego neuronu ukrytego

$$z_i = \sum_{j=0}^{N} w_{i,j}^1 x_j \tag{3.8}$$

przy czym $i = 1,\ldots,K$. Analogicznie, symbolem v_i oznaczono sygnały wyjściowe kolejnych neuronów warstwy ukrytej

$$v_i = \varphi(z_i) = \varphi\left(\sum_{j=0}^{N} w_{i,j}^1 x_j \right) \tag{3.9}$$

Sygnał na *l*-tym wyjściu sieci jest następujący

$$y_l = \sum_{i=0}^{K} w_{l,i}^2 v_i = \sum_{i=0}^{K} w_{l,i}^2 \varphi\left(\sum_{j=0}^{N} w_{i,j}^1 x_j \right) \tag{3.10}$$

gdzie $l = 1,\ldots,M$.

Sygnały wyjściowe przedstawionej na rys. 3.4 dwuwarstwowej sieci neuronowej z neuronami sigmoidalnymi w warstwie ukrytej i liniowymi neuronami wyjściowymi można zwięźle zapisać za pomocą (3.10). Wyjścia sieci zależą więc od wejść jej wejść, wartości wag obu warstw oraz od funkcji aktywacji φ.

Najczęstszym zastosowaniem sieci neuronowych jest aproksymacja funkcji nieliniowych. Badania eksperymentalne pokazują, że sieci bardzo dobrze sprawdzają się w tej roli. Co więcej, udowodniono teoretycznie, że sieć o dwóch warstwach ukrytych jest doskonałym aproksymatorem funkcji wielu zmiennych [11]. Sieć o dostatecznej liczbie neuronów ukrytych może aproksymować dowolną funkcję ciągłą z zadaną dokładnością. Aproksymacji można dokonać również w inny sposób, na przykład stosując wielomiany. Niestety, do aproksymacji nieliniowych funkcji wielu argumentów, które często występują w różnych dziedzinach techniki, należy zastosować wielomiany wysokich rzędów. W rezultacie, łączna liczba parametrów modelu wielomianowego może być dużo większa od liczby wag sieci neuronowej. Co więcej, aproksymacja neuronowa jest bez porównania bardziej dokładna niż wielomianowa.

3.3.1. Uczenie jednokierunkowych sieci neuronowych

Celem uczenia sieci neuronowej jest dobór wartości wag pierwszej i drugiej warstwy sieci w taki sposób, aby zminimalizować pewną miarę dokładności modelu neuronowego. Dane uczące składają się z wektorów wejściowych sieci $\boldsymbol{x}(s) = \left[x_1(s) \ \ldots \ x_N(s)\right]^T$ oraz odpowiadających im żądanych wektorów wyjściowych sieci $\boldsymbol{d}(s) = \left[d_1(s) \ \ldots \ d_M(s)\right]^T$, przy czym indeks *s* wskazuje numer wektorów danych, $s = 1,\ldots,S$, gdzie *S* jest liczbą wzorców uczących. Minimalizowana w procesie uczenia funkcja celu (funkcja kryterialna) ma zwykle postać

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{s=1}^{S} \sum_{l=1}^{M} (y_l(s) - d_l(s))^2 \tag{3.11}$$

Powyższa funkcja kryterialna jest sumą kwadratów błędów (czyli różnic między aktualnymi wartościami wyjść sieci $y_l(s)$ a wartościami wzorcowymi $d_l(s)$) dla wszystkich próbek

($s = 1,\ldots,S$) i wszystkich wyjść sieci ($l = 1,\ldots,M$). Wektor $\boldsymbol{w}$ grupuje wszystkie parametry modelu – wagi sieci

$$\boldsymbol{w} = \begin{bmatrix} w_{1,0}^1 & \ldots & w_{K,N}^1 & w_{1,0}^2 & \ldots & w_{M,K}^2 \end{bmatrix}^T \tag{3.12}$$

Do uczenia sieci neuronowej stosuje się algorytmy minimalizacji funkcji kryterialnej (3.11). Ponieważ funkcja ta zależy w nieliniowy sposób od wag sieci, zadanie minimalizacji

$$\min_{\boldsymbol{w}} E(\boldsymbol{w}) \tag{3.13}$$

nie może być rozwiązane w prosty sposób analityczny. Do minimalizacji stosuje się nieliniowe, zwykle gradientowe metody optymalizacji. W celu wyznaczenia gradientów funkcji kryterialnej (3.11) względem wag sieci stosuje się algorytm wstecznej propagacji błędów (ang. backpropagation). Jest on często utożsamiany z samą procedurą numerycznej optymalizacji wartości wag [29]. W celu wyprowadzenia gradientów minimalizowanej funkcji kryterialnej (3.11) względem wag sieci warto ją przekształcić wykorzystując zależności (3.9) i (3.10)

$$\begin{aligned} E(\boldsymbol{w}) &= \frac{1}{2}\sum_{s=1}^{S}\sum_{l=1}^{M}\left(\sum_{i=0}^{K} w_{l,i}^2 v_i(s) - d_l(s)\right)^2 \\ &= \frac{1}{2}\sum_{s=1}^{S}\sum_{l=1}^{M}\left(\sum_{i=0}^{K} w_{l,i}^2 \varphi\left(\sum_{j=0}^{N} w_{i,j}^1 x_j(s)\right) - d_l(s)\right)^2 \end{aligned} \tag{3.14}$$

Najłatwiej wyznaczyć pochodne cząstkowe funkcji kryterialnej względem wag drugiej warstwy sieci. Różniczkując funkcję (3.14) otrzymuje się

$$\begin{aligned} \frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}^2} &= \sum_{s=1}^{S}(y_i(s) - d_i(s))\frac{\partial y_i(s)}{\partial w_{i,j}^2} \\ &= \sum_{s=1}^{S}(y_i(s) - d_i(s))v_j(s) \end{aligned} \tag{3.15}$$

przy czym pochodne obliczane są dla wszystkich $i = 1,\ldots,M$, $j = 0,\ldots,K$. Wprowadzając dodatkowe oznaczenia

$$\delta_i^2(s) = y_i(s) - d_i(s) \tag{3.16}$$

szukane pochodne oblicza się ze wzoru

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}^2} = \sum_{s=1}^{S}\delta_i^2(s)v_j(s) \tag{3.17}$$

W przypadku pochodnych cząstkowych względem wag pierwszej warstwy sieci obliczenia są nieco bardziej złożone, ponieważ trzeba postępować zgodnie z zasadami różniczkowania funkcji złożonej. Otrzymuje się

$$\begin{aligned} \frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}^1} &= \sum_{s=1}^{S}\sum_{l=1}^{M}(y_l(s) - d_l(s))\frac{\partial y_l(s)}{\partial v_i(s)}\frac{\partial v_i(s)}{\partial w_{i,j}^1} \\ &= \sum_{s=1}^{S}\sum_{l=1}^{M}(y_l(s) - d_l(s))w_{l,i}^2\frac{\partial \varphi(z_i(s))}{\partial z_i(s)}x_j(s) \end{aligned} \tag{3.18}$$

przy czym pochodne obliczane są dla wszystkich $i = 1,\ldots,K$, $j = 0,\ldots,N$. Wprowadzając dodatkowe oznaczenia

$$\delta_i^1(s) = \sum_{l=1}^{M} (y_l(s) - d_l(s)) w_{l,i}^2 \frac{\partial \varphi(z_i(s))}{\partial z_i(s)} \tag{3.19}$$

szukane pochodne oblicza się ze wzoru

$$\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}^1} = \sum_{s=1}^{S} \delta_i^1(s) x_j(s) \tag{3.20}$$

Warto zauważyć, że pochodne cząstkowe względem wag pierwszej (3.20) i drugiej warstwy (3.17) oblicza się podobnie. Pochodne oblicza się mianowicie jako iloczyn dwóch sygnałów i sumuje po wszystkich dostępnych próbkach. Pierwszy sygnał ($x_j(s)$ lub $v_j(s)$) odpowiada początkowemu węzłowi danej wagi, natomiast drugi sygnał odpowiada błędowi przeniesionemu do węzła, z którym dana waga jest połączona.

Pochodne funkcji aktywacji względem sygnału $z_i(s)$ oblicza się bardzo prosto, w sposób analityczny. Dla funkcji unipolarnej (3.3) obowiązuje

$$\frac{\partial \varphi(z_i(s))}{\partial z_i(s)} = \beta z_i(s)(1 - z_i(s)) \tag{3.21}$$

natomiast dla funkcji bipolarnej (3.4) stosuje się

$$\frac{\partial \varphi(z_i(s))}{\partial z_i(s)} = \beta(1 - (z_i(s))^2) \tag{3.22}$$

Aktualizacja wag sieci odbywa się w wyniku minimalizacji nieliniowej funkcji kryterialnej (3.11). Jest to algorytm iteracyjny. Dla *n*-tej iteracji algorytmu aktualizacja parametrów sieci odbywa się zgodnie z ogólnym wzorem

$$\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \eta_n \boldsymbol{p}_n(\boldsymbol{w}_n) \tag{3.23}$$

przy czym wektory $\boldsymbol{w}_{n+1}$ oraz $\boldsymbol{w}_n$ o strukturze (3.12) zawierają wszystkie wagi sieci w dwu kolejnych iteracjach algorytmu optymalizacji, $\boldsymbol{p}_n(\boldsymbol{w}_n)$ jest kierunkiem optymalizacji, natomiast skalar η_n jest krokiem w danym kierunku (w terminologii sieci neuronowych krok zwany jest również współczynnikiem uczenia). W najprostszej wersji kierunek optymalizacji jest przeciwny do gradientu (algorytm najszybszego spadku)

$$\boldsymbol{p}(\boldsymbol{w}_n) = -\frac{\mathrm{d}E(\boldsymbol{w}_n)}{\mathrm{d}\boldsymbol{w}_n} \tag{3.24}$$

Struktura wektora pochodnych odpowiada oczywiście strukturze wektora wag, a mianowicie

$$\boldsymbol{g}(\boldsymbol{w}) = \frac{\mathrm{d}E(\boldsymbol{w})}{\mathrm{d}\boldsymbol{w}} = \left[\frac{\partial E(\boldsymbol{w})}{\partial w_{1,0}^1} \quad \cdots \quad \frac{\partial E(\boldsymbol{w})}{\partial w_{K,N}^1} \quad \frac{\partial E(\boldsymbol{w})}{\partial w_{1,0}^2} \quad \cdots \quad \frac{\partial E(\boldsymbol{w})}{\partial w_{M,K}^2} \right]^T \tag{3.25}$$

Struktura klasycznego algorytmu uczenia sieci neuronowej jest następująca:

1. Nadanie wagom sieci wartości początkowych (np. losowych), inicjalizacja licznika iteracji (n=0).

2. Obliczenie pochodnych cząstkowych minimalizowanej funkcji kryterialnej względem wag pierwszej (3.20) i drugiej warstwy (3.17).
3. Obliczenie kierunku optymalizacji ze wzoru (3.24).
4. Wyznaczenie współczynnika kroku η_n.
5. Aktualizacja parametrów sieci zgodnie ze wzorem (3.23).
6. Wyznaczenie aktualnej wartości funkcji celu (3.11).
7. Jeżeli spełnione jest kryterium stopu – koniec obliczeń, w przeciwnym przypadku zwiększenie numeru iteracji (n=n+1), przejście do kroku 2.

Aktualizacja wag odbywa się zwykle po prezentacji wszystkich S próbek ze zbioru uczącego. Minimalizowana funkcja kryterialna uwzględnia błędy sieci wszystkich próbek, oblicza się więc gradienty tej funkcji dla wszystkich próbek. Podejście takie jest najczęściej stosowane, gdy sieć neuronowa jest uczona w trakcie projektowania (ang. off-line), np. syntezy układu regulacji. Alternatywnie, możliwa jest również aktualizacja wag po prezentacji tylko jednej próbki. Podejście takie stosuje się wówczas, gdy sieć jest uczona (lub douczana) na bieżąco (ang. on-line).

Algorytm uczenia jest przerywany gdy przekroczona jest maksymalna liczba iteracji. Zwykle stosuje się również dodatkowe kryteria stopu, a mianowicie proces uczenia zostaje przerwany gdy funkcja kryterialna osiągnie pewną, dopuszczalnie małą wartość

$$E(\boldsymbol{w}_n) \le E_{\text{dop}} \tag{3.26}$$

Inny rozwiązaniem jest przerwanie obliczeń wówczas, gdy norma gradientu jest niewielka, to znaczy gdy

$$\left\|\boldsymbol{g}(\boldsymbol{w}_n)\right\|^2 \le \varepsilon \tag{3.27}$$

przy czym $\left\|\boldsymbol{g}(\boldsymbol{w}_n)\right\|^2 = \left(\boldsymbol{g}(\boldsymbol{w}_n)\right)^T \boldsymbol{g}(\boldsymbol{w}_n)$, natomiast wartość ε jest dobierana eksperymentalnie, np. $\varepsilon = 10^{-6}$. Możliwe jest również przerwanie obliczeń gdy różnica wartości funkcji kryterialnej w kolejnych iteracjach jest mała

$$E(\boldsymbol{w}_n) - E(\boldsymbol{w}_{n+1}) \le \varepsilon \tag{3.28}$$

ale algorytm najszybszego spadku jest zwykle bardzo wolno zbieżny i zastosowanie powyższego kryterium mogłoby zbyt wcześnie przerwać obliczenia. Kryterium bazujące na normie gradientu jest pod tym względem lepsze. Im większe składowe wektora gradientu, tym dalej od aktualnego punktu położone jest minimum lokalne optymalizowanej funkcji.

Komentarza wymaga sposób doboru współczynnika kroku η_n. W najprostszej implementacji algorytmu uczącego można założyć, że jest on stały. Podejście takie ma niestety same wady. Po pierwsze, nie wykorzystuje w żaden sposób informacji o minimalizowanej funkcji i jej pochodnych. Po drugie, nie wiadomo, jaką wartość należy przyjąć. Mała wartość η_n prowadzi do bardzo wolnej zbieżności algorytmu uczącego (wiele iteracji), natomiast zbyt duża wartość grozi wyznaczeniem punktu, dla którego wartość funkcji kryterialnej jest większa niż w poprzedniej iteracji, co grozi rozbieżnością algorytmu. Okazuje się, że dla niektórych iteracji algorytmu korzystne jest przyjęcie małych wartości współczynnika η_n, dla niektórych iteracji najlepsze jest przyjęcie dużych wartości.

Sposób doboru współczynnika kroku ma ogromny wpływ na szybkość zbieżności całego algorytmu uczenia sieci neuronowej. W literaturze można znaleźć wiele różnych reguł empirycznego lub adaptacyjnego doboru kroku [29]. Najlepsze rezultaty można jednak

osiągnąć stosując podejście znane w teorii optymalizacji pod nazwą minimalizacji kierunkowej [34]. Przy danym kierunku $\boldsymbol{p}_n(\boldsymbol{w}_n)$ oblicza się mianowicie taką optymalną wartość kroku η_n, dla której wartość funkcji kryterialnej w nowym punkcie $\boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \eta_n \boldsymbol{p}_n(\boldsymbol{w}_n)$ jest najmniejsza. Tak postawione zadanie prowadzi do następującego zadania optymalizacji

$$\min_{\eta_n} E(\boldsymbol{w}_n + \eta_n \boldsymbol{p}_n(\boldsymbol{w}_n)) \tag{3.29}$$

Jest to jednowymiarowe zadanie optymalizacji. Może być ono rozwiązane bardzo efektywnie, to znaczy dokładnie i przy niewielkim nakładzie obliczeń, przy wykorzystaniu metody złotego podziału [34]. Inną metodą, nieco bardziej złożoną, jest algorytm z aproksymacja kwadratową lub sześcienną minimalizowanej funkcji jednej zmiennej $E(\boldsymbol{w}_n + \eta_n \boldsymbol{p}_n(\boldsymbol{w}_n))$.

Z uwagi na bardzo wolną zbieżność klasycznej metody uczenia sieci neuronowych wykorzystującej do obliczania kierunku metodę najszybszego spadku, warto zastosować znacznie bardziej efektywne algorytmy optymalizacji, a mianowicie metody gradientów sprzężonych, zmiennej metryki oraz Levenberga-Marquardta.

W metodzie gradientów sprzężonych kierunek optymalizacji w aktualnej iteracji $\boldsymbol{p}_n$ jest wyznaczany w taki sposób, aby był ortogonalny i sprzężony do wszystkich poprzednich kierunków w poprzednich iteracjach [34]. Wymagania te spełnia kierunek

$$\boldsymbol{p}_n = -\boldsymbol{g}_n + \beta_{n-1} \boldsymbol{p}_{n-1} \tag{3.30}$$

gdzie dla zwięzłości zapisu $\boldsymbol{p}_n = \boldsymbol{p}(\boldsymbol{w}_n)$, $\boldsymbol{g}_n = \boldsymbol{g}(\boldsymbol{w}_n)$, $\boldsymbol{p}_{n-1} = \boldsymbol{p}(\boldsymbol{w}_{n-1})$. Kierunek minimalizacji w aktualnej iteracji jest funkcją aktualnego gradientu oraz poprzedniego kierunku. Opracowano kilka metod gradientów sprzężonych [34]. Najpopularniejsze z nich to metoda Fletchera-Reevesa, w której współczynnik β_{n-1} oblicza się ze wzoru

$$\beta_{n-1} = \frac{\boldsymbol{g}_n^T \boldsymbol{g}_n}{\boldsymbol{g}_{n-1}^T \boldsymbol{g}_{n-1}} = \frac{\|\boldsymbol{g}_n\|^2}{\|\boldsymbol{g}_{n-1}\|^2} \tag{3.31}$$

oraz metoda Polaka-Ribiery, w której

$$\beta_{n-1} = \frac{(\boldsymbol{g}_n - \boldsymbol{g}_{n-1})^T \boldsymbol{g}_n}{\boldsymbol{g}_{n-1}^T \boldsymbol{g}_{n-1}} \tag{3.32}$$

Metody gradientów sprzężonych są dużo szybsze niż metoda najszybszego spadku, wymagają znacznie mniej iteracji. Warto podkreślić, że nie są one zbyt złożone obliczeniowo, aktualny kierunek wyznacza się na podstawie kierunku w poprzedniej iteracji oraz wektorów gradientu w poprzedniej i aktualnej iteracji. Nie są wykorzystywane żadne operacje macierzowe wymagające dużego nakładu obliczeń, wykorzystywane przez algorytm struktury danych nie zajmują wiele pamięci. Dlatego też algorytmy gradientów sprzężonych stosuje się do uczenia dużych sieci, liczących tysiące wag.

Ponieważ w trakcie obliczeń kumulują się błędy zaokrągleń, kolejne wektory kierunku nie są ortogonalne. Dlatego też w rozwiązaniach praktycznych co ustaloną liczbę iteracji (np. równą liczbie zmiennych decyzyjnych algorytmu – liczbie wag sieci) należy odnowić kierunek, to znaczy przyjąć kierunek przeciwny do wektora gradientu (jak w metodzie najszybszego spadku).

Innymi algorytmami optymalizacji, dzięki którym możliwe jest szybkie uczenie sieci

neuronowych, są algorytmy zmiennej metryki [34]. W metodach tych wykorzystuje się kwadratowe przybliżenie minimalizowanej funkcji $E(\boldsymbol{w})$. Rozwinięcie w szereg Taylora funkcji kryterialnej w otoczeniu punktu $\boldsymbol{w}_n$ w kierunku $\boldsymbol{p}_n$ jest następujące

$$E(\boldsymbol{w}_n + \boldsymbol{p}_n) = E(\boldsymbol{w}_n) + \boldsymbol{g}_n^T \boldsymbol{p}_n + \frac{1}{2} \boldsymbol{p}_n^T \boldsymbol{H}_n \boldsymbol{p}_n + \ldots \tag{3.33}$$

gdzie symetryczna macierz drugich pochodnych, zwana hesjanem, ma postać odpowiadającą wektorowi wag (3.12) oraz wektorowi gradientu (3.25)

$$\boldsymbol{H}_n = \begin{bmatrix} \dfrac{\partial^2 E(\boldsymbol{w})}{\partial w_{1,0}^1 \partial w_{1,0}^1} & \cdots & \dfrac{\partial^2 E(\boldsymbol{w})}{\partial w_{1,0}^1 \partial w_{M,K}^2} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial^2 E(\boldsymbol{w})}{\partial w_{M,K}^2 \partial w_{1,0}^1} & \cdots & \dfrac{\partial^2 E(\boldsymbol{w})}{\partial w_{M,K}^2 \partial w_{M,K}^2} \end{bmatrix} \tag{3.34}$$

W punkcie optymalnym funkcji $E(\boldsymbol{w}_n + \boldsymbol{p}_n)$ musi zachodzić

$$\frac{\mathrm{d}E(\boldsymbol{w}_n + \boldsymbol{p}_n)}{\mathrm{d}\boldsymbol{p}_n} = 0 \tag{3.35}$$

co pozwala jednoznacznie określić aktualny kierunek na podstawie gradientu i odwrotności hesjanu

$$\boldsymbol{p}_n = -(\boldsymbol{H}_n)^{-1} \boldsymbol{g}_n \tag{3.36}$$

Warto podkreślić, że wyznaczony kierunek zapewnia minimum funkcji celu w aktualnej iteracji. Powyższy wzór stanowi podstawę tzw. Newtonowskich metod optymalizacji.

W praktyce macierz drugich pochodnych optymalizowanej funkcji kryterialnej nie jest wyznaczana analitycznie w sposób dokładny, a jedynie aproksymowana. Co więcej, aproksymowana jest od razu cała macierz odwrotna, bez potrzeby wykonywania kosztownej obliczeniowo operacji odwracania macierzy. Istnieje kilka metod zmiennej metryki. W metodzie BFGS (od nazwisk autorów: Broyden, Fletcher, Goldfarb, Shanno) aproksymacja odwrotności hesjanu $\boldsymbol{V}_n \approx (\boldsymbol{H}_n)^{-1}$ w aktualnej iteracji obliczana jest ze wzoru rekurencyjnego

$$\boldsymbol{V}_n = \boldsymbol{V}_{n-1} + \left[1 + \frac{\boldsymbol{r}_n^T \boldsymbol{V}_{n-1} \boldsymbol{r}_n}{\boldsymbol{s}_n^T \boldsymbol{r}_n} \right] \frac{\boldsymbol{s}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{r}_n} - \frac{\boldsymbol{s}_n \boldsymbol{r}_n^T \boldsymbol{V}_{n-1} + \boldsymbol{V}_{n-1} \boldsymbol{r}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{r}_n} \tag{3.37}$$

gdzie $\boldsymbol{s}_n = \boldsymbol{w}_n - \boldsymbol{w}_{n-1}$ oznacza wektor zmiany wartości wag, $\boldsymbol{r}_n = \boldsymbol{g}_n - \boldsymbol{g}_{n-1}$ jest wektorem zmiany gradientów. W metodzie DFP (od nazwisk autorów: Davidon, Fletcher, Powell) stosuje się aproksymację

$$\boldsymbol{V}_n = \boldsymbol{V}_{n-1} + \frac{\boldsymbol{s}_n \boldsymbol{s}_n^T}{\boldsymbol{s}_n^T \boldsymbol{r}_n} - \frac{\boldsymbol{V}_{n-1} \boldsymbol{r}_n \boldsymbol{r}_n^T \boldsymbol{V}_{n-1}}{\boldsymbol{r}_n^T \boldsymbol{V}_{n-1} \boldsymbol{r}_n} \tag{3.38}$$

Metody zmiennej metryki są zwane metodami quasi-Newtonowskimi, ponieważ wykorzystuje się kierunek (3.36), ale macierz hesjanu nie jest obliczana dokładnie.

Aproksymację odwrotności hesjanu inicjalizuje się macierzą jednostkową ($\boldsymbol{V}_0 = \boldsymbol{I}$), co oznacza, że pierwsza iteracja jest realizowana zgodnie z algorytmem najszybszego spadku.

Analogicznie jak w przypadku algorytmów gradientów sprzężonych, konieczna jest odnowa kierunku.

Algorytmy zmiennej metryki wykazują znacznie lepszą (kwadratową) zbieżność w porównaniu z algorytmami gradientów sprzężonych. Z drugiej jednak strony, należy pamiętać o tym, że są znacznie bardziej złożone obliczeniowo i wymagają znacznie większej ilości pamięci.

Inną, bardzo dobrą metodą nieliniowej optymalizacji stosowaną do uczenia sieci neuronowych jest metoda Levenberga-Marquardta [29]. W metodzie tej również korzysta się z ogólnego wzoru (3.36) określającego kierunek optymalizacji w metodach Newtonowskich, ale zamiast określać dokładnie macierz drugich pochodnych, stosuje się jej aproksymację $\boldsymbol{G}_n \approx \boldsymbol{H}_n$. Jest ona obliczana na podstawie informacji zawartej w wektorze gradientu. Dla uproszczenia zapisu przyjmuje się, że dostępny jest tylko jeden wzorzec uczący. Gradient jest aproksymowany w następujący sposób

$$\boldsymbol{G}_n = \boldsymbol{J}_n^T \boldsymbol{J}_n + \mu_n \boldsymbol{I} \tag{3.39}$$

gdzie $\boldsymbol{J}$ jest jakobianem, czyli macierzą pierwszych pochodnych błędów sieci względem wszystkich wag. Macierz ta jest obliczana znacznie prościej niż macierz drugich pochodnych, której odwrotność musi być aproksymowana w algorytmie zmiennej metryki. Współczynnik μ_n jest czynnikiem regularyzującym, zwanym też parametrem Levenberga-Marquardta. Jego wartość jest zmieniana podczas uczenia sieci. Na początku, gdy wartość minimalizowanej funkcji kryterialnej jest duża, przyjmuje się duże wartości czynnika regularyzującego. W takiej sytuacji

$$\boldsymbol{G}_n \cong \mu_n \boldsymbol{I} \tag{3.40}$$

co prowadzi do metody najszybszego spadku, ponieważ kierunek optymalizacji przyjmuje wartość

$$\boldsymbol{p}_n = -\frac{\boldsymbol{g}_n}{\mu_n} \tag{3.41}$$

W miarę postępów w uczeniu sieci błąd zmniejsza się. Wartość parametru regularyzującego zostaje przez algorytm zmniejsza. W rezultacie, czynnik $\boldsymbol{J}_n^T \boldsymbol{J}_n$ we wzorze (3.39) jest dominujący, kierunek obliczony zgodnie ze wzorem (3.36) jest zbliżony do metody Newtona. O skuteczności omawianej metody optymalizacji decyduje dobór czynnika regularyzującego.

Warto podkreślić, że krótko omówione metody uczenia sieci neuronowych bazują na informacji zawartej w wektorze gradientu, i, ewentualnie, macierzy hesjanu lub macierzy jakobianu. Są to metody optymalizacji lokalnej, to znaczy poszukiwane jest minimum lokalne, algorytm zostaje zatrzymany, gdy norma wektora gradientu w aktualnym punkcie jest mała (w minimum lokalnym $\left\|\boldsymbol{g}(\boldsymbol{w}_n)\right\|^2 = 0$ i algorytm kończy pracę, ponieważ kierunek dalszych poszukiwań jest zerowy). W praktyce zwykle stosuje się dość skuteczną metodę multistartu, to znaczy proces uczenia sieci zostaje powtórzony kilka (lub kilkanaście) razy dla różnych, zwykle losowych, wartości wag. Podejście takie zwykle umożliwia otrzymanie sieci, dla której wartość funkcji błędu jest dostatecznie mała, a tym samym sieć może być zastosowana do rozwiązania konkretnego zadania.

Opracowano wiele różnych algorytmów heurystycznych do uczenia sieci neuronowych [29]. Co ciekawe, nie mają one zwykle uzasadnienia teoretycznego, ale działają całkiem nieźle. Przykładem niech będzie algorytm RPRP (ang. Resilient backPROPagation). Przy

zmianie wartości wag w kolejnych iteracjach algorytmu uczącego uwzględnia się jedynie znak gradientu, jego konkretna wartość nie ma znaczenia. Modyfikacja wag dowolnej warstwy przebiega zgodnie ze schematem

$$w_{i,j,n+1} = w_{i,j,n} - \eta_{i,j,n} \operatorname{sgn}\left(\frac{\partial E(\boldsymbol{w})}{\partial w_{i,j}} \right) \tag{3.42}$$

Współczynnik długości kroku $\eta_{i,j,n}$ jest dobierany oddzielnie dla każdej wagi $w_{i,j}$ na podstawie zmian gradientu w następujący sposób

$$\eta_{i,j,n} = \begin{cases} \min(a\eta_{i,j,n-1}, \eta_{\max}) & \text{gdy } S_{i,j,n} S_{i,j,n-1} > 0 \\ \max(b\eta_{i,j,n-1}, \eta_{\min}) & \text{gdy } S_{i,j,n} S_{i,j,n-1} < 0 \\ \eta_{i,j,n-1} & \text{w pozostaych przypadkach} \end{cases} \tag{3.43}$$

gdzie $S_{i,j,n} = \dfrac{\partial E(\boldsymbol{w})}{\partial w_{i,j}}$, natomiast *a* oraz *b* są stałymi.

W porównaniu z klasycznym algorytmem najszybszego spadku, algorytm RPROP umożliwia przyspieszenie uczenia w tych obszarach, w których nachylenie minimalizowanej funkcji kryterialnej jest niewielkie [29].

Warto również wspomnieć, że do uczenia sieci neuronowych są stosowane algorytmy optymalizacji globalnej, których celem jest znalezienie minimum globalnego danej funkcji celu. Przykładem takiego podejścia są algorytmy ewolucyjne omówione w rozdziale piątym.

3.4. Statyczne i dynamiczne modele neuronowe

Wspomniano już, że dwuwarstwowa sieć neuronowa jest doskonałym aproksymatorem funkcji wielu zmiennych (3.5). Bez żadnych modyfikacji sieć może być więc zastosowana do modelowania właściwości statycznych procesów. Warto jednak wspomnieć, że najczęściej stosuje się sieci o wielu wejściach i jednym wyjściu. Choć w niektórych przypadkach sieci wielowyjściowe okazują się skutecznym rozwiązaniem (szczególnie wówczas, gdy sygnały wyjściowe są skorelowane), generalnie lepszym rozwiązaniem jest oddzielne uczenie *M* sieci o jednym wyjściu. Pojedyncza sieć realizuje odwzorowanie funkcyjne

$$y = f(x_1, \ldots, x_N) \tag{3.44}$$

Sieci neuronowe można również z powodzeniem zastosować do modelowania procesów dynamicznych. Ponieważ uwzględnienie dynamiki może być zrealizowane różnymi sposobami, opracowano wiele różnych struktur modeli neuronowych modeli dynamicznych. W najprostszym modelu dynamicznym skończonej odpowiedzi impulsowej (ang. Finite Impulse Response, FIR) aktualna wartość sygnału wyjściowego, oznaczona przez $\hat{y}(k)$, jest funkcją sygnału wejściowego w poprzednich chwilach próbkowania

$$\hat{y}(k) = f(u(k-\tau), \ldots, u(k-n_{\mathrm{B}})) \tag{3.45}$$

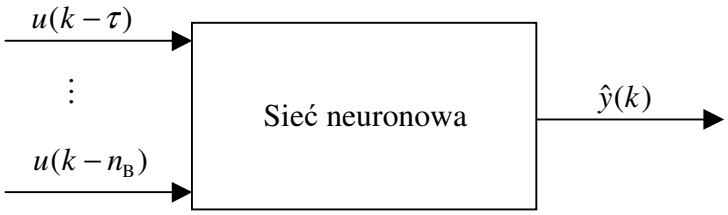
Jeżeli funkcja *f* jest realizowana przez sieć neuronową, przy czym przeważnie stosuje się jednokierunkowe sieci sigmoidalne o jednej warstwie ukrytej, otrzymany model nosi nazwę NNFIR (ang. Neural Network Finite Impulse Response). Jego ogólna struktura została pokazana na rys. 3.5.

Założenie, że sygnał wyjściowy modelu jest funkcją wyłącznie opóźnionych sygnałów wejściowych powoduje, że dynamika modelu (określona liczba naturalną n_{B}) musi być

zwykle wysokiego rzędu. Niestety, model neuronowy typu FIR nadaje się do modelowania ograniczonej klasy procesów. Znacznie lepszym rozwiązaniem jest przyjęcie, że aktualna wartość sygnału wyjściowego jest funkcją nie tylko sygnału wejściowego, ale również wyjściowego procesu w poprzednich iteracjach (chwilach próbkowania)

$$\hat{y}(k) = f(u(k-\tau),\ldots,u(k-n_{\mathrm{B}}), y(k-1),\ldots, y(k-n_{\mathrm{A}})) \tag{3.46}$$

Otrzymany model nosi nazwę NNARX (ang. Neural Network Auto Regressive with eXternal input), jego struktura została pokazana na rys. 3.6. Omawiany model jest czasami nazywany modelem szeregowo-równoległym.

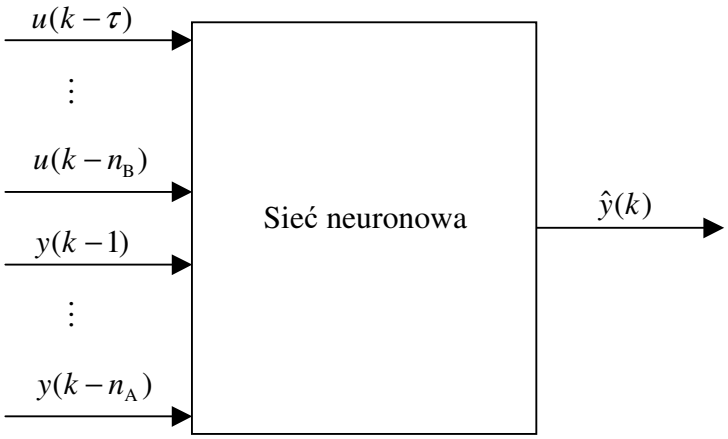


Rys. 3.5. Struktura modelu neuronowego typu FIR (NNFIR)

W bardzo wielu zastosowaniach, na przykład w algorytmach regulacji predykcyjnej, pożądane jest zastosowanie modelu rekurencyjnego typu NNOE (ang. Neural Network Output Error), w którym aktualna wartość sygnału wyjściowego jest funkcją sygnału wejściowego procesu w poprzednich iteracjach oraz sygnału wyjściowego modelu obliczonego w poprzednich iteracjach (a nie sygnału wyjściowego procesu jak ma to miejsce w modelu NNARX)

$$\hat{y}(k) = f(u(k-\tau),\ldots,u(k-n_{\mathrm{B}}), \hat{y}(k-1),\ldots, \hat{y}(k-n_{\mathrm{A}})) \tag{3.47}$$

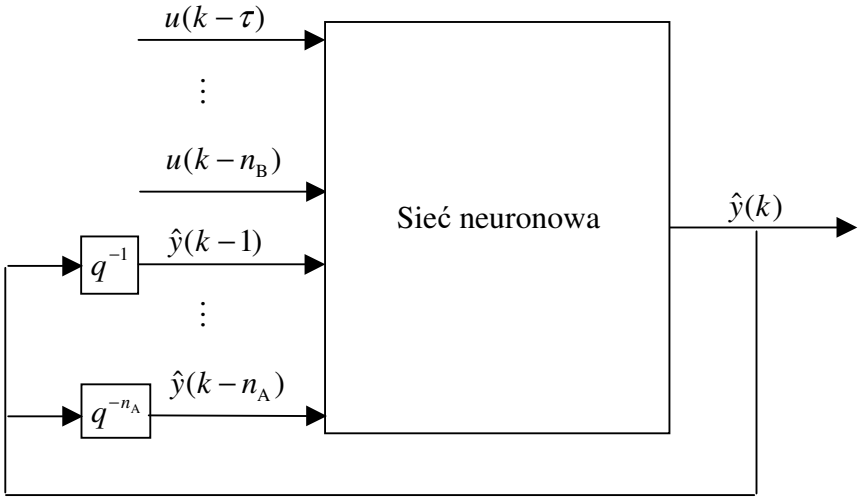
Strukturę modelu NNOE przedstawiono na rys. 3.7. Model ten, w odróżnieniu od szeregowo-równoległego modelu NNARX, jest nazywany modelem równoległym. W literaturze spotyka się również określenie „model symulacyjny", co podkreśla zdolność modelu do obliczania wartości sygnału wyjściowego bez konieczności pomiaru sygnału wyjściowego rzeczywistego procesu.



Rys. 3.6. Struktura modelu neuronowego typu ARX (NNARX)

Należy jednak podkreślić, że o ile uczenie modelu neuronowego typu NNARX jest proste, odbywa się właściwie tak samo jak uczenie modelu statycznego (z tą tylko różnicą, że wejściami modelu są przesunięte w czasie sygnały wejściowe i wyjściowe procesu), to algorytm uczenia modelu typu NNOE jest znacznie bardziej złożony obliczeniowo. Jest to

spowodowane tym, że podczas uczenia musi być uwzględniona rekurencja, zależność sygnału wyjściowego modelu od wartości tego sygnału w poprzednich iteracjach. W praktyce jednak bardzo częste, właściwie dominujące, jest uczenie prostszego modelu typu NNARX, a następnie jego weryfikację na reprezentatywnym zbiorze danych w trybie rekurencyjnym. Choć oczywiście uczenie modelu typu NNOE jest koncepcyjnie bardziej poprawne, stosowany w trybie rekurencyjnym model NNARX (uczony bez rekurencji) zwykle pracuje poprawnie, jego błąd jest dostatecznie mały.



Rys. 3.7. Struktura modelu neuronowego typu OE (NNOE)

Kolejną odmianą modelu neuronowego procesu dynamicznego jest struktura NNARMAX (ang. Neural Network Auto Regressive Moving Average with eXternal input), w którym aktualna wartość sygnału wyjściowego modelu, analogicznie jak w modelu NNARX, jest funkcją sygnału wejściowego i wyjściowego procesu w poprzednich iteracjach oraz, dodatkowo, sygnału błędu modelu w poprzednich iteracjach

$$\hat{y}(k) = f(u(k-\tau),\ldots,u(k-n_B), y(k-1),\ldots, y(k-n_A), \varepsilon(k-1),\ldots,\varepsilon(k-n_C)) \quad (3.48)$$

Błąd modelu obliczany jest jako

$$\varepsilon(k) = y(k) - \hat{y}(k) \quad (3.49)$$

Struktura modelu typu NNARMAX jest pokazana na rys. 3.8.

Warto wspomnieć o strukturach neuronowych służących do modelowania szeregów czasowych. Są to modele typu NNAR (ang. Neural Network Auto Regressive), w których aktualna wartość sygnału wyjściowego jest funkcją wyłącznie sygnału wyjściowego procesu w poprzednich iteracjach (chwilach próbkowania)

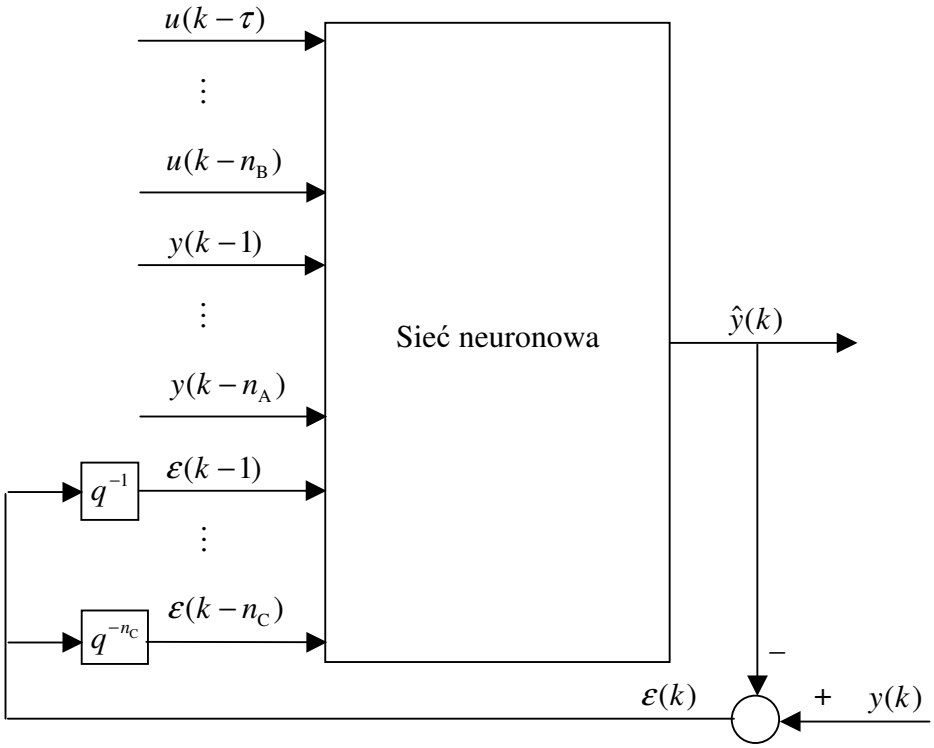
$$\hat{y}(k) = f(y(k-1),\ldots, y(k-n_A)) \quad (3.50)$$

oraz modele NNARMAX (ang. Neural Network Auto Regressive Moving Average), w którym dodatkowo uwzględnia się wpływ sygnału błędu modelu w poprzednich iteracjach

$$\hat{y}(k) = f(y(k-1),\ldots, y(k-n_A), \varepsilon(k-1),\ldots,\varepsilon(k-n_C)) \quad (3.51)$$

Omówione do tej pory struktury neuronowych modeli dynamicznych (tzn. modele NNFIR, NNARX, NNOE, NNARMAX) są modelami typu wejście-wyjście. Sygnał wyjściowy modelu jest funkcją sygnałów wejściowych i wyjściowych (procesu lub modelu) w poprzednich iteracjach. Bardzo szeroka grupa rzeczywistych procesów może być opisana za pomocą modeli typu wejście-wyjście, są one najczęściej stosowane w praktyce w

algorytmach regulacji lub detekcji uszkodzeń. Z drugiej jednak strony warto przypomnieć, że modele tego typu nie są tak ogólne jak modele w przestrzeni stanu. Ograniczenia modeli wejście-wyjście dotyczą procesów z niejednoznaczną nieliniowością, np. histerezą lub luzem oraz częściowo procesów z nieliniowościami nieodwracalnymi. W takich przypadkach decydującą rolę odgrywają niemierzalne stany wewnętrzne. W ogólności, modele typu wejście-wyjście można zastosować, pod warunkiem jednak, że stan jest skończoną funkcją sygnałów wejściowych i wyjściowych w poprzednich iteracjach. Jeżeli nie jest to możliwe, system ma stany ukryte, konieczne jest zastosowanie modeli w przestrzeni stanu.



Rys. 3.8. Struktura modelu neuronowego typu ARMAX (NNARMAX)

W ogólnym przypadku model procesu nieliniowego w przestrzeni stanu ma postać

$$\begin{aligned} x(k+1) &= f(x(k), u(k)) \\ y(k) &= g(x(x)) \end{aligned} \tag{3.52}$$

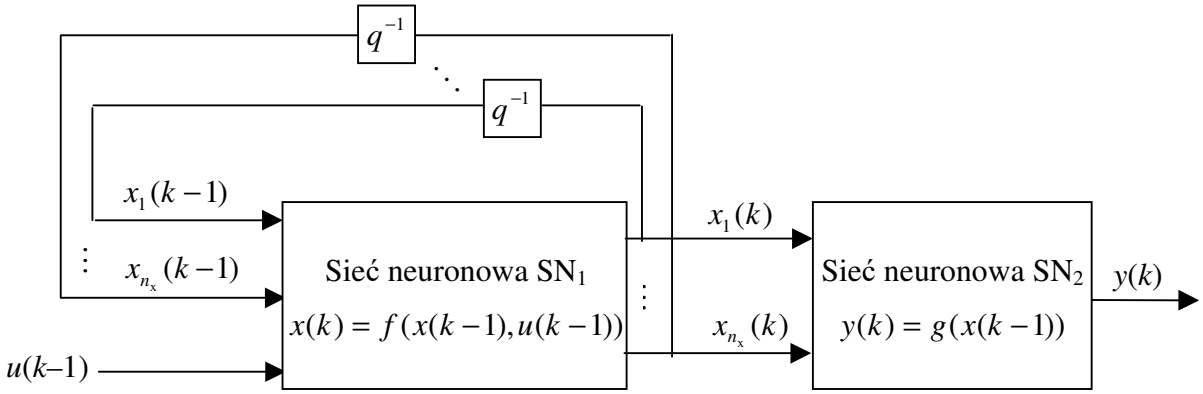
gdzie f oraz g są pewnymi funkcjami nieliniowymi. Wektor stanu ma n_x elementów, tzn. $x(k) = \left[x_1(k) \quad \ldots \quad x_{n_x}(k)\right]^T$. Równoważny model można przedstawić następująco

$$\begin{aligned} x(k) &= f(x(k-1), u(k-1)) \\ y(k) &= g(x(x)) \end{aligned} \tag{3.53}$$

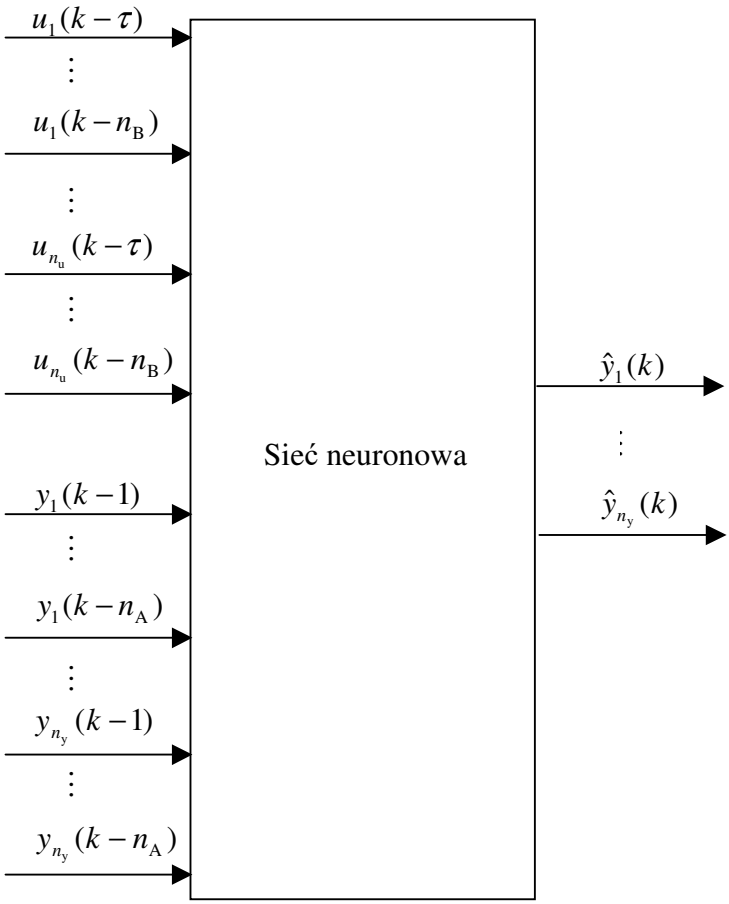
Struktura modelu neuronowego w przestrzeni stanu NNSS (ang. Neural Network State Space) została pokazana na rys. 3.9. W odróżnieniu od omówionych uprzednio modeli typu wejście-wyjście, w modelu typu NNSS stosuje się dwie sieci neuronowe. Pierwsza z nich realizuje nieliniowe równanie stanu, druga – nieliniowe równanie wyjścia.

Przedstawione do tej pory neuronowe modele dynamiczne dotyczą procesów o jednym wejściu. Wiele spotykanych w praktyce procesów ma naturę wielowymiarową, przy czym sprzężenia skrośne są silne, nie mogą być one pominięte podczas modelowania. Niech liczba zmiennych wejściowych wynosi n_u, natomiast liczba zmiennych wyjściowych n_y. Proces ma

wówczas wejścia $u_1, u_2, \ldots, u_{n_u}$ oraz wyjścia $u_1, u_2, \ldots, u_{n_y}$.



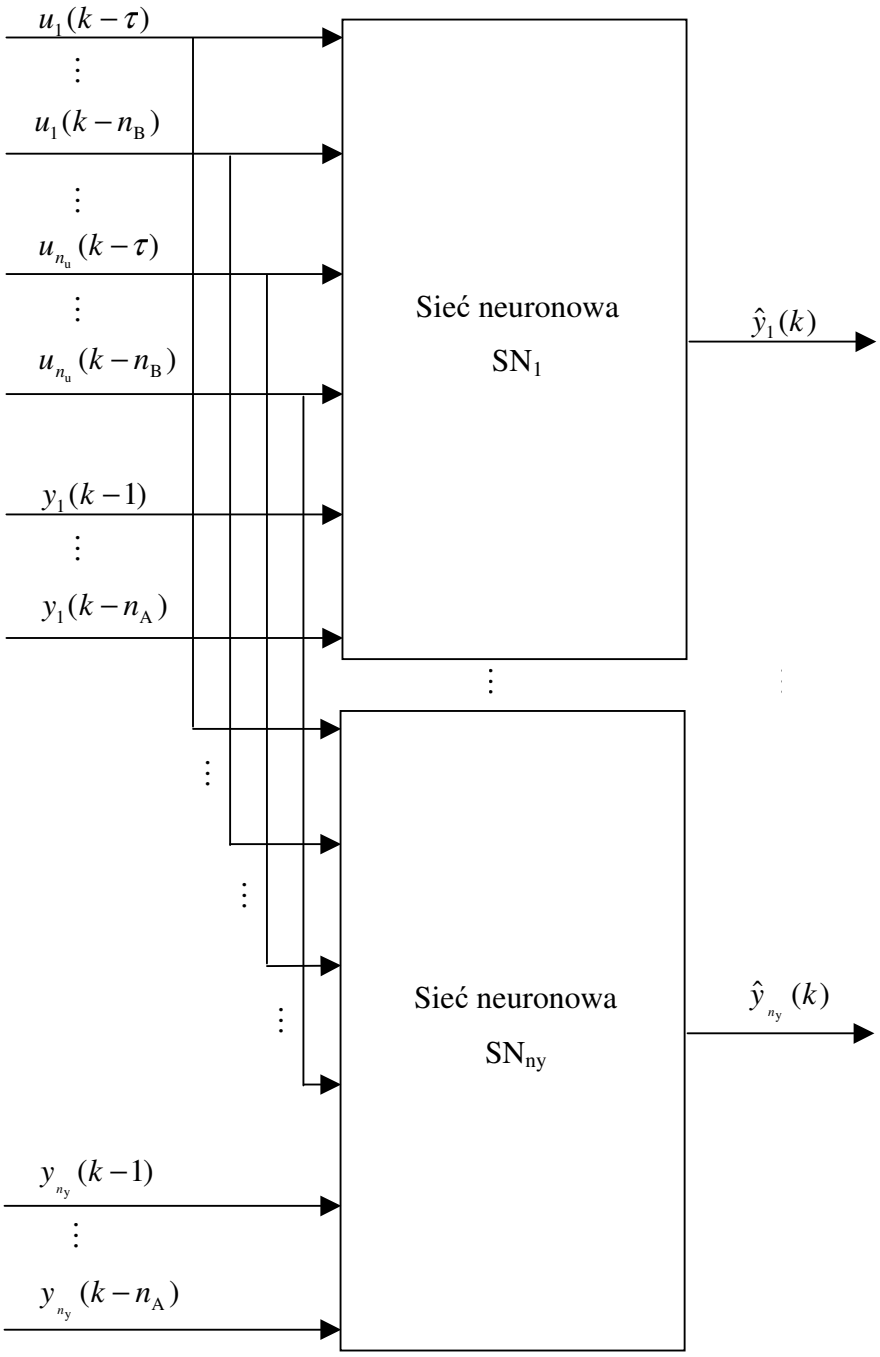
Rys. 3.9. Struktura modelu neuronowego w przestrzeni stanu (NNSS)



Rys. 3.10. Struktura wielowymiarowego modelu neuronowego (NNARX) z pojedynczą siecią

Istnieją dwa podejścia do modelowania procesów wielowymiarowych. W pierwszym przypadku stosuje się jedną sieć neuronową o n_y wyjściach. Struktura wielowymiarowego modelu typu NNARX pokazana jest na rys. 3.10 (struktury modeli NNFIR, NNOE i NNARMAX są analogiczne). Zakładając, że rząd dynamiki wszystkich wejść i wyjść jest taki sam (opóźnienie τ i liczba naturalna n_B jest stała dla wszystkich wejść oraz liczba naturalna n_A jest stałą dla wszystkich wyjść), sieć neuronowa ma aż $n_u(n_B - \tau + 1) + n_y n_A$ wejść i n_y wyjść. Uczenie takiej sieci jest zwykle trudne, konieczne jest użycie bardzo wielu neuronów

ukrytych. W przypadku modelu rekurencyjnego typu NNOE uczenie jest jeszcze bardziej złożone.



Rys. 3.11. Struktura wielowymiarowego modelu neuronowego (NNARX) z n_y sieciami

Znacznie bardziej uniwersalnym podejściem do modelowania procesów wielowymiarowych jest użycie struktury pokazanej na rys. 3.11. Zamiast jednej sieci neuronowej stosuje się n_y sieci. Model ma ogólną postać

$$\begin{aligned}
\hat{y}_1(k) &= f_1(u_1(k-\tau),\ldots,u_1(k-n_B),\ldots,u_{n_u}(k-\tau),\ldots,u_{n_u}(k-n_B),\\
&\qquad y_1(k-1),\ldots,y_1(k-n_A))\\
&\vdots\\
\hat{y}_{n_y}(k) &= f_{n_y}(u_1(k-\tau),\ldots,u_1(k-n_B),\ldots,u_{n_u}(k-\tau),\ldots,u_{n_u}(k-n_B),\\
&\qquad y_{n_y}(k-1),\ldots,y_{n_y}(k-n_A))
\end{aligned} \tag{3.54}$$

Ponieważ funkcje $f_1,\ldots,f_{n_y}$ są realizowane przez oddzielne sieci neuronowe, są one uczone niezależnie. Jest to ogromna zaleta modelu w porównaniu z modelem złożonym tylko z jednej sieci.

Jeżeli chodzi o modele neuronowe w przestrzeni stanu (3.53), również mogą być one zastosowane w przypadku wielowymiarowym. Pokazana na rys. 3.9 struktura modelu ulega niewielkim modyfikacjom (zmienia się liczba wejść i wyjść, liczba zmiennych stanu pozostaje bez zmian). Wejściami pierwszej sieci są wszystkie sygnały wejściowe w poprzedniej iteracji, tzn. $u_1(k-1),\ldots,u_{n_u}(k-1)$ oraz, podobnie jak poprzednio, sygnały stanu $x_1(k-1),\ldots,x_{n_x}(k-1)$. Jeżeli chodzi o neuronową realizację równania wyjścia, można zastosować sieć o n_x wejściach i n_y wyjściach lub n_y oddzielnych sieci o n_x wejściach i jednym wyjściu.

3.5. Dobór architektury sieci

3.5.1. Identyfikacja modelu neuronowego

Identyfikacja modelu, w tym modelu neuronowego składa się z kilku, ściśle ze sobą powiązanych, etapów. Ogólna sieć działań algorytmu identyfikacji została schematycznie przedstawiona na rys. 3.12. Pierwszym etapem jest pozyskanie danych. Zazwyczaj pozyskanie danych odbywa się jednokrotnie, ale można sobie również wyobrazić sytuacje, w których istniejący zbiór danych musi być uzupełniony (np. wówczas, gdy liczba danych jest duża, ale ich zakres jest mniejszy niż zakres pracy modelowanego zjawiska lub urządzenia).

Liczba zmiennych wejściowych modelu statycznego $y = f(x_1,\ldots,x_N)$ odpowiada liczbie argumentów aproksymowanej funkcji (N). W przypadku modeli dynamicznych sprawa jest bardziej złożona, ponieważ liczba wejść modelu zależy od liczby wejść procesu oraz od rzędu dynamiki modelu. Dla procesu o jednym wejściu u i jednym wyjściu y, przyjmując najpopularniejszy model NNARX (3.46) (lub model NNOE (3.47)), w zależności od rzędu dynamiki określonego przez stałe τ, n_A, n_B, można otrzymać model pierwszego rzędu

$$\tau = 1,\ n_A = 1,\ n_B = 1:\quad \hat{y}(k) = f(u(k-1), y(k-1)) \tag{3.55}$$

drugiego rzędu

$$\begin{aligned}
\tau = 1,\ n_A = 2,\ n_B = 2:&\quad \hat{y}(k) = f(u(k-1), u(k-2), y(k-1), y(k-2))\\
\tau = 2,\ n_A = 2,\ n_B = 2:&\quad \hat{y}(k) = f(u(k-2), y(k-1), y(k-2))
\end{aligned} \tag{3.56}$$

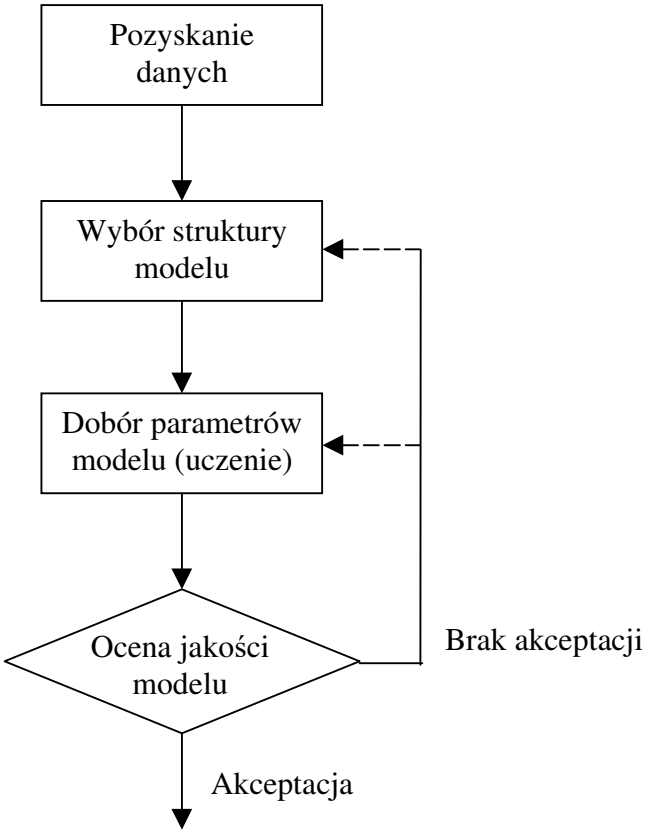
trzeciego rzędu

$$\begin{aligned}
&\tau = 1,\ n_A = 3,\ n_B = 3:\quad \hat{y}(k) = f(u(k-1), u(k-2), u(k-3), y(k-1), y(k-2), y(k-3))\\
&\tau = 2,\ n_A = 3,\ n_B = 3:\quad \hat{y}(k) = f(u(k-2), u(k-3), y(k-1), y(k-2), y(k-3))\\
&\tau = 3,\ n_A = 3,\ n_B = 3:\quad \hat{y}(k) = f(u(k-3), y(k-1), y(k-2), y(k-3))
\end{aligned} \tag{3.57}$$

i wyższych rzędów. W niektórych zastosowaniach może okazać się, że najlepsze rezultaty można osiągnąć przy $n_A \neq n_B$, na przykład stosując model

$$\tau = 3,\ n_A = 1,\ n_B = 3: \quad \hat{y}(k) = f(u(k-3), y(k-1)) \tag{3.58}$$

Liczba wejść modelu neuronowego jest równa $n_A + n_B - \tau + 1$. Dla wszystkich powyższych modeli należy zastosować jeden neuron wyjściowy, ponieważ proces ma jedno wyjście.



Rys. 3.12. Identyfikacja modelu

Przy modelowaniu procesów dynamicznych o n_u wejściach i n_y wyjściach najwygodniej zastosować n_y niezależnych, uczonych niezależnie, sieci neuronowych o jednym wyjściu. Przyjmując, że model każdego wyjścia ma taki sam rząd dynamiki określony przez stałe τ, n_A, n_B (model dany równaniami (3.54)), każda z sieci ma $n_A + n_u(n_B - \tau + 1)$ wejść. Zakładając natomiast, że modele poszczególnych wyjść mogą mieć różną dynamikę, czyli przyjmując ogólny model w postaci

$$\begin{aligned}
\hat{y}_1(k) = f_1(&u_1(k-\tau^{1,1}),\ldots,u_1(k-n_B^{1,1}),\ldots,u_{n_u}(k-\tau^{1,n_u}),\ldots,u_{n_u}(k-n_B^{1,n_u}),\\
&y_1(k-1),\ldots,y_1(k-n_A^1))\\
&\vdots\\
\hat{y}_{n_y}(k) = f_{n_y}(&u_1(k-\tau^{n_y,1}),\ldots,u_1(k-n_B^{n_y,1}),\ldots,u_{n_u}(k-\tau^{n_y,n_u}),\ldots,u_{n_u}(k-n_B^{n_y,n_u}),\\
&y_{n_y}(k-1),\ldots,y_{n_y}(k-n_A^{n_y}))
\end{aligned} \tag{3.59}$$

każda z sieci ($m = 1,\ldots,n_y$) ma $n_A^m + \sum_{n=1}^{n_u}(n_B^{m,n} - \tau^{m,n} + 1)$ węzłów wejściowych. Liczba wejść w

przypadku innych typów modeli dynamicznych (NNFIR, NNAR, NNARMAX) również zależy od liczby wejść procesu i przyjętego rzędu dynamiki. W przypadku modelu neuronowego w przestrzeni stanu (NNSS) określonego równaniem (3.53), liczba węzłów wejściowych pierwszej sieci wynosi $n_x + 1$, liczba wejść drugiej sieci to n_x, pierwsza sieć ma n_x wyjść, druga sieć ma jedno wyjście (n_x oznacza liczbę zmiennych stanu).

Po wyborze wejść i wyjść modelu konieczne jest ustalenie liczby warstw sieci, liczby neuronów ukrytych w każdej warstwie oraz funkcji aktywacji. Ponieważ dwuwarstwowa sieć neuronowa z nieliniową warstwą ukrytą jest doskonałym aproksymatorem funkcji wielu zmiennych [11], najczęściej wykorzystuje się właśnie sieć z jedną nieliniową warstwą ukrytą, przypadki zastosowania sieci trójwarstwowych są bardzo rzadkie. Przy aproksymacji funkcji ciągłych (i identyfikacji modeli dynamicznych procesów dynamicznych) najczęściej stosuje się bipolarne lub unipolarne funkcje aktywacji pokazane na rys. 3.2.

Powracając do ogólnej procedury identyfikacji przedstawionej schematycznie na rys. 3.12, po wyborze struktury odbywa się uczenie modelu. Ponieważ uczenie jest w istocie nieliniową optymalizacją funkcji błędu modelu (problem minimów lokalnych), dla tej samej struktury sieci proces uczenia powtarza się, np. 10 razy. Jeżeli najlepszy z nauczonych modeli nie jest wystarczająco dokładny, można spróbować douczyć modele lub też nauczyć je od początku, startując z innego punktu początkowego (określonego przez wagi). Zazwyczaj jednak należy zmodyfikować strukturę modelu (zmieniając liczbę neuronów ukrytych lub też rząd dynamiki, a tym samym jego wejścia), a następnie nauczyć model o zmienionej strukturze. Uczenie bardzo często rozpoczyna się od sieci o minimalnej liczbie neuronów ($K = 1$). Jeżeli jakość modelu nie spełnia oczekiwań (jest za mało dokładny), przyjmuje się sieć o dwóch neuronach ukrytych, następnie o trzech itd. Po kilku (lub kilkunastu) próbach można znaleźć sieć, która aproksymuje dane z żądaną dokładnością.

Struktura modelu dynamicznego określona jest nie tylko liczbą neuronów ukrytych, ale też rzędem dynamiki. Dlatego też zazwyczaj wstępnie przyjmuje się pewne wejścia sieci (np. $u(k-1),\ y(k-1)$), a następnie uczy kilka modeli o różnej liczbie neuronów ukrytych. Oczywiście dla każdej struktury sieć uczona jest np. 10 razy. Jeżeli zwiększanie liczby neuronów ukrytych nie poprawia znacząco jakości modelu, zwiększa się rząd dynamiki (np. przyjmując wejścia modelu $u(k-1), u(k-2),\ y(k-1),\ y(k-2)$)) i uczy modele o różnej liczbie neuronów ukrytych. Zazwyczaj, po kilku próbach można dobrać taki rząd dynamiki i liczbę neuronów ukrytych, dla których sieć jest wystarczająco dokładna.

3.5.2. Dobór danych i generalizacja sieci

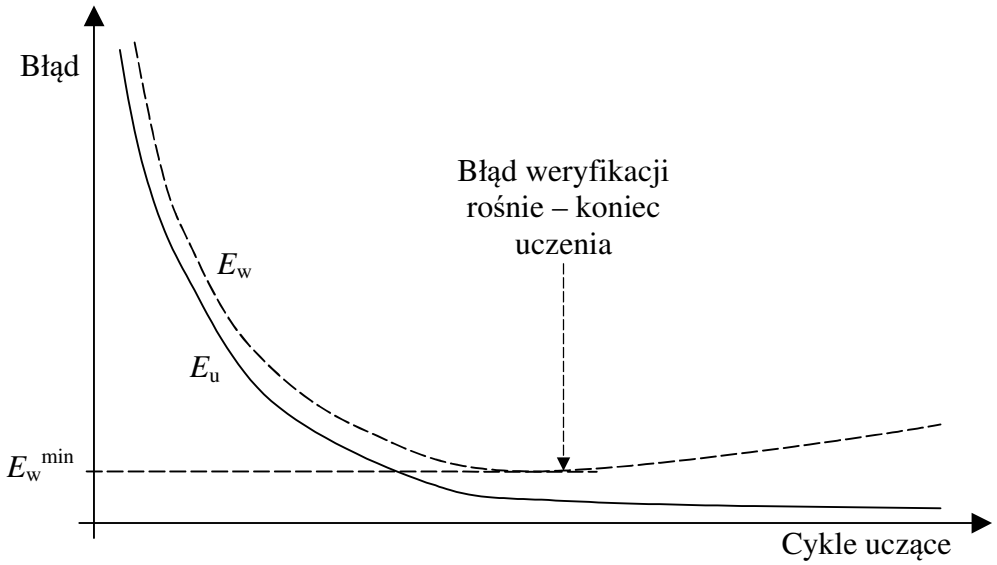
Jakość i liczba danych stosowanych podczas identyfikacji jest sprawą zasadniczą, ponieważ uczenie i weryfikacja modelu neuronowego odbywa się wyłącznie na podstawie danych, żadna inna informacja o modelowanym procesie nie jest brana pod uwagę. Na przykład, podczas modelowania procesu technologicznego zakres zmienności dostępnych danych musi odpowiadać zakresowi pracy urządzenia. Jeżeli dane nie obejmują całego zakresu pracy, otrzymuje się model lokalny.

Każdy model empiryczny, w tym również oczywiście model neuronowy, musi mieć zdolność generalizacji (uogólniania). Sieć uczona jest na podstawie pewnego, w miarę bogatego i licznego, zbioru danych, natomiast podczas pracy modelu, np. w algorytmie regulacji, model powinien poprawnie reagować na nieco inne pobudzenia, ale oczywiście z przyjętego zakresu. Aby model neuronowy posiadał zdolność generalizacji kluczowe znaczenie ma sposób oceny jego jakości (ostatni etap na rys. 3.12). Ocena modelu nie może odbywać się na tym samym zbiorze danych, który służy do uczenia. W najprostszym podejściu wykorzystuje się dwa zbiory danych:

a) zbiór uczący,

b) zbiór weryfikujący.

Pierwszy ze zbiorów służy *wyłącznie* do uczenia (błąd modelu jest minimalizowany właśnie na tym zbiorze), natomiast ocena jakości modelu odbywa się *tylko* dla danych weryfikujących. Model uczony jest tak długo, aż błąd uczenia osiągnie akceptowalną wartość. Po zakończeniu procesu uczenia oblicza się wartość funkcji błędu dla danych weryfikujących. Wybór konkretnego modelu spośród kilku różniących się architekturą lub początkowymi wartościami wag odbywa się *wyłącznie* na podstawie błędu weryfikacji. Niestety, z opisanym podejściem związane jest pewne niebezpieczeństwo, które zostało zilustrowane na rys. 3.13. W początkowym okresie uczenia zarówno błąd uczenia jak i weryfikacji szybko maleją. W drugiej części uczenia błąd uczenia również maleje, ale wolno. Niestety, od pewnej iteracji błąd dla danych weryfikujących rośnie. Sieć dopasowuje się do danych uczących, traci zdolność generalizacji. Zastosowanie sieci „przeuczonej" np. w algorytmie regulacji może prowadzić do złych rezultatów. Uczenie należy przerwać wówczas, gdy błąd weryfikacji zaczyna rosnąc. Podczas uczenia, po każdej aktualizacji wag sieci, należy obliczyć wartość błędu dla danych weryfikujących i zdecydować o kontynuacji algorytmu lub o jego zatrzymaniu.



Rys. 3.13. Błąd uczenia E_u i błąd weryfikacji E_w w kolejnych iteracjach algorytmu uczącego

Aby zapewnić dobrą generalizację modelu należy wykorzystać trzy zbiory danych:

a) zbiór uczący,

b) zbiór weryfikujący,

c) zbiór testowy.

Analogicznie jak poprzednio, pierwszy ze zbiorów służy do uczenia, w trakcie którego obliczany jest na bieżąco błąd weryfikacji. Algorytm uczący jest przerywany gdy wartość błędu weryfikacji zaczyna wzrastać. Wybór modelu odbywa się wyłącznie na podstawie błędu weryfikacji. Na końcu procesu identyfikacji, aby niezależnie sprawdzić wybrany model, oblicza się błąd dla trzeciego zbioru – zbioru testowego. Zbiór ten nie jest stosowany ani podczas uczenia ani podczas weryfikacji. Jeżeli błąd testowania jest akceptowalny uznaje się, że wybrany model ma dobre właściwości generalizacji.

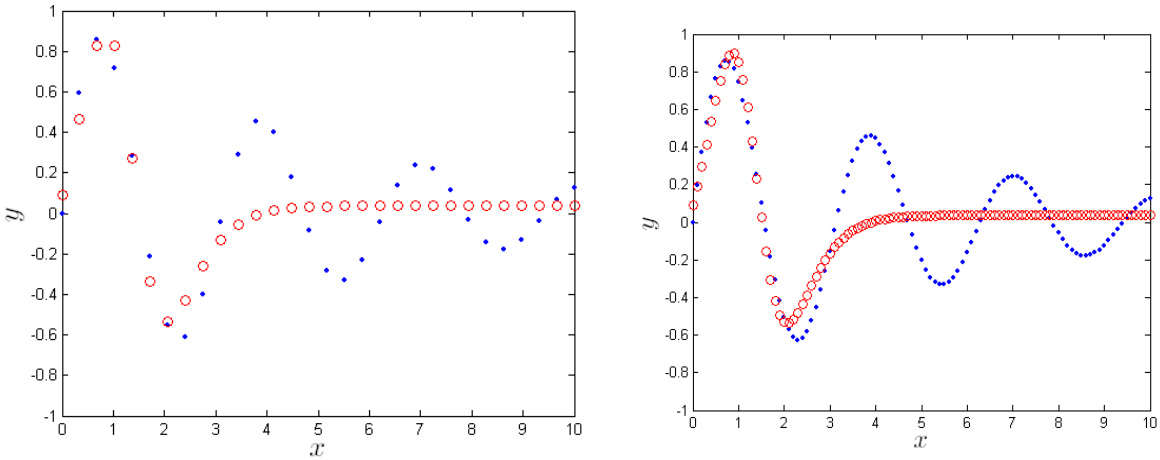
Przykład

Aby zilustrować wpływ liczby neuronów ukrytych na zdolności generalizacji modelu, rozpatruje się uczenie dwuwarstwowej sieci neuronowej aproksymującej funkcję nieliniową

$$y(x) = \frac{\sin(2x)}{\exp(0{,}2x)} \tag{3.60}$$

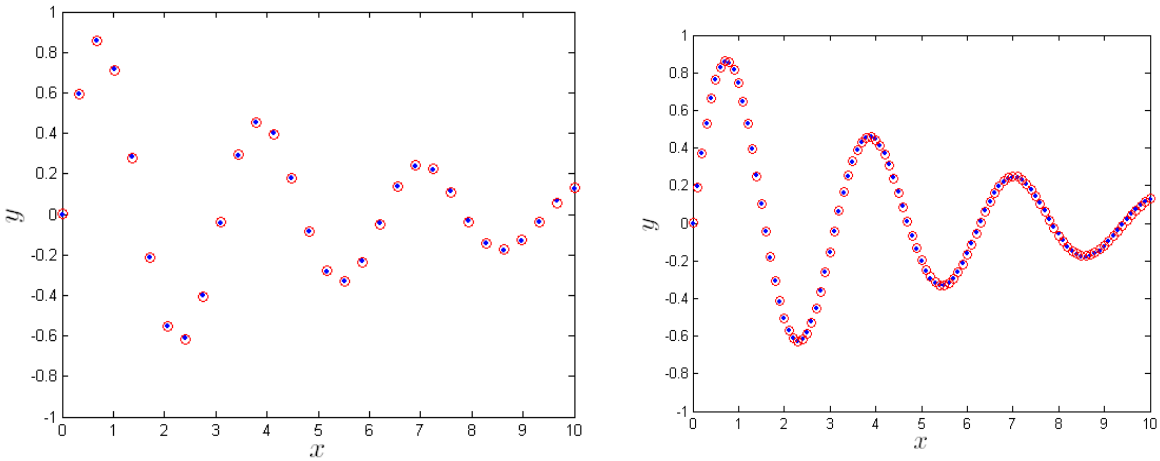
przy czym $0 \le x \le 10$. W warstwie ukrytej sieci zastosowano neurony z funkcją aktywacji typu tangens hiperboliczny. Zbiór uczący liczy 34 próbek, zbiór weryfikujący liczy 101 próbek. Zastosowano najprostszy schemat uczenia, w którym następuje maksymalnie 500 iteracji algorytmu Levenberga-Marquardta, a następnie zostaje obliczony błąd dla danych weryfikujących (błąd weryfikacji nie jest obliczany na bieżąco w trakcie uczenia). Na rys. 3.14 pokazano dane oraz wyjście modelu neuronowego zawierającego K=2 neurony ukryte. Niestety, za pomocą dwóch neuronów ukrytych nie można aproksymować danej funkcji. Po zwiększeniu liczebności warstwy ukrytej do K=5 udaje się otrzymać dokładny model. Sieć działa prawidłowo zarówno dla danych uczących jak i weryfikujących, które nie uczestniczą w uczeniu co pokazano na rys. 3.15. Błąd uczenia oraz błąd testowania (tabela 3.1) są tego samego rzędu. Dla K=5 neuronów ukrytych sieć ma bardzo dobre właściwości generalizacji.

a) b)

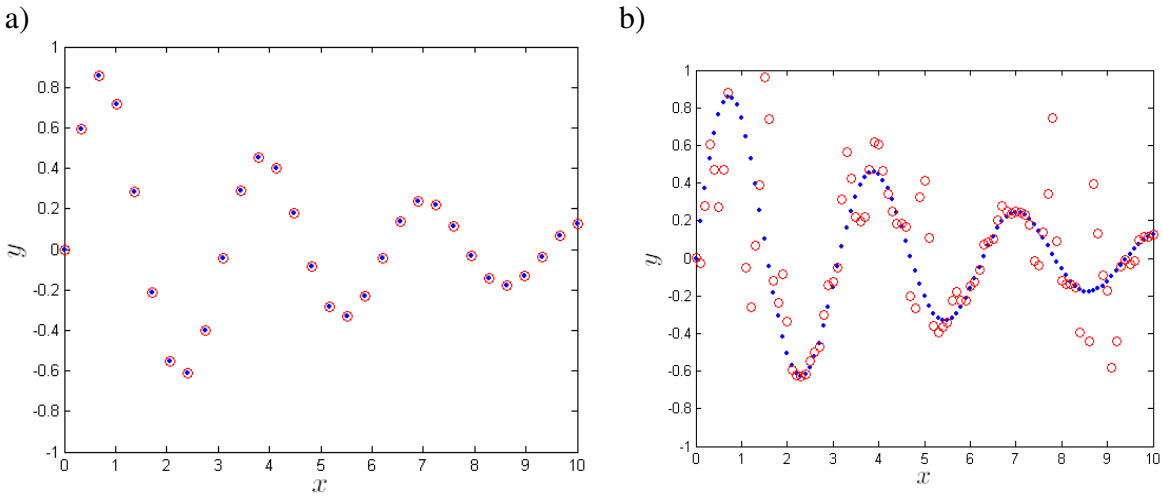


Rys. 3.14. Dane oraz wyjście modelu neuronowego zawierającego K=2 neurony ukryte: a) zbiór danych uczących, b) zbiór danych weryfikujących

a) b)



Rys. 3.15. Dane oraz wyjście modelu neuronowego zawierającego K=5 neuronów ukrytych: a) zbiór danych uczących, b) zbiór danych weryfikujących



Rys. 3.16. Dane oraz wyjście modelu neuronowego zawierającego K=50 neuronów ukrytych: a) zbiór danych uczących, b) zbiór danych weryfikujących

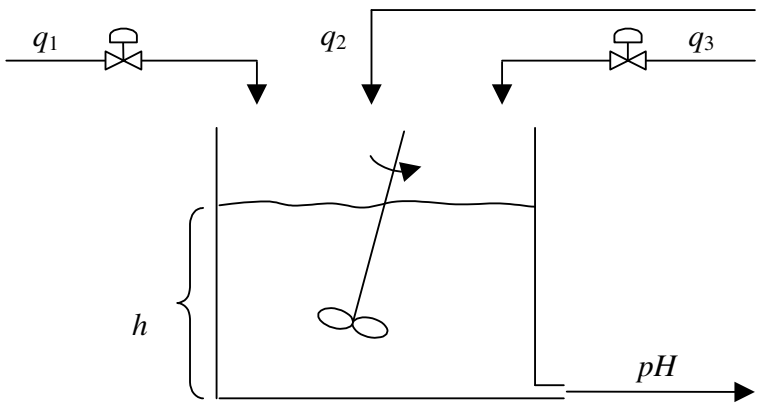
Przygotowano również model z nadmiarową (w stosunku do zbioru uczącego) liczbą neuronów. Na rys. 3.16 przedstawiono dane oraz wyjście modelu neuronowego zawierającego K=50 neuronów ukrytych. Dla danych uczących sieć osiąga błąd porównywalny z błędem maszynowym (tabela 3.1). Niestety, jakość generalizacji jest fatalna. Sieć nie potrafi prawidłowo aproksymować funkcji dla tych próbek, które nie uczestniczyły w uczeniu.

Tabela 3.1. Wpływ liczby neuronów ukrytych na błąd modelu neuronowego

K	Zbiór uczący	Zbiór weryfikujący
2	$1{,}158255 \cdot 10^{0}$	$3{,}957997 \cdot 10^{0}$
5	$6{,}207156 \cdot 10^{-4}$	$2{,}063321 \cdot 10^{-3}$
50	$9{,}122748 \cdot 10^{-12}$	$7{,}209912 \cdot 10^{0}$

Przykład

Rozpatrywanym procesem jest reaktor pH (reaktor neutralizacji) pokazany na rys. 3.17. W zbiorniku odbywa się mieszanie substancji: HNO_3, NaOH oraz $NaHCO_3$. Proces ma dwie sterowane zmienne wejściowe, a mianowicie q_1 – natężenie przepływu HNO_3, q_3 – natężenie przepływu NaOH, zmienną niesterowaną (zakłóceniem) jest q_2 – natężenie przepływu $NaHCO_3$. Proces ma dwa wyjścia: h – wysokość słupa cieczy w reaktorze, oraz pH produktu.



Rys. 3.17. Reaktor pH

Model matematyczny procesu [4] złożony jest z trzech równań różniczkowych zwyczajnych

$$\begin{aligned}
\frac{\mathrm{d}W_{\mathrm{a}4}(t)}{\mathrm{d}t} &= \frac{q_1(t)(W_{\mathrm{a}1}-W_{\mathrm{a}4}(t))+q_2(t)(W_{\mathrm{a}2}-W_{\mathrm{a}4}(t))+q_3(t)(W_{\mathrm{a}3}-W_{\mathrm{a}4}(t))}{Ah(t)} \\
\frac{\mathrm{d}W_{\mathrm{b}4}(t)}{\mathrm{d}t} &= \frac{q_1(t)(W_{\mathrm{b}1}-W_{\mathrm{b}4}(t))+q_2(t)(W_{\mathrm{b}2}-W_{\mathrm{b}4}(t))+q_3(t)(W_{\mathrm{b}3}-W_{\mathrm{b}4}(t))}{Ah} \\
A\frac{\mathrm{d}h(t)}{\mathrm{d}t} &= q_1(t)+q_2(t)+q_3(t)-C_V\sqrt{h(t)}
\end{aligned} \tag{3.61}$$

oraz z nieliniowego równania algebraicznego

$$W_{\mathrm{a}4}(t)+10^{pH(t)-14}+W_{\mathrm{b}4}(t)\frac{1+2\times 10^{pH(t)-pK_2}}{1+10^{pK_1-pH(t)}+10^{pH(t)-pK_2}}-10^{-pH(t)}=0 \tag{3.62}$$

gdzie $pK_1=-\log_{10}K_{\mathrm{a}1}$, $pK_2=-\log_{10}K_{\mathrm{a}2}$. Wartości wszystkich parametrów modelu fizykochemicznego reaktora pH podano w tabeli 3.2.

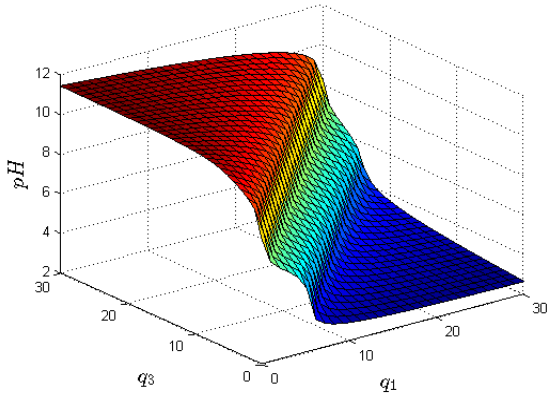
Reaktor pH charakteryzuje się nieliniowymi właściwościami statycznymi i dynamicznymi. W szczególności, przedstawiona na rys. 3.18 charakterystyka statyczna $pH(q_1,q_3)$ jest silnie nieliniowa. Przyjęto następujący zakres sygnałów sterujących

$$0\ \text{ml/s} \le q_1 \le 30\ \text{ml/s}, \quad 0\ \text{ml/s} \le q_3 \le 30\ \text{ml/s} \tag{3.63}$$

oraz założono, że zakłócenie ma stałą wartość $q_2=0{,}55$ ml/s.

Tabela 3.2. Parametry modelu fizykochemicznego reaktora pH

A=207 cm^2	$W_{\mathrm{a}1}$=0,003 M	$W_{\mathrm{b}1}$=0 M
$K_{\mathrm{a}1}$=4,47·10^{-7}	$W_{\mathrm{a}2}$=–0,03 M	$W_{\mathrm{b}2}$=0,03 M
$K_{\mathrm{a}1}$=5,62·10^{-11}	$W_{\mathrm{a}3}$=–0,00305 M	$W_{\mathrm{b}3}$=0,00005 M



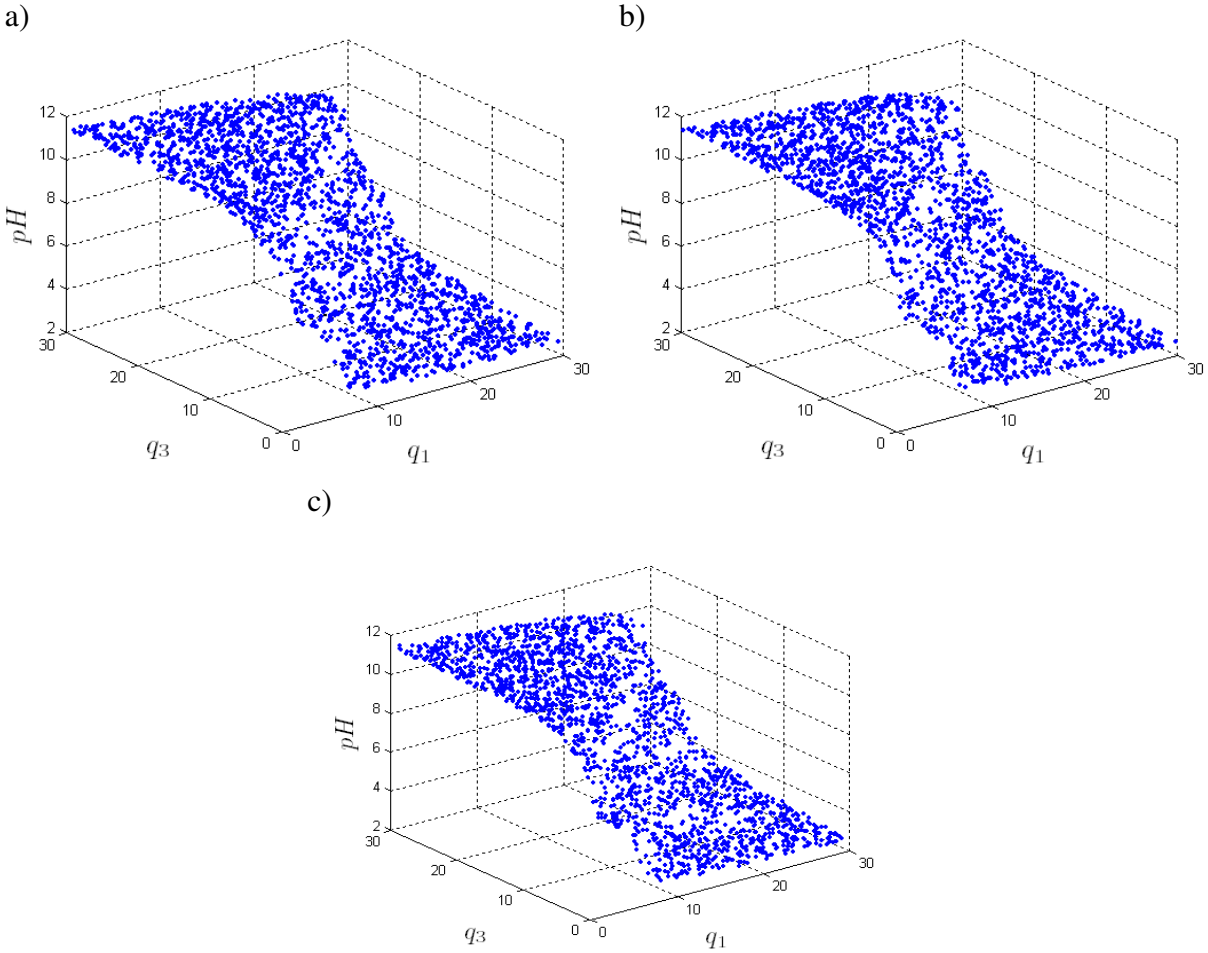
Rys. 3.18. Charakterystyka statyczna reaktora pH

Celem badań jest aproksymacja charakterystyki statycznej $pH(q_1,q_3)$ reaktora pH. Oczywiście, charakterystyka taka jest określona w sposób dokładny przez statyczny model fizykochemiczny (w modelu dynamicznym (3.61), (3.62) należy zaniedbać zmiany czasu, wszystkie pochodne przyrównać do 0). Niestety, otrzymany w taki sposób model zawiera cztery równania algebraiczne, ostatnie z nich jest silnie nieliniowe. Zastosowanie takiego modelu do bieżącej optymalizacji punktu pracy może być złożone obliczeniowo. Dlatego też warto zastosować znacznie prostszy model statyczny, np. typu wielomianowego lub neuronowego.

Fizykochemiczny model statyczny posłużył do wygenerowania w sposób losowy danych uczących, weryfikujących i testowych przedstawionych na rys. 3.19. Wszystkie zbiory liczą

2000 próbek. Dane zostały przeskalowane. Wejściami modelu są sygnały $(q_1 - q_{10})/15$ i $(q_3 - q_{30})/15$, natomiast wyjściem sygnał $(pH - pH_0)/4$, przy czym wielkości $q_{10} = 16{,}6$ ml/s, $q_{30} = 15{,}6$ ml/s, $pH_0 = 7{,}0255$ odpowiadają nominalnemu punktowi pracy.

Do modelowania charakterystyki statycznej $pH(q_1, q_3)$ użyto dwuwarstwową jednokierunkową sieć neuronową (MLP), badano modele liczące od $K = 1$ do $K = 10$ neuronów ukrytych z funkcją aktywacji $\varphi = \mathrm{tgh}$. Do uczenia zastosowano algorytm Levenberga-Marquardta. Dla każdej konfiguracji sieci uczenie powtórzono 10 razy. W tabeli 3.3 pokazano wpływ liczby neuronów ukrytych (*K*) na liczbę parametrów i dokładność (błąd średniokwadratowy) modeli. Mając na uwadze zarówno dokładność (błąd dla zbioru uczącego) oraz złożoność modelu, wybrano model o $K = 6$ neuronach ukrytych, który ma 25 wag. Na rys. 3.20a porównano dane oraz wyjście modelu dla zbioru danych weryfikujących. Aby zademonstrować dużą dokładność modelu, na rys. 3.21 przedstawiono płaszczyznę określoną przez wyjście modelu dla zbioru danych weryfikujących oraz błąd modelu. Płaszczyzna jest gładka, jest optycznie bardzo zbliżona do rzeczywistej charakterystyki przedstawionej na rys. 3.18.



Rys. 3.19. Zbiory danych: a) zbiór uczący, b) zbiór weryfikujący, c) zbiór testowy

Na podstawie dostępnych danych wyznaczono metodą najmniejszych kwadratów modele wielomianowe. W tabeli 3.4 pokazano wpływ rzędu modelu (*n*) na liczbę parametrów i dokładność. W porównaniu z wybranym modelem neuronowym liczącym 6 neuronów ukrytych, który ma 25 parametrów i dla którego błąd dla zbioru weryfikującego wynosi $1{,}549976 \cdot 10^{-1}$, jakość modeli wielomianowych jest zła. Dla modelu czwartego rzędu

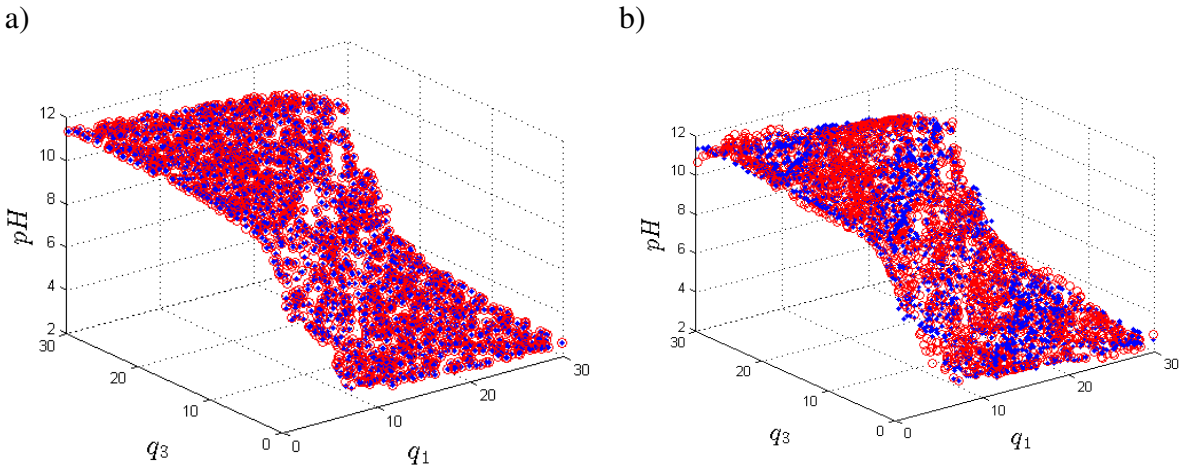
liczącego też 25 parametrów, błąd dla zbioru weryfikującego wynosi $1{,}135175 \cdot 10^{1}$. Dla najlepszego modelu wielomianowego (rząd dziesiąty), liczącego aż 121 parametrów, błąd wynosi $3{,}796229 \cdot 10^{0}$. Na rys. 3.20b porównano dane oraz wyjście modelu wielomianowego czwartego rzędu dla zbioru danych weryfikujących, natomiast na rys. 3.22 przedstawiono płaszczyzny określone przez wyjścia modeli wielomianowych czwartego i dziesiątego rzędu dla zbioru danych weryfikujących oraz błędy modeli. W odróżnieniu od gładkiej płaszczyzny modelu neuronowego (rys. 3.21), płaszczyzna modelu niższego rzędu znacznie odbiega od charakterystyki statycznej procesu. Model zupełnie nie uwzględnia specyficznego pofałdowania. Model dziesiątego rzędu ma natomiast bardzo duże trudności z aproksymacją płaskich obszarów charakterystyki.

Tabela 3.3. Wpływ liczby neuronów ukrytych (K) na liczbę parametrów i dokładność modelu neuronowego

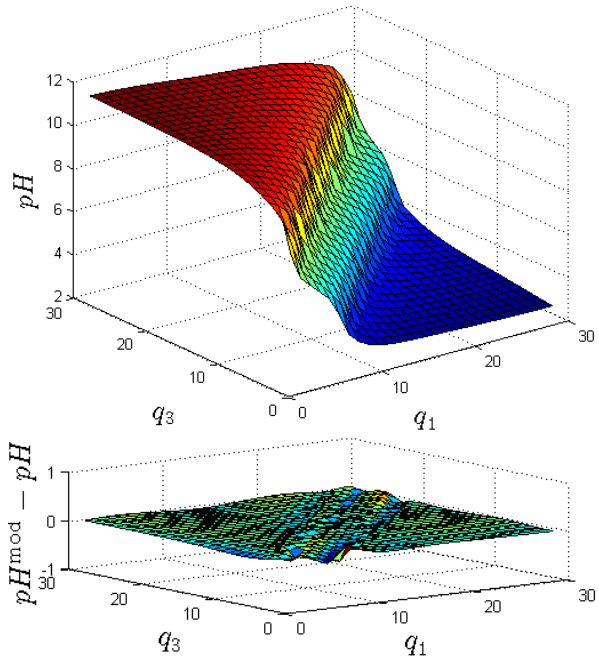
K	Parametry	Zbiór uczący	Zbiór weryfikujący	Zbiór testowy
1	5	$9{,}785150 \cdot 10^{0}$	$9{,}053030 \cdot 10^{0}$	–
2	9	$8{,}421854 \cdot 10^{0}$	$7{,}480121 \cdot 10^{0}$	–
3	13	$2{,}269748 \cdot 10^{0}$	$2{,}284515 \cdot 10^{0}$	–
4	17	$8{,}120631 \cdot 10^{-1}$	$8{,}448123 \cdot 10^{-1}$	–
5	21	$3{,}544456 \cdot 10^{-1}$	$3{,}732008 \cdot 10^{-1}$	–
6	25	$1{,}373313 \cdot 10^{-1}$	$1{,}549976 \cdot 10^{-1}$	$1{,}329495 \cdot 10^{-1}$
7	29	$8{,}696929 \cdot 10^{-2}$	$1{,}059207 \cdot 10^{-1}$	–
8	33	$6{,}443916 \cdot 10^{-2}$	$8{,}265459 \cdot 10^{-2}$	–
9	37	$3{,}712659 \cdot 10^{-2}$	$5{,}610176 \cdot 10^{-2}$	–
10	41	$3{,}206455 \cdot 10^{-2}$	$3{,}950227 \cdot 10^{-2}$	–

Tabela 3.4. Wpływ rzędu (n) na liczbę parametrów i dokładność modelu wielomianowego

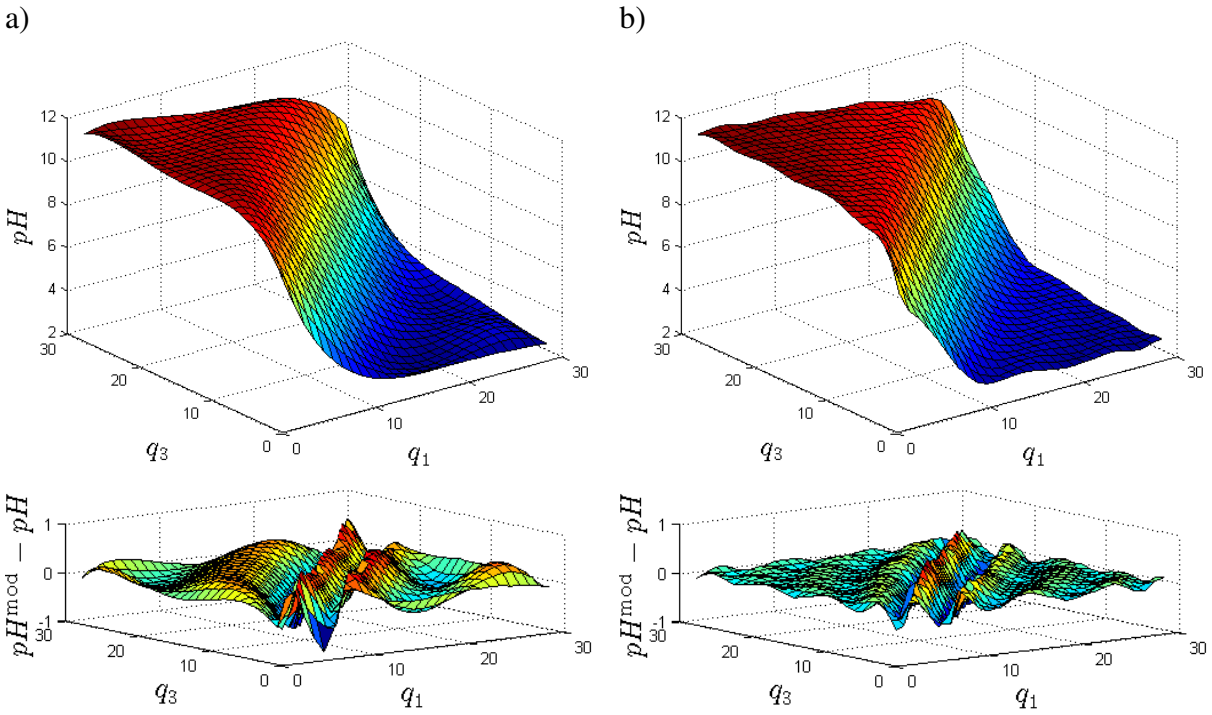
n	Parametry	Zbiór uczący	Zbiór weryfikujący
1	3	$2{,}022207 \cdot 10^{2}$	$2{,}044060 \cdot 10^{2}$
2	9	$5{,}608685 \cdot 10^{1}$	$5{,}437370 \cdot 10^{1}$
3	16	$2{,}317827 \cdot 10^{1}$	$2{,}170247 \cdot 10^{1}$
4	25	$1{,}206249 \cdot 10^{1}$	$1{,}135175 \cdot 10^{1}$
5	36	$8{,}011410 \cdot 10^{0}$	$7{,}161093 \cdot 10^{0}$
6	49	$6{,}577970 \cdot 10^{0}$	$5{,}899105 \cdot 10^{0}$
7	64	$5{,}972083 \cdot 10^{0}$	$5{,}158721 \cdot 10^{0}$
8	81	$5{,}522579 \cdot 10^{0}$	$4{,}701767 \cdot 10^{0}$
9	100	$5{,}003074 \cdot 10^{0}$	$4{,}259483 \cdot 10^{0}$
10	121	$4{,}430965 \cdot 10^{0}$	$3{,}796229 \cdot 10^{0}$



Rys. 3.20. Dane weryfikujące oraz wyjście a) modelu neuronowego z K=6 neuronami ukrytymi, b) modelu wielomianowego rzędu n=4 (oba modele mają 25 parametrów)



Rys. 3.21. Wyjście wybranego modelu neuronowego z K=6 neuronami ukrytymi (płaszczyzna) dla zbioru danych weryfikujących oraz błąd modelu



Rys. 3.22. Wyjście modelu wielomianowego (płaszczyzna): a) rzędu n=4, b) oraz rzędu n=10 dla zbioru danych weryfikujących oraz błędy modeli

3.5.3. Metody redukcji sieci

Po skończeniu identyfikacji modelu neuronowego może okazać się, że model ma dość dużą liczbę neuronów, a tym samym wag. Z drugiej strony wiadomo, że np. modelowany proces dynamiczny ma właściwości inercyjne, jedynie jego wzmocnienie zależy od punktu pracy. Intuicyjnie, do modelowania takiego procesu powinna wystarczyć sieć o kilkunastu, kilkudziesięciu wagach. Warto wówczas wyeliminować mało istotne wagi sieci, ale przy zachowaniu odpowiedniej dokładności i zdolności generalizacji modelu. Istnieją specjalne algorytmy redukcji sieci, spadkowi liczby wag zwykle towarzyszy zwiększenie generalizacji sieci.

Najprostszym kryterium redukcji wag jest ich wartość (moduł wartości). Uznaje się, że wagi o wartościach znacznie mniejszych od pozostałych mają niewielki wpływ na wyjście modelu i zostają one usunięte (ich wartość wynosi 0). Niestety, takie rozumowanie w przypadku modeli nieliniowych, jakimi są sieci neuronowe, nie zawsze jest uzasadnione. Niektóre małe wagi są istotne z punktu widzenia całego modelu i nie mogą być usunięte.

Inną koncepcją, potencjalnie lepszą, jest uwzględnienie podczas redukcji nie wartości wag, ale wrażliwości sieci na zmiany wag. Wagi o najmniejszej wrażliwości, bez względu na wartość, mogą być usunięte. Stosując rozwinięcie Taylora funkcji błędu sieci, zmiana wartości tej funkcji spowodowana perturbacją wag jest następująca

$$\Delta E = \sum_i g_i \Delta w_i + \frac{1}{2}\left(\sum_i h_{i,i} (\Delta w_{i,i})^2 + \sum_{i \neq j} h_{i,j} \Delta w_i \Delta w_j \right) + \ldots \tag{3.64}$$

gdzie Δw_i oznacza perturbację i-tej wagi, g_i jest i-tym składnikiem wektora gradientu względem tej wagi, natomiast wielkości h_{ij} są elementami macierzy drugich pochodnych (hesjanu)

$$g_i = \frac{\partial E}{\partial w_i}, \quad h_{i,j} = \frac{\partial^2 E}{\partial w_i \partial w_j} \tag{3.65}$$

Redukcja sieci musi odbywać się po jej nauczeniu. Dla modelu o małym błędzie elementy wektora gradientu są małe (idealnie w minimum powinny być równe 0). Dlatego też w celu określenia istotności wag wykorzystuje się tylko drugie pochodne funkcji błędu.

Ze względu na prostotę i skuteczność warto omówić metodę OBD (ang. Optimal Brain Damage) redukcji wag sieci [29]. Dla uproszczenia przyjmuje się, że macierz drugich pochodnych $\boldsymbol{H}$ jest diagonalnie dominująca. Uwzględnia się zatem wyłącznie jej elementy diagonalne, wszystkie pozostałe są pomijane. Miarą wrażliwości wagi w_i jest współczynnik asymetrii

$$S_i = \frac{\partial^2 E}{\partial (w_i)^2} (w_i)^2 \tag{3.66}$$

Wagi, które mają najmniejsze wartości współczynnika asymetrii mogą zostać usunięte, gdyż nie mają one istotnego wpływu na działanie sieci. Algorytm OBD jest następujący:

1. Nauczenie pełnej sieci neuronowej.
2. Obliczenie elementów diagonalnych macierzy drugich pochodnych oraz współczynników asymetrii S_i dla wszystkich wag sieci.
3. Obcięcie wagi (lub wag) o najmniejszych współczynnikach asymetrii.
4. Douczenie sieci z usuniętymi wagami.
5. Jeżeli bieżąca redukcja prowadzi do zwiększenia błędu sieci (w porównaniu z siecią z poprzedniej iteracji), następuje zatrzymanie algorytmu, wynikiem jest sieć otrzymana w poprzedniej iteracji.
6. Przejście do kroku 2.

Inną metodą redukcji sieci jest algorytm OBS (ang. Optimal Brain Surgeon) [29]. Analogicznie jak w metodzie OBD uznaje się, że wstępnie dobrana sieć jest w pełni nauczona, a więc elementy wektora gradientu są zerowe. Uwzględnia się natomiast wszystkie (nie tylko diagonalne) elementy macierzy drugich pochodnych. Współczynniki asymetrii definiuje się jako

$$S_i = \frac{1}{2} \frac{(w_i)^2}{(\boldsymbol{H}^{-1})_{i,i}} \tag{3.67}$$

W odróżnieniu od metody OBD, po obcięciu wybranej wagi wartości pozostałych wag są modyfikowane w sposób analityczny, bez potrzeby douczenia sieci (zapis wektorowy)

$$\boldsymbol{w} := \boldsymbol{w} + \frac{w_i}{(\boldsymbol{H}^{-1})_{i,i}} \boldsymbol{H}^{-1} \boldsymbol{e}_i \tag{3.68}$$

gdzie $\boldsymbol{e}_i$ jest wektorem z 1 na *i*-tej pozycji i zerami na pozostałych. Algorytm OBS jest następujący:

1. Nauczenie pełnej sieci neuronowej.
2. Obliczenie macierzy odwrotnej drugich pochodnych $\boldsymbol{H}^{-1}$ oraz współczynników asymetrii S_i dla wszystkich wag sieci.
3. Obcięcie wagi (lub wag) o najmniejszych współczynnikach asymetrii.

4. Jeżeli bieżąca redukcja prowadzi do zwiększenia błędu sieci (w porównaniu z siecią z poprzedniej iteracji), następuje zatrzymanie algorytmu, wynikiem jest sieć otrzymana w poprzedniej iteracji.
5. Korekta pozostałych wag.
6. Przejście do kroku 2.

Zastosowanie przedstawionych powyżej wrażliwościowych algorytmów redukcji musi być poprzedzone nauczeniem modelu neuronowego. Podejściem alternatywnym jest taka organizacja procesu uczenia, która wymusza zmniejszanie wartości wag. Modyfikuje się mianowicie minimalizowaną podczas uczenia funkcję błędu sieci wprowadzając dodatkowy człon kary. Dzięki temu podczas uczenia wartości wag ulegają ograniczeniu, gdy wartość niektórych z nich spadnie poniżej pewnego progu, są one eliminowane.

Najprostszą koncepcją jest dodanie do funkcji błędu sieci składnika kary proporcjonalnego do kwadratu wartości wag

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{s=1}^{S}\sum_{l=1}^{M}(y_l(s) - d_l(s))^2 + \gamma\left(\sum_{i=1}^{K}\sum_{j=0}^{N}(w_{i,j}^1)^2 + \sum_{i=1}^{M}\sum_{j=0}^{K}(w_{i,j}^2)^2\right) \tag{3.69}$$

Niestety, powyższa funkcja celu wymusza zmniejszanie wszystkich wag, nawet wówczas, gdy w uczonej sieci wagi powinny mieć duże wartości. Znacznie lepsze rezultaty można osiągnąć tak modyfikując funkcję celu, aby eliminować neurony ukryte o najmniejszej zmianie aktywności w trakcie uczenia. Funkcja celu może mieć postać

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{s=1}^{S}\sum_{l=1}^{M}(y_l(s) - d_l(s))^2 + \gamma\left(\sum_{i=1}^{K}\sum_{s=1}^{S}e(\Delta_{i,s}^2)\right) \tag{3.70}$$

przy czym $\Delta_{i,j}^2$ oznacza zmianę wartości sygnału wyjściowego *i*-tego neuronu ukrytego dla *s*-tej próbki uczącej, natomiast $e(\Delta_{i,s}^2)$ stanowi czynnik korekcyjny funkcji celu. Postać funkcji korekcyjnej dobiera się w taki sposób, aby zmiana funkcji celu była uzależniona od aktywności neuronu ukrytego. Przy dużej aktywności zmiana powinna być mała, przy małej aktywności – duża. Można przyjąć

$$\frac{\partial e(\Delta_{i,s}^2)}{\partial \Delta_{i,s}^2} = \frac{1}{(1+\Delta_{i,s}^2)^n}, \quad \text{np. dla } n=2 \quad e(\Delta_{i,s}^2) = \frac{1}{1+\Delta_{i,s}^2} \tag{3.71}$$

Mała aktywność neuronów jest karana bardziej niż duża. Neurony pasywne zostają wyeliminowane.

3.5.4. Metody rozbudowy sieci

Podejściem alternatywnym w stosunku do redukcji sieci jest rozbudowa sieci [7]. Wejścia i wyjścia sieci są zdefiniowane przez specyfikę problemu, adaptacji podlega liczba neuronów ukrytych. Na początku procesu doboru architektury przyjmuje się, że sieć jest bardzo prosta, zawiera np. jeden neuron ukryty. W trakcie uczenia, w miarę potrzeb, dodawane są automatycznie kolejne neurony. Rozbudowa sieci trwa do momentu redukcji błędu modelu do akceptowalnej wartości. Po dodaniu pojedynczego neuronu sieć jest oczywiście douczana. Istotną cechą algorytmu, znanego także pod nazwą dynamicznej kreacji neuronów, jest to, że po dodaniu kolejnego neuronu ukrytego proces uczenia jest inicjowany z wagami otrzymanymi dla mniejszej sieci. W sposób losowy są jedynie inicjowane wagi związane z nowo dodanym neuronem.

Autorzy metod rozrostu sieci podają przykłady świadczące o ich skuteczności. Z drugiej

jednak strony, po wykonaniu wielu eksperymentów można zauważyć, że często stosunkowo szybko (w małej liczbie cykli uczących) można nauczyć sieć o nadmiarowej w stosunku do spodziewanej liczbie neuronów, natomiast uczenie sieci o małej liczbie neuronów jest trudne – bardzo często algorytm kończy działanie w płytkim minimum lokalnym. Opisane spostrzeżenie może być inspiracją do nauczenia sieci z nadmiarowymi neuronami, a później jej redukcji, np. stosunkowo prostym, ale skutecznym algorytmem OBD.

Wśród wielu istniejących metod rozbudowy sieci należy wyróżnić algorytm kaskadowej korelacji Fahlmana [29]. W przeciwieństwie do innych metod jest ona bardzo skuteczna. Wagi każdego neuronu zostają dobrane w taki sposób, aby był on użyteczny z punktu widzenia całej sieci. Warto sobie zdać sprawę, że w klasycznych sieciach w trakcie uczenia zmiany wag nie są skoordynowane. Każdy neuron ma dostarczoną informację o własnym sygnale wejściowym i sygnale błędu propagacji wstecznej. Co więcej, w trakcie uczenia nie ma komunikacji między poszczególnymi neuronami, wagi każdego neuronu adaptowane są niezależnie. W algorytmie kaskadowej korelacji Fahlmana uczenie rozpoczyna się od sieci, która nie zawiera żadnych neuronów ukrytych, w trakcie uczenia neurony dodawane są pojedynczo. Podczas dodawania neuronu stosuje się specjalną procedurę, która najpierw kształtuje jego wagi wejściowe, a następnie je zamraża. Początkowo, sygnał wyjściowy nowego neuronu nie jest nigdzie podłączony. W trakcie uczenia maksymalizuje się korelację między aktywnością neuronu (mierzoną sygnałem wyjściowym) a błędem na wyjściu sieci. W celu uzyskania najlepszej korelacji trenuje się kilka neuronów kandydatów, zamiast jednego. Po dołączeniu najlepszego neuronu (dającego maksymalną korelację) następuje ponowne douczenie wag neuronów ukrytych.

3.5.5. Eliminacja nieistotnych wejść sieci

Przy modelowaniu skomplikowanych zjawisk lub procesów wejścia sieci zwykle dobiera się w sposób eksperymentalny, ucząc wiele różnych modeli o różnych wejściach. Otrzymany model może być jednak zbyt skomplikowany. Redukcję sieci prowadzącą do eliminacji niektórych wag warto wówczas poprzedzić próbą ograniczenia wejść sieci. Eliminacja wejść może być dokonana w wyniku tzw. analizy wrażliwościowej danych uczących. Wrażliwość l-tego wyjścia sieci na i-ty składnik wektora wejściowego $\boldsymbol{x}$ przy s-tej próbce definiuje się jako

$$S_{l,i}(s) = \frac{\partial y_l(s)}{\partial x_i(s)} \tag{3.72}$$

Wynik dla s-tego wzorca uczącego należy uśrednić. Można zastosować normę Euklidesową $S_{l,i}^{\text{śr}}$, normę wartości bezwzględnej $S_{l,i}^{\text{abs}}$ lub normę maksimum

$$S_{l,i}^{\text{śr}} = \sqrt{\frac{\sum_{s=1}^{S}(S_{l,i}(s))^2}{S}}, \quad S_{l,i}^{\text{abs}} = \frac{\sum_{s=1}^{S}|S_{l,i}(s)|}{S}, \quad S_{l,i}^{\max} = \max_{s=1,\ldots,S} S_{l,i}(s) \tag{3.73}$$

Dane muszą występować w jednakowej skali. W przeciwnej sytuacji, dane muszą być przeskalowane lub wrażliwość powinna być zdefiniowana w sposób względny jako

$$S_{l,i}(s) = \frac{\partial y_l(s)}{\partial x_i(s)} \frac{x_i(s)}{y_l(s)} \tag{3.74}$$

Względna istotność i-tego wejścia sieci jest następująca

$$F_i = \max_{l=1,\ldots,M} S_{l,i} \tag{3.75}$$

Szeregując wielkości F_i od największej do najmniejszej i uwzględniając różnice między kolejnymi wyrazami szeregu wnioskuje się o ich wpływie na wynik końcowy. Jeżeli wyraźnie widać różnicę między dwoma kolejnymi wyrazami szeregu, z sieci można usunąć wejścia odpowiadające tym małym wielkościom F_i.

3.6. Sieci neuronowe o radialnych funkcjach bazowych

W najpopularniejszych wielowarstwowych sieciach neuronowych typu perceptronowego stosuje się zazwyczaj sigmoidalne funkcje aktywacji. Konsekwencją tego faktu jest to, że neuron pozostaje aktywny wówczas, gdy suma jego sygnałów wejściowych jest większa od pewnej wartości progowej. W rezultacie, wszystkie neurony aktywne uczestniczą w formowaniu sygnału wyjściowego sieci. Klasyczne sieci neuronowe nazywa się czasami aproksymatorami globalnymi [29].

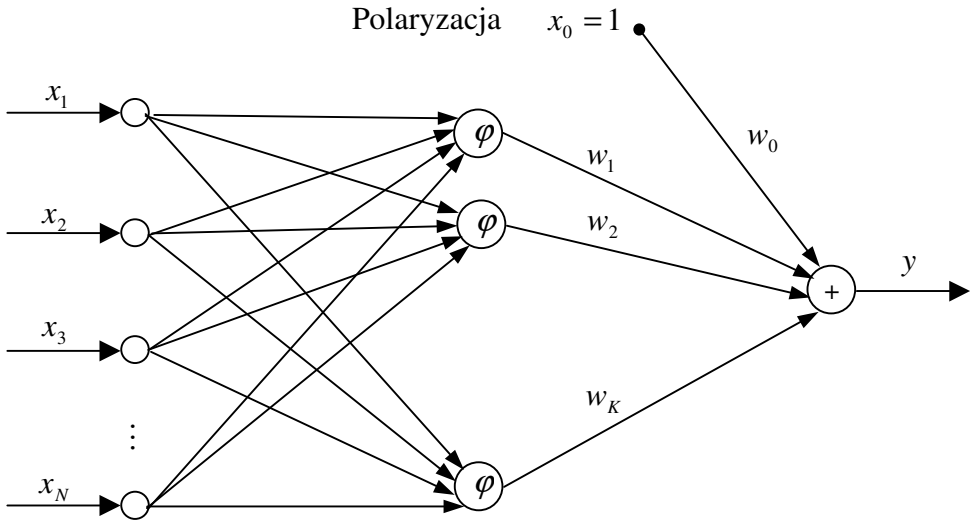
Podejściem alternatywnym jest aproksymacja lokalna, w której sygnał wyjściowy sieci jest sumą odwzorowań lokalnych. Neurony ukryte stanowią zbiór funkcji bazowych typu lokalnego. Poszczególne neurony są aktywne tylko w wąskim obszarze przestrzeni danych. W opisywany sposób działają sieci neuronowe o radialnych funkcjach bazowych (RBF – ang. Radial Basis Function). Neurony ukryte realizują funkcje zmieniające się radialnie wokół centrum $\boldsymbol{c}$ i przyjmujące wartości niezerowe wyłącznie w otoczeniu centrum. Ogólna postać radialnych funkcji bazowych jest następująca

$$\varphi(\boldsymbol{x}) = \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|\right) \tag{3.76}$$

Zakładając dla uproszczenia, że sieć ma tylko jedno wyjście, jest ono opisane zależnością

$$y(\boldsymbol{x}) = w_0 + \sum_{i=1}^{K} w_i \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|\right) \tag{3.77}$$

przy czym K jest liczbą neuronów ukrytych, natomiast wagi sieci oznaczone są przez w_i ($i = 1,\ldots,K$). Wektory podawane na wejścia sieci mają, analogicznie jak w sieciach perceptronowych, długość N, tzn. $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \ldots & x_N \end{bmatrix}^{\mathrm{T}}$. Wektory $\boldsymbol{c}_i = \begin{bmatrix} c_{i,1} & x_{i,2} & \ldots & x_{i,N} \end{bmatrix}^{\mathrm{T}}$ reprezentują centra poszczególnych funkcji bazowych. Ogólna struktura sieci radialnej została przedstawiona na rys. 3.23. Otrzymana sieć ma strukturę dwuwarstwową, neurony ukryte o radialnych funkcjach bazowych są oczywiście nieliniowe, natomiast węzeł wyjściowy (sumator) jest liniowy. Analogicznie jak w sieciach perceptronowych dodatkowe wejście $x_0 = 1$ jest polaryzacją sieci.



Rys. 3.23. Ogólna struktura radialnej sieci neuronowej

Najczęściej używaną funkcją radialną jest funkcja Gaussa

$$\varphi(\boldsymbol{x}) = \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|\right) = \exp\left(-\frac{\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|}{2\sigma_i^2}\right) \tag{3.78}$$

gdzie parametry σ_i $(i = 1,\ldots,K)$ decydują o szerokości funkcji. Wpływ tego parametru na kształt funkcji Gaussa przedstawiono na rys. 3.24. Wyjście sieci z radialną funkcją Gaussa można również zapisać w następujący sposób

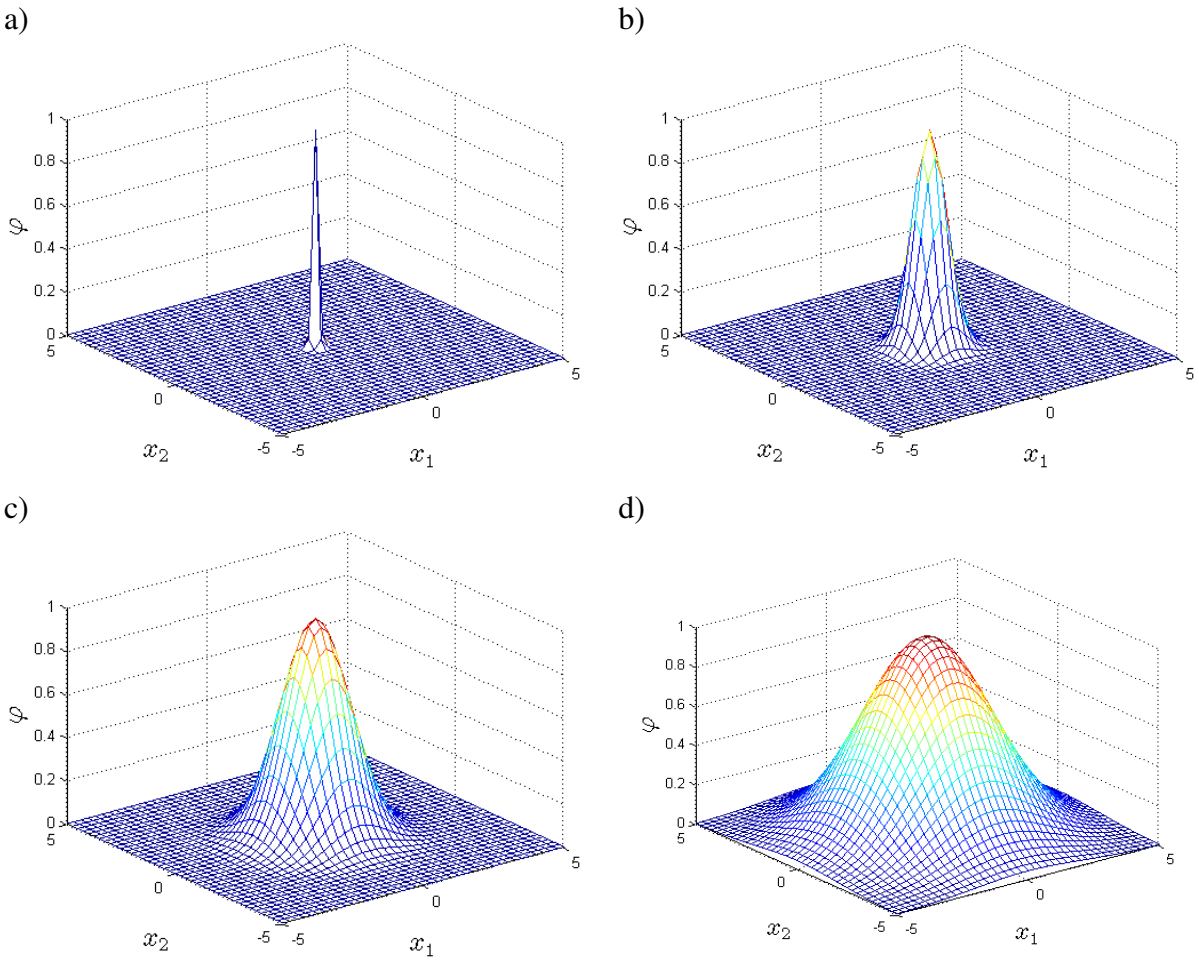
$$y(\boldsymbol{x}) = w_0 + \sum_{i=1}^{K} w_i \exp\left(-\frac{1}{2\sigma_i^2}\sum_{j=1}^{N}(x_j - c_{i,j})^2\right) \tag{3.79}$$

W przeciwieństwie do sieci perceptronowych, które mogą mieć dowolną liczbę warstw, sieć radialna ma stałą strukturę. Co więcej, wszystkie nieliniowe neurony warstwy ukrytej stosowane w sieciach perceptronowych mają zazwyczaj taką samą funkcję aktywacji, o takich samych parametrach (np. tangens hiperboliczny), natomiast parametry $\boldsymbol{c}_i$ funkcji radialnych poszczególnych neuronów są, z definicji, inne. W przeciwnym wypadku wszystkie neurony działałyby lokalnie wokół tych samych punktów przestrzeni wielowymiarowej wektora wejściowego. Parametry σ_i poszczególnych funkcji radialnych też są zazwyczaj różne, jedynie w uproszczonych przypadkach są one stałe.

Ponieważ rząd poszczególnych składowych wektora wejściowego może być różny, dobrze jest zastosować skalowanie funkcji aktywacji poszczególnych neuronów. Macierze skalujące o wymiarowości $N \times N$ oznaczone są przez $\boldsymbol{Q}_i$ $(i = 1,\ldots,K)$. Otrzymuje się wówczas uogólnioną funkcję Gaussa postaci

$$\begin{aligned}\varphi(\boldsymbol{x}) = \varphi\left(\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|_{\boldsymbol{Q}_i}\right) &= \exp\left(-\frac{\left\|\boldsymbol{x} - \boldsymbol{c}_i\right\|_{\boldsymbol{Q}_i}}{2\sigma_i^2}\right) = \exp\left(-(\boldsymbol{x} - \boldsymbol{c}_i)^{\mathrm{T}}\boldsymbol{Q}_i^{\mathrm{T}}\boldsymbol{Q}_i(\boldsymbol{x} - \boldsymbol{c}_i)\right) \\ &= \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{c}_i)^{\mathrm{T}}\boldsymbol{C}_i(\boldsymbol{x} - \boldsymbol{c}_i)\right)\end{aligned} \tag{3.80}$$

gdzie macierz $\frac{1}{2}\boldsymbol{C}_i = \boldsymbol{Q}_i^{\mathrm{T}}\boldsymbol{Q}_i$ pełni rolę skalarnego czynnika $\frac{1}{2\sigma_i^2}$ standardowej funkcji Gaussa (3.78). Sieci, w których wykorzystuje się macierze skalujące nazywane są sieciami HRBF (ang. Hyper Radial Basis Function). Elementy macierzy skalujących są dodatkowymi parametrami modelu neuronowego, ich dobór pozwala dopasować sieć do rozwiązywanego problemu. Z drugiej jednak strony, wprowadzenie macierzy skalujących powoduje, że liczba parametrów sieci gwałtownie wzrasta, co ma duże znaczenie podczas uczenia. Dlatego też największe praktyczne znacznie mają sieci HRBF o diagonalnych macierzach skalujących $\boldsymbol{Q}_i = \mathrm{diag}(q_{i,1},\ldots,q_{i,N})$.



Rys. 3.24. Wykresy funkcji bazowej Gaussa: a) $\sigma = 0{,}1$, b) $\sigma = 0{,}5$, c) $\sigma = 1$, d) $\sigma = 2$

3.6.1. Uczenie sieci neuronowych o radialnych funkcjach bazowych

Uczenie, czyli dobór parametrów sieci radialnych można sformułować jako minimalizację błędu modelu dla wszystkich *S* próbek

$$E = \frac{1}{2}\sum_{s=1}^{S}(y(s)-d(s))^2 = \frac{1}{2}\sum_{s=1}^{S}\left(w_0 + \sum_{i=1}^{K} w_i\varphi\left(\|\boldsymbol{x}-\boldsymbol{c}_i\|\right) - d(s)\right)^2 \tag{3.81}$$

W wyniku rozwiązania powyższego zadania optymalizacji wyznacza się wagi sieci, centra funkcji bazowych oraz parametry określające kształt funkcji bazowych. W przypadku klasycznych sieci RBF będą to współczynniki σ_i $(i = 1,\ldots,K)$, dla sieci HRBF będą to

macierze $\boldsymbol{Q}_i$ $(i = 1,\ldots,K)$, natomiast dla uproszczonych sieci HRBF będą to diagonalne macierze $\boldsymbol{Q}_i$.

Uwzględniając wyłącznie wagi sieci, funkcja błędu jest kwadratowa, a więc jeżeli optymalizacji podlegają tylko wagi, to zadanie optymalizacji można bardzo efektywnie numerycznie (bez iteracji, znajdując minimum globalne) rozwiązać metodą najmniejszych kwadratów. Spostrzeżenie to jest podstawą algorytmu hybrydowego uczenia radialnych sieci neuronowych. Składa się on z dwóch, powtarzających się etapów. Struktura algorytmu jest następująca:

1. Wybór początkowych wartości wag (zwykle w sposób losowy), wybór centrów funkcji bazowych, wybór parametrów σ_i lub macierzy $\boldsymbol{Q}_i$.
2. Dobór wag sieci przy wykorzystaniu metody najmniejszych kwadratów.
3. Optymalizacja parametrów σ_i lub $\boldsymbol{Q}_i$ na drodze nieliniowej optymalizacji.
4. Przejście do kroku 2.

Inicjalizacja parametrów funkcji bazowych ma bardzo duże znaczenie, nawet większe niż w przypadku sieci perceptronowych. Jest to spowodowane tym, że funkcje wykładnicze charakteryzują się bardzo silnymi nieliniowościami, prawdopodobieństwo utknięcia w minimum lokalnym jest bardzo duże. Dlatego też stosuje się bardzo efektywne algorytmy samoorganizacji [29].

W drugim kroku algorytmu hybrydowego parametry σ_i lub $\boldsymbol{Q}_i$ są zamrożone, wyznacza się wyłącznie wagi sieci. Dla *S* próbek uczących można sformułować układ równań

$$\begin{bmatrix} 1 & \varphi_1(\boldsymbol{x}(1)) & \varphi_2(\boldsymbol{x}(1)) & \ldots & \varphi_K(\boldsymbol{x}(1)) \\ 1 & \varphi_1(\boldsymbol{x}(2)) & \varphi_2(\boldsymbol{x}(2)) & \ldots & \varphi_K(\boldsymbol{x}(2)) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \varphi_1(\boldsymbol{x}(S)) & \varphi_2(\boldsymbol{x}(S)) & \ldots & \varphi_K(\boldsymbol{x}(S)) \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_K \end{bmatrix} = \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(S) \end{bmatrix} \tag{3.82}$$

który można zapisać jako $\boldsymbol{G}\boldsymbol{w} = \boldsymbol{y}$. Podstawiając w miejsce wektora wyjściowego wzorce $\boldsymbol{d} = [d_1 \quad d_2 \quad \ldots \quad d_S]^{\mathrm{T}}$ można wyznaczyć analitycznie optymalny wektor wag

$$\boldsymbol{w} = \boldsymbol{G}^{+}\boldsymbol{d} \tag{3.83}$$

Do obliczenia macierzy pseudoodwrotnej $\boldsymbol{G}^{+}$ można zastosować rozkład SVD.

W trzecim kroku algorytmu hybrydowego wagi zostają zamrożone, optymalizacji podlegają jedynie parametry mające nieliniowy wpływ na wyjście sieci. Stosuje się do tego celu dowolną procedurę nieliniowej optymalizacji, analogicznie jak ma to miejsce w przypadku klasycznej sieci jednokierunkowej wielowarstwowej, np. procedurę zmiennej metryki lub gradientów sprzężonych. Podczas obliczeń składowe wektora gradientu funkcji kryterialnej *E* względem parametrów σ_i lub $\boldsymbol{Q}_i$ oblicza się w sposób analityczny. Na przykład, dla klasycznej sieci radialnej pochodne względem parametrów σ_i mają postać

$$\begin{aligned} \frac{\partial E}{\partial c_{i,j}} &= \sum_{s=1}^{S} (y(s) - d(s)) \frac{\partial y(s)}{\partial c_{i,j}} \\ &= \sum_{s=1}^{S} (y(s) - d(s)) \left(w_i \exp\left(-\frac{1}{2\sigma_i^2} \sum_{j=1}^{N} (x_j - c_{i,j})^2 \right) \frac{(x_j - c_{i,j})}{\sigma_i^2} \right) \end{aligned} \tag{3.84}$$

dla wszystkich $i = 1,\ldots,K$, $j = 1,\ldots,N$, oraz

$$\begin{aligned}\frac{\partial E}{\partial \sigma_i} &= \sum_{s=1}^{S}(y(s)-d(s))\frac{\partial y(s)}{\partial \sigma_i} \\ &= \sum_{s=1}^{S}(y(s)-d(s))\left(w_i \exp\left(-\frac{1}{2\sigma_i^2}\sum_{j=1}^{N}(x_j - c_{i,j})^2\right)\frac{1}{\sigma_i^3}\right)\end{aligned} \tag{3.85}$$

dla $i = 1,\ldots,K$.

Uczenie sieci radialnych można również zorganizować tak samo jak sieci perceptronowych, gdzie wszystkie parametry sieci są optymalizowane jednocześnie. Wagi względem wag dla $i = 0$ oblicza się ze wzoru

$$\frac{\partial E}{\partial w_0} = \sum_{s=1}^{S}(y(s)-d(s))\frac{\partial y(s)}{\partial w_0} = \sum_{s=1}^{S}(y(s)-d(s)) \tag{3.86}$$

natomiast dla $i = 1,\ldots,K$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \sum_{s=1}^{S}(y(s)-d(s))\frac{\partial y(s)}{\partial w_i} \\ &= \sum_{s=1}^{S}(y(s)-d(s))\left(w_i \exp\left(-\frac{1}{2\sigma_i^2}\sum_{j=1}^{N}(x_j - c_{i,j})^2\right)\right)\end{aligned} \tag{3.87}$$

W praktyce jednak najskuteczniejszy okazuje się algorytm hybrydowy.

3.6.2. Sieci perceptronowe a radialne

Jak już wspomniano na wstępie, klasyczne sieci wielowarstwowe (perceptronowe) są aproksymatorami globalnymi, podczas gdy sieci radialne, w których stosuje się funkcje aktywacji mające wartości niezerowe jedynie w pewnym otoczeniu centrów, są aproksymatorami lokalnymi. Sieci perceptronowe, ze względu na globalny charakter funkcji aktywacji, nie mają wbudowanego mechanizmu pozwalającego zidentyfikować region, w którym aktywność danego neuronu jest największa. Bardzo trudno jest powiązać obszar aktywności poszczególnych neuronów z odpowiednim obszarem danych uczących. Oznacz to, że trudno jest wyznaczyć wartości początkowe wag. Uczenie sieci perceptronowej sprowadza się do nieliniowej optymalizacji, w której podstawowym problemem jest występowanie minimów lokalnych.

Ponieważ stosowane w sieciach radialnych funkcje aktywacji są funkcjami lokalnymi, stosunkowo łatwo można powiązać ich parametry z fizycznym rozmieszczeniem danych uczących. W rezultacie można wyznaczyć wartości początkowe parametrów modelu. Stosując hybrydowy algorytm uczenia sieci radialnych można oddzielić etap doboru parametrów funkcji bazowych od etapu dobory wag, co znacznie upraszcza i przyspiesza uczenie. Można dodatkowo zastosować bardzo skuteczną metodę kontroli liczby neuronów ukrytych – algorytm ortogonalizacji Grahama-Schmidta [29]. Dzięki temu, w odróżnieniu od sieci perceptronowych gdzie architektura sieci dobierana jest zwykle na drodze eksperymentalną metodą prób i błędów (należy zwykle nauczyć wiele sieci o różnej architekturze), dla sieci radialnych kształtowanie architektury jest integralnym fragmentem procesu uczenia.

3.7. Rekurencyjne sieci neuronowe

W ogólności, istnieją dwie metody wprowadzenia dynamiki do sieci neuronowych, a mianowicie zastosowanie zewnętrznej pamięci w postaci linii opóźniającej lub też użycie sprzężeń zwrotnych. Najprostszym sposobem, zastosowanym w modelu neuronowym typu NNOE przedstawionym na rys. 3.7, jest podanie na wejścia klasycznej sieci dwuwarstwowej sygnału wyjściowego w poprzednich chwilach dyskretnych. Model można podzielić na dwie części: nieliniowy statyczny aproksymator (sieć neuronowa) i zewnętrzną pamięć. Dlatego też omawiany model jest często nazywany siecią neuronową z liniami opóźniającymi. Podejście takie jest najczęściej stosowane w praktyce, umożliwia skuteczne modelowanie bardzo wielu nieliniowych procesów dynamicznych. Warto jednak pamiętać, że model taki jest znacznie mniej ogólny niż model w przestrzeni stanu (3.53). Na przykład, model typ NNOE nie nadaje się do procesów, w których występują nieliniowości niejednoznaczne, np. histereza lub luz.

Drugą metodą umożliwiającą wprowadzenie dynamiki do sieci neuronowych jest wprowadzenie do nich sprzężeń zwrotnych. Ze względu na rodzaj sprzężeń zwrotnych można wyszczególnić dwie klasy sieci rekurencyjnych:

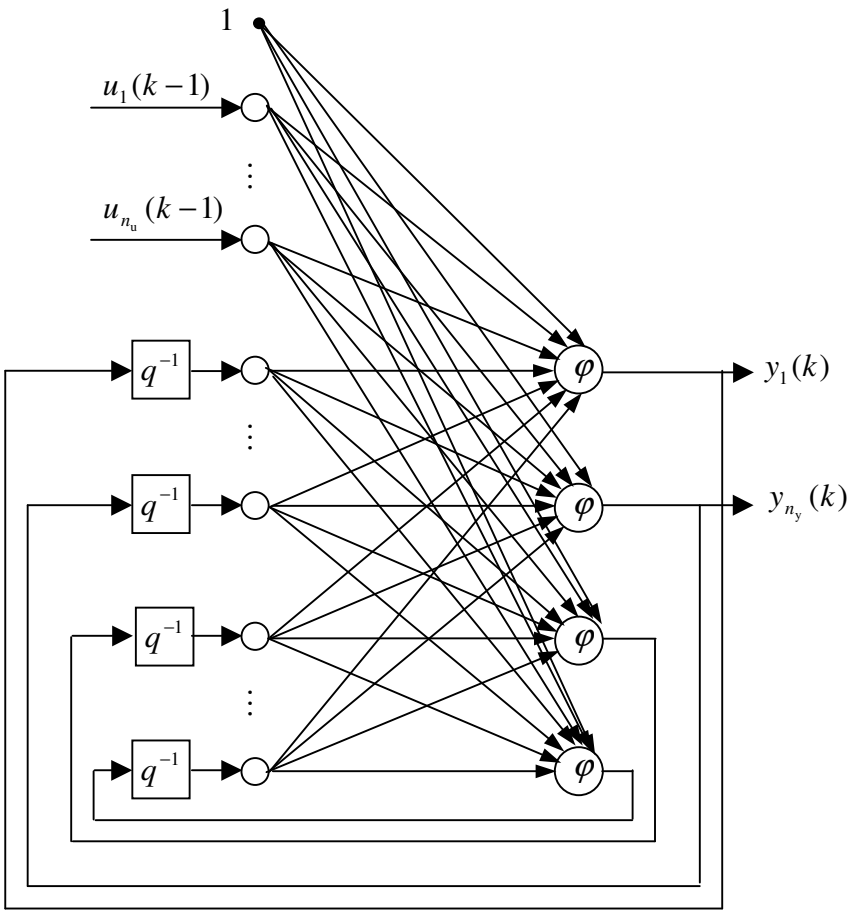
Sieci globalnie rekurencyjne, w których sprzężenia występują między neuronami tej samej warstwy lub między neuronami różnych warstw.

Sieci lokalnie rekurencyjne, w których sprzężenia zwrotne znajdują się wyłącznie wewnątrz pojedynczych neuronów. Ogólna struktura sieci jest natomiast taka sama jak w przypadku klasycznej sieci perceptronowej.

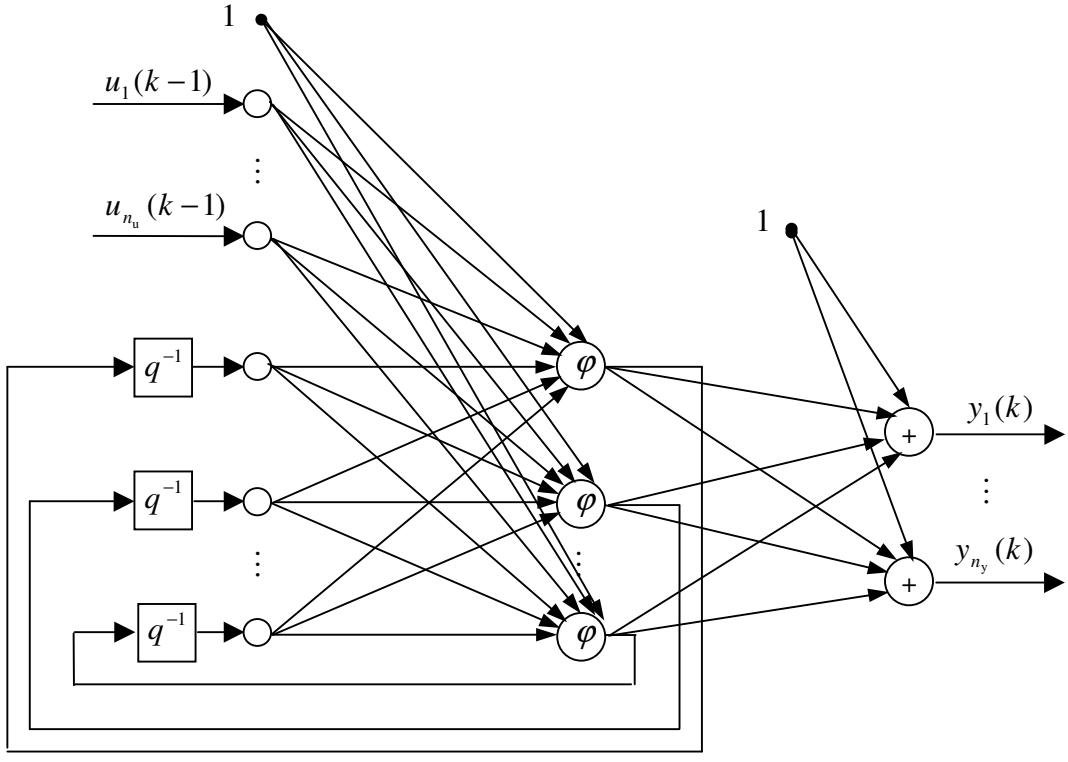
Sieć RTRN (ang. Real Time Recurrent Network) służy do przetwarzania sygnałów w czasie rzeczywistym. Jej struktura została przedstawiona na rys. 3.25. Zawiera ona n_{u} wejść zewnętrznych, na które podawane są sygnały $u_1(k-1), u_2(k-1), \ldots, u_{n_{\mathrm{u}}}(k-1)$. Sieć ma tylko jedną warstwę, liczba neuronów wynosi *K*. Każdy z nich objęty jest sprzężeniem zwrotnym w taki sposób, że sygnały wyjściowe opóźnione o jeden takt podawane są na wejście sieci. Wśród *K* neuronów tylko n_{y} stanowi wyjście sieci. Pozostałe $K - n_{\mathrm{y}}$ neuronów to neurony ukryte, niedostępne dla sygnałów zewnętrznych.

Nieco inną strukturę ma rekurencyjna sieć neuronowa Elmana pokazana na rys. 3.26. Jest to sieć częściowo rekurencyjna o strukturze dwuwarstwowej. Sieć zawiera n_{u} wejść zewnętrznych, liczba neuronów ukrytych wynosi *K*. W odróżnieniu od sieci RTRN, sprzężenie dotyczy wyłącznie warstwy ukrytej. Wyjście sieci stanowią neurony połączone tylko z warstwą ukrytą, podobnie jak w sieciach jednokierunkowych. Przepływ sygnałów odbywa się tylko w jedną stronę, tzn. od warstwy ukrytej do warstwy wyjściowej. Różnica występuje w warstwie ukrytej, ponieważ opóźnione sygnały wyjściowe neuronów ukrytych są podawane na wejścia sieci.

Główną zaletą rekurencyjnych sieci neuronowych typu RTRN i Elmana jest możliwość aproksymacji bardzo szerokiej klasy procesów dynamicznych, mogą być one też zastosowane do opisu szeregów czasowych. Do uczenia sieci stosuje się zazwyczaj algorytmy gradientowe. W porównaniu z algorytmami uczenia klasycznej sieci jednokierunkowej ich złożoność obliczeniowa jest większa, ponieważ obliczając gradient należy uwzględnić rekurencję.

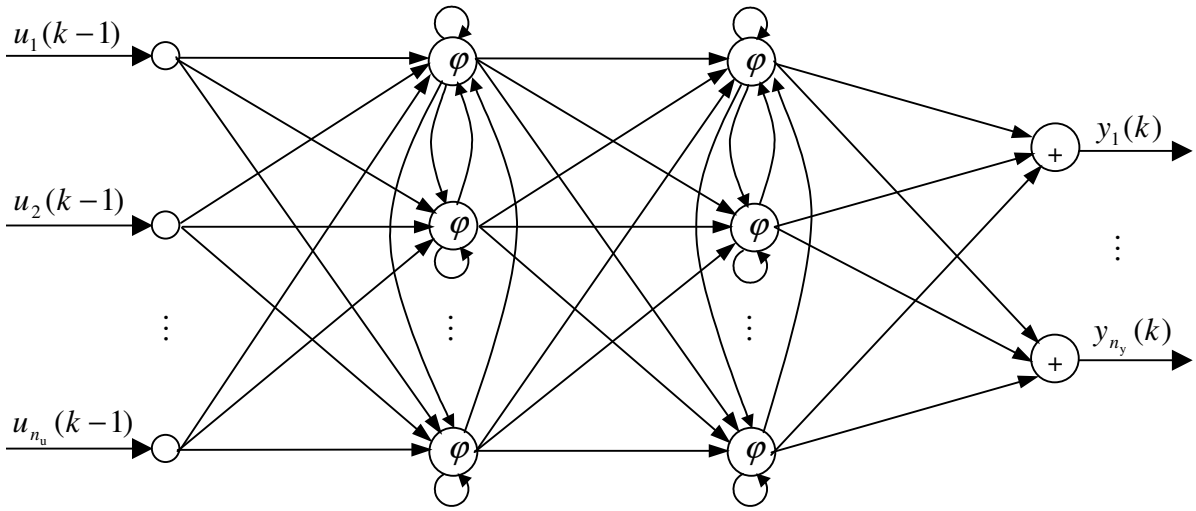


Rys. 3.25. Struktura sieci rekurencyjnej RTRN



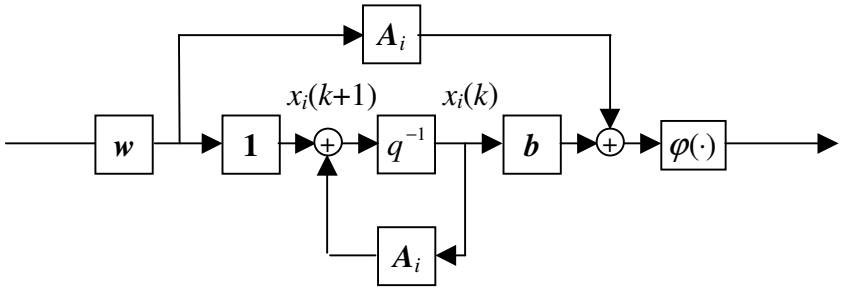
Rys. 3.26. Struktura sieci rekurencyjnej Elmana

Nieco inną strukturę ma sieć neuronowa RMLP (ang. Recurrent Multi Layer Perceptron) pokazana na rys. 3.27. Omawiana sieć jest rozszerzeniem klasycznej sieci jednokierunkowej o dwóch warstwach ukrytych. Istnieją w niej dodatkowe połączenia skrośne między neuronami danej warstwy oraz połączenia rekurencyjne, natomiast poszczególne warstwy są połączone jednokierunkowo. Sieć nadaje się do aproksymacji wielu nieliniowych procesów dynamicznych. Istotną wadą sieci jest jednak znaczna złożoność struktury, co oznacza, że proces uczenia jest zazwyczaj złożony obliczeniowo.

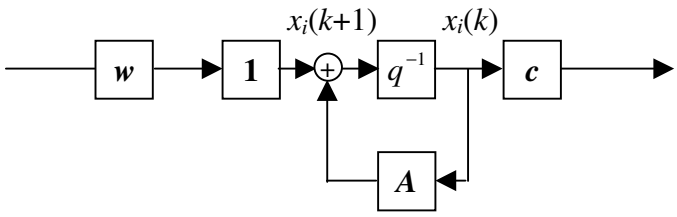


Rys. 3.27. Struktura sieci rekurencyjnej RMLP (dla przejrzystości nie zaznaczono sygnałów polaryzacji)

W stosunku do rekurencyjnych sieci neuronowych omówionych do tej pory interesującą alternatywę stanowią sieci lokalnie rekurencyjne globalnie jednokierunkowe (ang. Locally Recurrent Globally Feedforward, w skrócie LRGF) [30]. Sieci LRGF mają strukturę podobną do jednokierunkowych sieci perceptronowych, ponieważ składają się z kilku warstw neuronów, natomiast wszystkie sygnały przepływają przez sieć tylko w jednym kierunku – od wejść, przez warstwy ukryte, do wejść. W przeciwieństwie do np. rekurencyjnych sieci RTRN lub Elmana, w sieci LRGF nie występują żadne sprzężenia zwrotne między neuronami. Właściwości dynamiczne sieci uzyskuje się w wyniku zastosowania neuronów dynamicznych, które zawierają wewnętrzne sprzężenia zwrotne. Istnieje wiele typów neuronów dynamicznych, najpopularniejsze z nich to neurony z filtrem o Nieskończonej Odpowiedzi Impulsowej (NOI) i neurony z filtrem o Skończonej Odpowiedzi Impulsowej (SOI). Strukturę obu neuronów pokazano na rys. 3.28 i rys. 3.29.



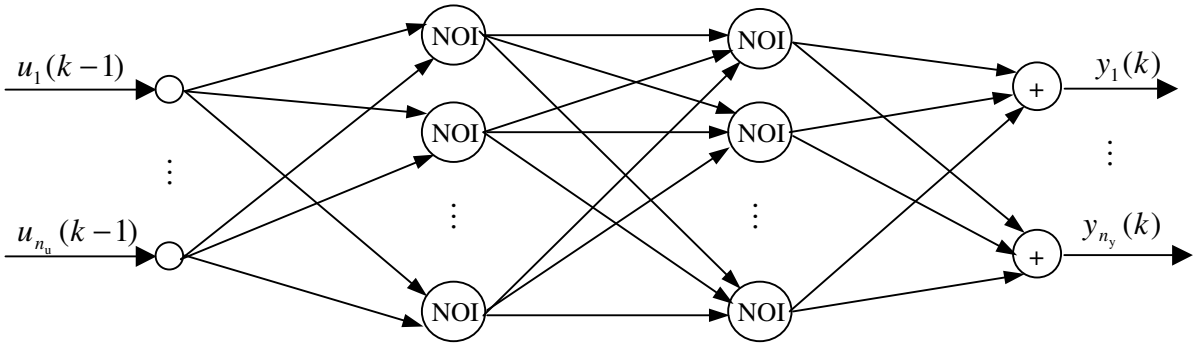
Rys. 3.28. Schemat blokowy neuronu z filtrem NOI



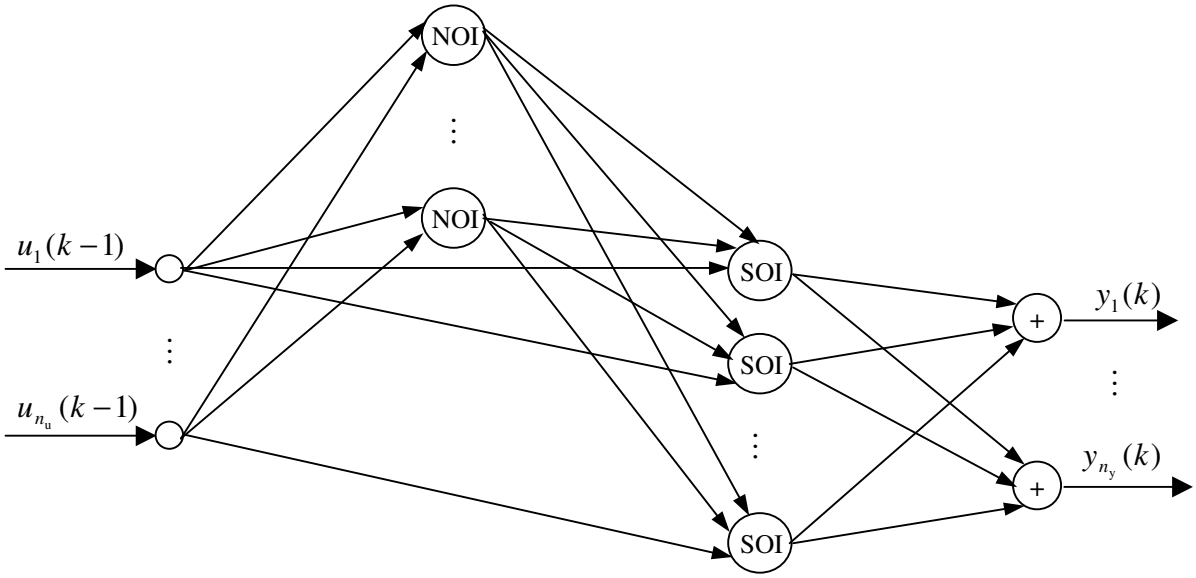
Rys. 3.29. Schemat blokowy neuronu z filtrem SOI

Neurony z filtrem NOI oraz z filtrem SOI mogą posłużyć do budowy sieci jednowarstwowych oraz dwuwarstwowych. Można udowodnić, że sieci dwuwarstwowe, przy odpowiedniej liczbie neuronów z filtrem NOI, są w stanie odtworzyć trajektorię stanu wygenerowaną przez dowolną funkcję klasy C^1 (funkcja i jej pierwsze pochodne są ciągłe). Przedstawiona na rys. 3.30 dwuwarstwowa sieć LRGF może być więc z powodzeniem zastosowane do modelowania bardzo szerokiej klasy procesów dynamicznych.

Istnieje możliwość uproszczenia dwuwarstwowej sieci LRGF z filtrami NOI. W przedstawionej na rys. 3.31 sieci kaskadowej stosuje się w drugiej warstwie prostsze neurony SOI oraz wprowadza się dodatkowe połączenia między wejściami modelu a warstwą ukrytą neuronów z filtrami SOI. Sieć kaskadowa również jest w stanie odtworzyć dowolną trajektorię stanu, natomiast jej struktura i uczenie jest prostsze niż pełnej sieci LRGF.



Rys. 3.30. Struktura sieci lokalnie rekurencyjnej



Rys. 3.31. Struktura kaskadowej sieci lokalnie rekurencyjnej

3.8. Sieci neuronowe w automatyce

Właściwości statyczne i dynamiczne wielu procesów, na przykład reaktorów chemicznych lub kolumn destylacyjnych, są w rzeczywistości nieliniowe. Z tego też powodu algorytmy regulacji opracowane wyłącznie na podstawie modeli liniowych o stałych parametrach mogą działać w sposób niezadowalający, szczególnie wówczas, gdy zmiany punktu pracy, na przykład spowodowane interwencją warstwy optymalizacji, są częste, szybkie i następują w stosunkowo szerokim zakresie.

W ogólności, sieci neuronowe mogą być wykorzystane w układach regulacji w sposób bezpośredni oraz pośredni [27]. W strukturach bezpośrednich sieć neuronowa realizuje algorytm regulacji, oblicza bieżące wartości sygnału (lub sygnałów sterujących). Zasadniczą zaletą podejść bezpośrednich jest łatwość implementacji, obliczanie sygnałów sterujących jest czynnością prostą i szybką. Poważną wadą może być jednak proces uczenia odpowiedniej sieci, ewentualne strojenie lub zmiana parametrów algorytmu wymaga zazwyczaj powtórzenia całej procedury uczenia. Wśród rozwiązań bezpośrednich należy wymienić następujące struktury:

a) sterowanie bezpośrednie z neuronowym modelem odwrotnym,

b) neuronowy algorytm regulacji IMC,

c) układ regulacji z linearyzacją w pętli sprzężenia zwrotnego,

d) układ regulacji PID z siecią neuronową.

Drugą klasę układów regulacji stanowią podejścia, w których sieci neuronowe wykorzystane są w sposób pośredni. Stosowane sieci neuronowe są wówczas modelami dynamicznymi procesów. Algorytmy regulacji wykorzystują te modele do obliczenia wartości aktualnych sygnałów sterujących. W omawianej grupie mieszczą się przede wszystkim omówione w dalszej części rozdziału różnego rodzaju algorytmy regulacji predykcyjnej, w których model jest wykorzystywany do prognozowania zachowania się procesu w przyszłości. Wśród innych podejść bezpośrednich należy wymienić algorytm minimalnowariancyjny lub algorytm z lokowaniem biegunów układu zamkniętego, przy czym oba te rozwiązania wykorzystują lokalne przybliżenie liniowe nieliniowego modelu realizowanego przez sieć neuronową [27].

3.8.1. Układ sterowania bezpośredniego z neuronowym modelem odwrotnym

Historycznie pierwsza koncepcją wykorzystania sieci neuronowych w regulacji polegała na bezpośrednim zastosowaniu neuronowego modelu odwrotnego procesu. Niech model procesu ma postać

$$y(k) = f(u(k-1),\ldots,u(k-n_{\mathrm{B}}), y(k-1),\ldots, y(k-n_{\mathrm{A}})) \tag{3.88}$$

Może być on zapisany dla kolejnej iteracji, tzn. dyskretnej chwili k+1

$$y(k+1) = f(u(k),\ldots,u(k-n_{\mathrm{B}}+1), y(k),\ldots, y(k-n_{\mathrm{A}}+1)) \tag{3.89}$$

Celem sieci neuronowej jest obliczanie bieżącego sygnału sterującego $u(k)$. Może być on wyznaczony przy wykorzystaniu modelu odwrotnego

$$u(k) = f^{-1}(u(k-1),\ldots,u(k-n_{\mathrm{B}}+1), y(k+1), y(k),\ldots, y(k-n_{\mathrm{A}}+1)) \tag{3.90}$$

Wprowadzając żądaną wartość zadaną, sieć powinna obliczać sygnał

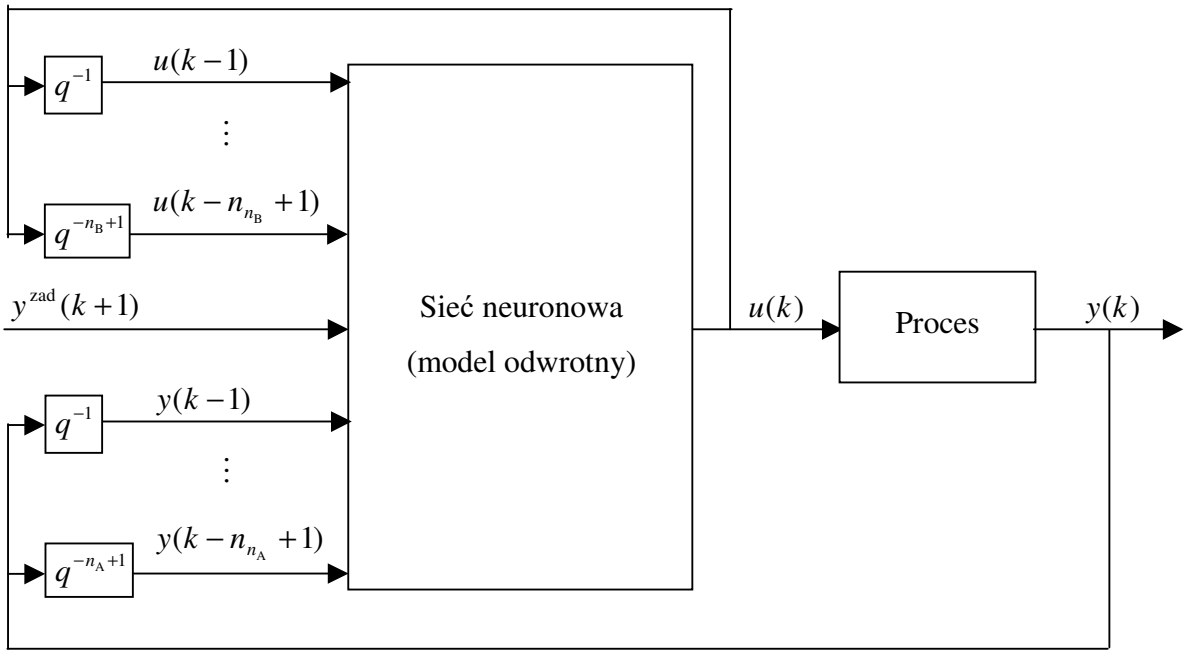
$$u(k) = f^{-1}(u(k-1),\ldots,u(k-n_{\mathrm{B}}+1), y^{\mathrm{zad}}(k+1), y(k),\ldots, y(k-n_{\mathrm{A}}+1)) \tag{3.91}$$

Strukturę układu sterowania bezpośredniego z neuronowym modelem odwrotnym

przedstawiono na rys. 3.32. Zastosowanie modelu odwrotnego linearyzuje nieliniowy obiekt. Co więcej, kompensuje jego dynamikę. Transmitancja między wartością zadaną a wyjściem procesu jest równa opóźnieniu jednostkowemu

$$\frac{y(k+1)}{y^{\text{zad}}(k+1)} = q^{-1} \tag{3.92}$$

Gdy w procesie występuje opóźnienie τ, wyprowadzenie modelu odwrotnego jest trochę bardziej skomplikowane (konieczne jest zastosowanie modelu w sposób rekurencyjny) [27]. Transmitancja między wartością zadaną a wyjściem będzie wówczas równa $q^{-\tau}$.



Rys. 3.32. Struktura układu sterowania bezpośredniego z neuronowym modelem odwrotnym

Istnieją dwie metody uczenia modelu odwrotnego: uczenie ogólne (ang. general training) oraz uczenie specjalizowane (ang. specialised training). W pierwszym trybie sieć neuronowa mająca być modelem odwrotnym jest uczona na podstawie zarejestrowanych wcześniej danych (ang. off-line). W trakcie uczenia minimalizowana jest funkcja kryterialna definiująca błąd modelu odwrotnego

$$E = \sum_{k=1}^{S} (u_{\text{mod}}(k) - u(k))^2 \tag{3.93}$$

przy czym sygnał $u_{\text{mod}}(k)$ jest obliczany przez uczony model odwrotny, $u(k)$ jest zarejestrowanym rzeczywistym sygnałem, S jest liczbą próbek uczących. Warto zwrócić uwagę, że na podstawie danych uczony jest od razu model odwrotny, nie ma potrzeby uczenia zwykłego modelu postaci (3.88).

Przedstawiony układ sterowania z modelem odwrotnym jest koncepcyjnie bardzo prosty. W naturalny sposób wykorzystuje się główną zaletę sieci neuronowych (dobre zdolności aproksymacji) do budowy modelu odwrotnego. Niestety, w praktyce rozwiązanie takie ma wiele poważnych wad wynikających z istoty koncepcji. Po pierwsze, całkowita linearyzacja statyki i kompensacja dynamiki pociąga za sobą konieczność istnienia w układzie bardzo szybko zmieniającego się sygnału sterującego, o bardzo dużej amplitudzie. Po drugie, układ wykazuje małą odporność: zmiana parametrów procesu powoduje znaczne pogorszenie

jakości sterowania. Układ jest również wrażliwy na szumy pomiarowe. Warto również podkreślić, że nie dla wszystkich obiektów model odwrotny istnieje. Ma to miejsce gdy dwa różne sterowania $u_1(k)$ i $u_2(k)$ dają takie same wartości wyjścia

$$\begin{aligned} y(k+1) &= f(u_1(k), u(k-1), \ldots, u(k-n_B+1), y(k), \ldots, y(k-n_A+1)) \\ y(k+1) &= f(u_2(k), u(k-1), \ldots, u(k-n_B+1), y(k), \ldots, y(k-n_A+1)) \end{aligned} \tag{3.94}$$

Jakość modelu odwrotnego jest bardzo uzależniona od sposobu uczenia i dostępnych danych. W trakcie uczenia ogólnego stosuje się najczęściej dane takie, jak podczas uczenia zwykłych modeli (tzn. rejestruje się odpowiedź wyjścia na ciąg skoków sygnału wejściowego o losowej wartości). Często modele uczone na podstawie takich danych nie oddają dokładnie właściwości modelu odwrotnego. W rezultacie, cały układ regulacji nie pracują prawidłowo.

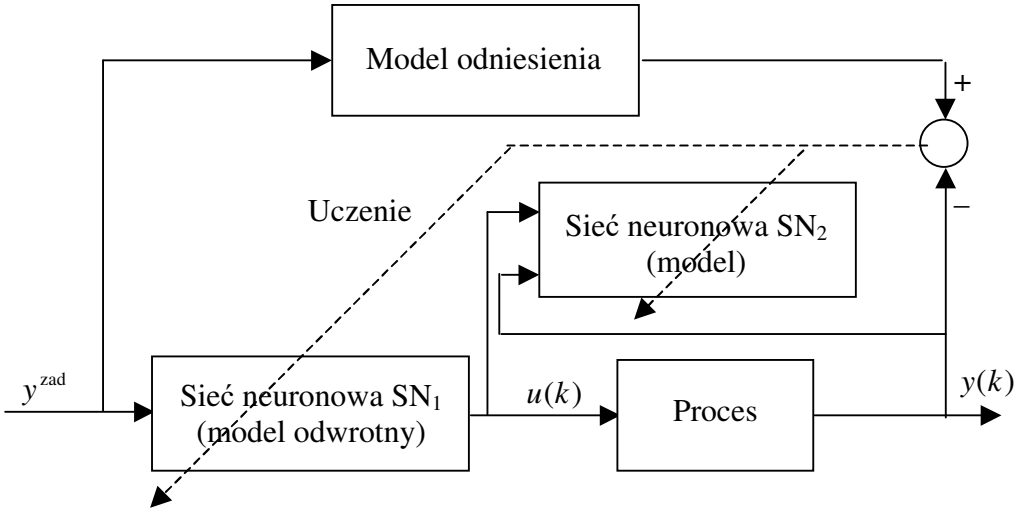
W uczeniu specjalizowanym inaczej zdefiniowany jest cel uczenia sieci neuronowej mającej być modelem odwrotnym. Nie chodzi o odtwarzanie wcześniej zarejestrowanych danych, ale o dobrą regulację. Buduje się więc układ sterowania z modelem odwrotnym, a następnie uczy sieć neuronową, która na bieżąco (ang. on-line) generuje sygnał sterujący. Celem uczenia sieci jest, aby na pewnym horyzoncie symulacji S osiągnąć małe różnice między wartością sygnału wyjściowego a sygnału zadanego w kolejnych iteracjach. Minimalizowana funkcja kryterialna ma postać

$$E = \sum_{k=1}^{S} (y^{\text{zad}}(k) - y(k))^2 \tag{3.95}$$

Uczenie specjalizowane przedstawiono na rys. 3.33. Ponieważ zwykle w układzie z modelem odwrotnym sygnał sterujący zmienia się szybko, można wymusić, aby transmitancja między wartością zadaną a wyjściem procesu nie była równa opóźnieniu jednostkowemu, a pewnej zadanej transmitancji, na przykład jednoinercyjnej

$$\frac{y^{\text{odn}}(k)}{y^{\text{zad}}(k)} = G(q^{-1}) \tag{3.96}$$

Dobierając model odniesienia można wpływać na jakość pracy całego układu sterowania.



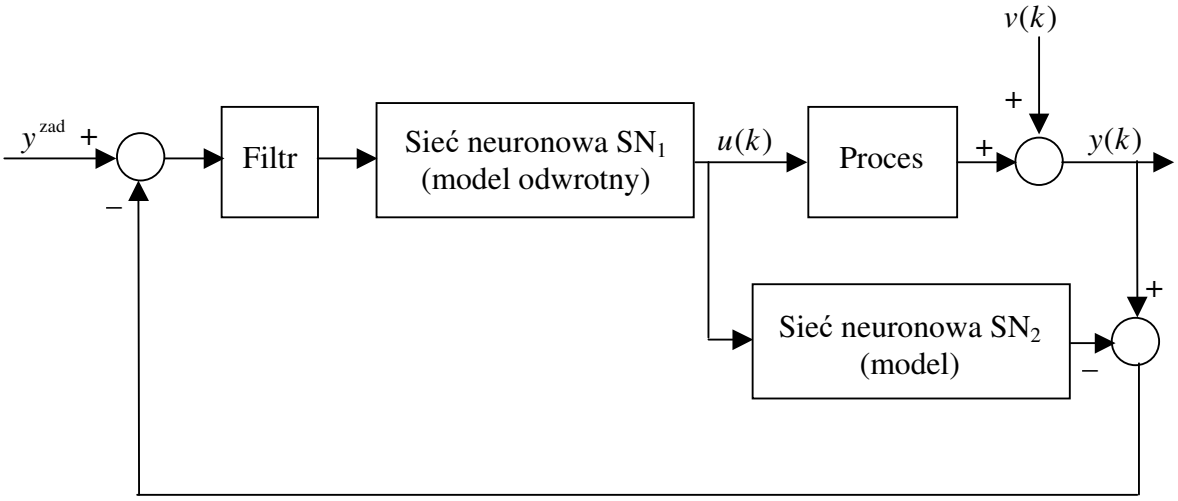
Rys. 3.33. Uczenie specjalizowane do celów sterowania bezpośredniego z neuronowym modelem odwrotnym

Przed przystąpieniem do uczenia specjalizowanego warto nauczyć neuronowy model odwrotny w trybie ogólnym [27]. Warto nie wyłączać uczenia modelu odwrotnego, nawet

wówczas, gdy wydaje się, że sieć neuronowa ma dobre cechy modelu odwrotnego. Dzięki bieżącemu douczaniu model odwrotny będzie aktualizowany przy zmianie właściwości obiektu, dzięki czemu układ sterowania będzie działał poprawnie.

3.8.2. Neuronowy algorytm regulacji IMC

Struktura neuronowego układu regulacji Internal Model Control (IMC) została przedstawiona na rys. 3.34. Na początku zostanie rozpatrzony przypadek liniowy. Transmitancja procesu oznaczona jest przez $P(q^{-1})q^{-\tau}$, natomiast transmitancja modelu przez $M(q^{-1})q^{-\tau}$. Algorytm wymaga również modelu odwrotnego obiektu, jego transmitancja jest oznaczona przez $C(q^{-1})$. Oczywiście, odwrotność nie dotyczy członu opóźniającego. Dodatkowy filtr ma transmitancję $F(q^{-1})$. W przeciwieństwie do układu bezpośredniego sterowania z modelem odwrotnym, w układzie regulacji IMC sygnał wyjściowy y nie jest bezpośrednio doprowadzony do modelu odwrotnego, ale wyznacza się różnicę między rzeczywistym sygnałem wyjściowym a sygnałem obliczonym na podstawie modelu. Dodatkowe zakłócenie sprowadzone do wyjścia procesu oznaczone jest przez $v(k)$. Gdy model oraz model odwrotny są jest doskonałe, sygnał $v(k) = 0$.



Rys. 3.34. Neuronowy algorytm regulacji IMC

Klasyczny układ regulacji IMC (z liniowym modelem) można opisać równaniem

$$\begin{aligned} y(k) = & \frac{F(q^{-1})C(q^{-1})P(q^{-1})q^{-\tau}}{1+F(q^{-1})C(q^{-1})(P(q^{-1})-M(q^{-1}))q^{-\tau}} y^{\text{zad}}(k) \\ & + \frac{1-F(q^{-1})C(q^{-1})M(q^{-1})q^{-\tau}}{1+F(q^{-1})C(q^{-1})(P(q^{-1})-M(q^{-1}))q^{-\tau}} v(k) \end{aligned} \tag{3.97}$$

co można przekształcić do postaci

$$y(k) = v(k) + \frac{F(q^{-1})C(q^{-1})P(q^{-1})q^{-\tau}}{1+F(q^{-1})C(q^{-1})(P(q^{-1})-M(q^{-1}))q^{-\tau}} (y^{\text{zad}}(k) - v(k)) \tag{3.98}$$

Sygnał sterujący ma wartość

$$u(k) = \frac{F(q^{-1})C(q^{-1})}{1+F(q^{-1})C(q^{-1})(P(q^{-1})-M(q^{-1}))q^{-\tau}} (y^{\text{zad}}(k) - v(k)) \tag{3.99}$$

Warunkiem stabilności układu zamkniętego jest stabilność procesu, jego modelu i modelu odwrotnego. Gdy model jest doskonały, czyli gdy $M(q^{-1}) = P(q^{-1})$ i $C(q^{-1}) = P^{-1}(q^{-1})$

$$y(k) = q^{-\tau} y^{\text{zad}}(k) + (1 - F(q^{-1})q^{-\tau})v(k) \tag{3.100}$$

oraz

$$u(k) = \frac{F(q^{-1})}{P(q^{-1})}(y^{\text{zad}}(k) - v(k)) \tag{3.101}$$

Filtr $F(q^{-1})$ ma zasadniczy wpływ na nadążanie za zmianami wartości zadanej i tłumienie zakłóceń. Dlatego transmitancja $F(q^{-1})$ musi być stabilna i mieć wzmocnienie jednostkowe.

Struktura nieliniowego układu regulacji IMC [12, 27] jest taka sama jak w przypadku liniowym. Nieliniowość modelu pociąga za sobą potrzebę zastosowania, w miejsce liniowego modelu i liniowego modelu odwrotnego, modeli nieliniowych.

3.8.3. Układ regulacji z linearyzacją w pętli sprzężenia zwrotnego

Układ regulacji z linearyzacją w pętli sprzężenia zwrotnego (ang. feedback linearisation) w podstawowej wersji jest formułowany w dziedzinie czasu ciągłego [14]. Podstawowym założeniem jest szczególna postać modelu (postać kanoniczna)

$$\dot{x}(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{n_x}(t) \end{bmatrix} = \begin{bmatrix} x_2(t) \\ x_3(t) \\ \vdots \\ \tilde{f}(x(t)) + \tilde{g}(x(t))u(t) \end{bmatrix} \tag{3.102}$$

gdzie $\tilde{f}$ i $\tilde{g}$ są nieliniowymi funkcjami zmiennych stanu. Jeżeli model nie ma żądanej postaci, istnieją specjalne metody pozwalające na odpowiednie przekształcenie modelu.

System może być zlinearyzowany przez zastosowanie sygnału sterującego

$$u(t) = \frac{w(t) - \tilde{f}(x(t))}{\tilde{g}(x(t))} \tag{3.103}$$

przy czym $w(t)$ jest tzw. wirtualnym sterowaniem. Zakłada się również, że $\tilde{g}(x(t)) \neq 0$. Jeżeli wszystkie zmienne stanu są mierzone (lub obserwowane), można zaprojektować regulator ze sprzężeniem od stanu. System zamknięty można wówczas opisać równaniami

$$\begin{aligned} \dot{x}(t) &= \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \vdots \\ \dot{x}_{n_x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_0 & -a_1 & -a_2 & \dots & -a_{n_x-1} \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} y^{\text{zad}}(t) \\ y(t) &= x_1(t) \end{aligned} \tag{3.104}$$

czemu odpowiada transmitancja ciągła

$$G(s) = \frac{1}{s^{n_x} + a_{n_x-1}s^{n_x-1} + \dots + a_0} \tag{3.105}$$

gdzie $a_0,\ldots,a_{n_x-1}$ określają wielomian charakterystyczny (bieguny układu zamkniętego).

Układ regulacji z linearyzacją w pętli sprzężenia zwrotnego można również zaprojektować dla układów dyskretnych [27]. Zakłada się, że model dyskretny ma postać

$$\begin{aligned} y(k) = {} & f(u(k-2),\ldots,u(k-n_B), y(k-1),\ldots, y(k-n_A)) \\ & + g(u(k-2),\ldots,u(k-n_B), y(k-1),\ldots, y(k-n_A))u(k-1) \end{aligned} \tag{3.106}$$

czemu odpowiada dyskretny model w przestrzeni stanu

$$\begin{aligned} x(k+1) &= \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ \vdots \\ x_{n_x}(k+1) \end{bmatrix} = \begin{bmatrix} x_2(k) \\ x_3(k) \\ \vdots \\ f(\ldots) + g(\ldots)u(k) \end{bmatrix} \\ y(k) &= x_{n_x}(k) \end{aligned} \tag{3.107}$$

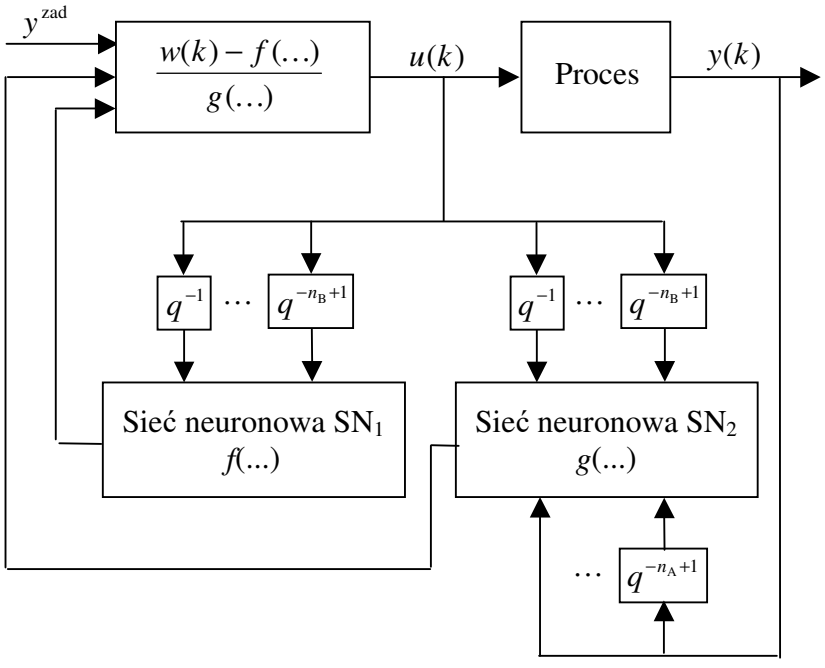
Wektor stanu jest następujący

$$x(k) = \begin{bmatrix} y(k-n_A+1) & \ldots & y(k-1) & y(k) \end{bmatrix}^T \tag{3.108}$$

Jeżeli funkcje *f* oraz *g* są znane, wprowadzając sterowanie

$$u(k) = \frac{w(k) - f(u(k-1),\ldots,u(k-n_B+1))}{g(u(k-1),\ldots,u(k-n_B+1), y(k),\ldots, y(k-n_A+1))} \tag{3.109}$$

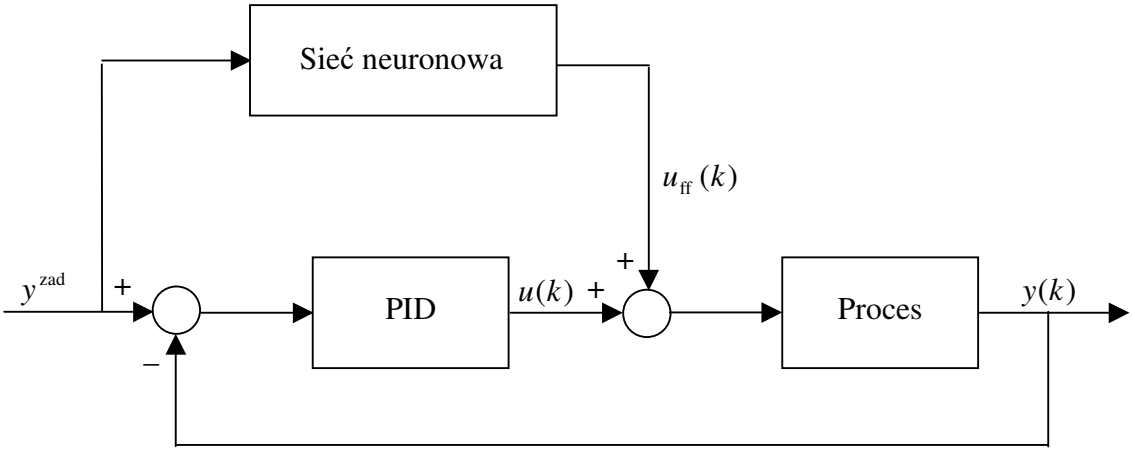
linearyzuje się system. Struktura otrzymanego układu regulacji została pokazana na rys. 3.35. Stosuje się dwie sieci neuronowe realizujące funkcje *f* oraz *g*. Zaletą opisywanego podejścia jest fakt, że w przeciwieństwie do układu sterowania bezpośredniego z modelem odwrotnym, sieci neuronowe mogą być nauczone w klasyczny sposób na podstawie zarejestrowanych wcześniej zbiorów danych. Niestety, wadą układu regulacji z linearyzacją w pętli sprzężenia zwrotnego są duże wartości generowanego sygnału sterującego.



Rys. 3.35. Układ regulacji z linearyzacją w pętli sprzężenia zwrotnego

3.8.4. Układ regulacji PID z siecią neuronową

W układach regulacji bardzo wielu procesów wykorzystuje się klasyczne regulatory PID. Dzięki swoim zaletom (łatwość strojenia, duża jakość regulacji, znaczna odporność przy zmianach parametrów obiektu) mają one ugruntowaną pozycję. W porównaniu z omówionymi do tej pory neuronowymi układami regulacji, w których sygnał sterujący ma zwykle niekorzystny przebieg (duże wartości, duża szybkość zmian), w układzie regulacji PID sygnał ten ma znacznie łagodniejszy przebieg. Regulatory PID wykorzystuje się również w procesach nieliniowych. Niestety, nieliniowość ta nie jest uwzględniona w algorytmie regulacji, co może prowadzić do regulacji gorszej niż oczekiwana. W celu poprawy pracy liniowego algorytmu PID można w prosty sposób zastosować dodatkowy człon nieliniowy realizowany przez sieć neuronową. Struktura układu regulacji PID z siecią neuronową przedstawiona została na rys. 3.36. Sieć neuronowa jest zastosowana w torze dodatkowego sprzężenia do przodu (ang. feedforward) od wartości zadanej.



Rys. 3.36. Układ regulacji PID z siecią neuronową

Omawiany układ regulacji występuje w dwóch wersjach. W pierwszej wersji wykorzystuje się model odwrotny, którego postać jest taka sama jak w układzie sterowania bezpośredniego z modelem odwrotnym, a mianowicie

$$u(k) = f^{-1}(u(k-1),\ldots,u(k-n_{\mathrm{B}}+1), y(k+1), y(k),\ldots, y(k-n_{\mathrm{A}}+1)) \tag{3.110}$$

W układzie regulacji PID należy podstawić $u = u_{\mathrm{ff}}$ oraz $y = y^{\mathrm{zad}}$, czyli otrzymuje się

$$u_{\mathrm{ff}}(k) = f^{-1}(u_{\mathrm{ff}}(k-1),\ldots,u_{\mathrm{ff}}(k-n_{\mathrm{B}}+1), y^{\mathrm{zad}}(k+1), y^{\mathrm{zad}}(k),\ldots, y^{\mathrm{zad}}(k-n_{\mathrm{A}}+1)) \tag{3.111}$$

Stosując model neuronowy skończonej odpowiedzi impulsowej (NNFIR), argumentami sieci neuronowej realizującej model odwrotny są wyłącznie wartości sygnału zadanego

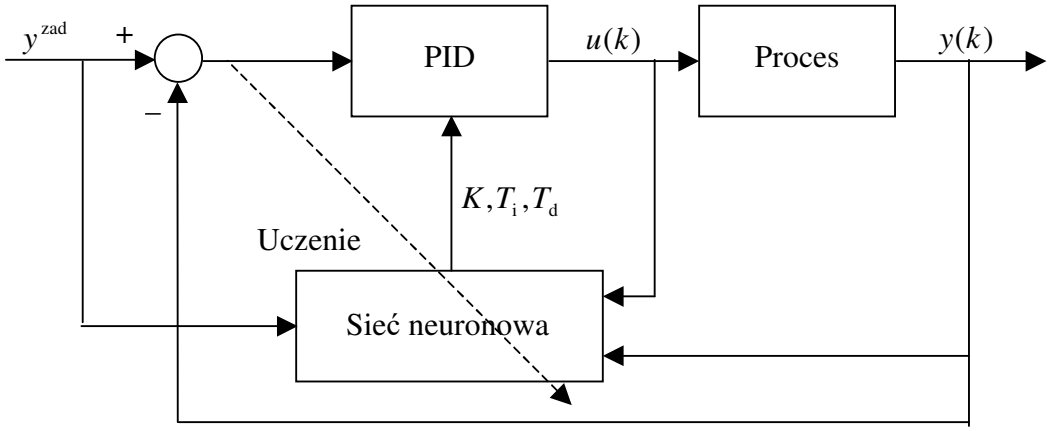
$$u_{\mathrm{ff}}(k) = f^{-1}(y^{\mathrm{zad}}(k+1), y^{\mathrm{zad}}(k),\ldots, y^{\mathrm{zad}}(k-n_{\mathrm{A}}+1)) \tag{3.112}$$

Oczywiście, modele typu NNFIR mogą nie być wystarczająco dokładne, ale w układzie regulacji znajduje się regulator PID, który ma właściwości całkujące, a więc kompensuje niedoskonałości modelu i zeruje uchyb.

Zaletą opisywanego podejścia jest łatwość implementacji. Warto zwrócić uwagę, że gdy zmiany sygnału wartości zadanej znane są w trakcie projektowania układu regulacji, to przebieg sygnału u_{ff} może być obliczony i zapamiętany w pamięci regulatora. Oczywistą wadą rozwiązania jest fakt, że wymagany jest model odwrotny.

3.8.5. Adaptacyjny układ regulacji PID z siecią neuronową

Strukturę adaptacyjnego algorytmu regulacji PID z siecią neuronową przedstawiono na rys. 3.37. W porównaniu z klasycznym algorytmem PID o stałych nastawach, w omawianym układzie regulacji nastawy regulatora PID są obliczane na bieżąco przez sieć neuronową w zależności od punktu pracy i wartości zadanej. Układ regulacji może występować w dwóch wersjach. W pierwszej, prostszej, sieć neuronowa jest uczona w trakcie projektowania układu. W drugiej wersji sieć neuronowa jest uczona na bieżąco w trakcie działania układu.



Rys. 3.37. Adaptacyjny układ regulacji PID z siecią neuronową

3.8.6. Algorytmy regulacji predykcyjnej z nieliniową optymalizacją

W każdej iteracji k algorytmu regulacji predykcyjnej, gdzie k=1, 2,..., na podstawie dynamicznego modelu procesu oraz pomiarów sygnałów procesowych zostaje wyznaczona przyszła sekwencja sygnałów sterujących. W zależności od konkretnej implementacji algorytmu może być obliczony wektor przyszłych wartości sygnału sterującego

$$\boldsymbol{u}(k) = \left[u(k \mid k) \quad u(k+1 \mid k) \quad u(k+2 \mid k) \quad \ldots \quad u(k+N_{\mathrm{u}}-1 \mid k)\right]^{\mathrm{T}} \tag{3.113}$$

lub też odpowiadający mu wektor przyszłych przyrostów sygnału sterującego

$$\Delta\boldsymbol{u}(k) = \left[\Delta u(k \mid k) \quad \Delta u(k+1 \mid k) \quad \Delta u(k+2 \mid k) \quad \ldots \quad \Delta u(k+N_{\mathrm{u}}-1 \mid k)\right]^{\mathrm{T}} \tag{3.114}$$

Zapis $u(k+p \mid k)$ oznacza sygnał sterujący dla chwili k+p obliczony w chwili k-tej (aktualnej iteracji algorytmu), $\Delta u(k+p \mid k)$ oznacza przyrost sygnału dla chwili w chwili k+p obliczony w chwili k. N_{u} jest horyzontem sterowania, długością wyznaczanego wektora. Związki między przyrostami a wartościami bezwzględnymi sygnału sterującego są następujące

$$\begin{aligned} \Delta u(k \mid k) &= u(k \mid k) - u(k-1) \\ \Delta u(k+1 \mid k) &= u(k+1 \mid k) - u(k \mid k) \\ &\vdots \\ \Delta u(k+N_{\mathrm{u}}-1 \mid k) &= u(k+N_{\mathrm{u}}-1 \mid k) - u(k+N_{\mathrm{u}}-2 \mid k) \end{aligned} \tag{3.115}$$

Przyjmuje się, że poza horyzontem sterowania sygnał sterujący jest stały

$$\Delta u(k+p \mid k) = 0 \quad \text{dla} \quad p \geq N_{\mathrm{u}} \tag{3.116}$$

Do sterowania procesu wykorzystuje się tylko pierwszy element wyznaczonego ciągu

$$u(k) = u(k \mid k) \quad \text{lub} \quad u(k) = \Delta u(k \mid k) + u(k-1) \tag{3.117}$$

W kolejnej iteracji (k+1), po aktualizacji pomiarów, cała procedura zostaje powtórzona.

Wektor zmiennych decyzyjnych algorytmu obliczany jest w wyniku minimalizacji pewnego wskaźnika jakości regulacji. Optymalizowana funkcja celu składa się zazwyczaj z dwóch członów. Pierwszy z nich, zdefiniowany na horyzoncie predykcji N, uwzględnia różnice między prognozowaną trajektorią wyjścia a trajektorią zadaną (jest to więc prognozowany uchyb regulacji). Drugi składnik funkcji kryterialnej (człon kary) ma za zadanie minimalizować duże przyrosty przyszłego sygnału sterującego. Najczęściej wykorzystuję się kwadratową funkcję celu

$$J(k) = \sum_{p=1}^{N} (y^{\text{zad}}(k) - \hat{y}(k+p \mid k))^2 + \lambda \sum_{p=0}^{N_u} (\Delta u(k+p \mid k))^2 \tag{3.118}$$

gdzie nieujemny współczynnik λ decyduje o szybkości algorytmu (zwiększanie tego parametru prowadzi do ograniczenia przyrostów sygnału sterującego i spowolnienia algorytmu). Aktualna wartość sygnału wartości zadanej oznaczona jest symbolem $y^{\text{zad}}(k)$, $\hat{y}(k+p \mid k)$ oznacza prognozowany sygnał wyjścia dla chwili k+p obliczony w aktualnej iteracji algorytmu.

Wielką zaletą algorytmów regulacji predykcyjnej jest możliwość uwzględnienia ograniczeń wartości sygnału sterującego i sygnału wyjściowego oraz ograniczeń szybkości narastania sygnału sterującego. Zadanie optymalizacji, w wyniku rozwiązania którego oblicza się przyszłą sekwencję sterującą jest następujące (dla zwięzłości zapisu ograniczenia wyjścia są twarde)

$$\begin{aligned}
&\min_{\boldsymbol{u}(k)} \{J(k)\} \\
&\text{przy ograniczeniach} \\
&\quad u^{\min} \le u(k+p \mid k) \le u^{\max} \qquad \text{dla} \quad p = 0,\ldots,N_u - 1 \\
&-\Delta u^{\max} \le \Delta u(k+p \mid k) \le \Delta u^{\max} \qquad \text{dla} \quad p = 0,\ldots,N_u - 1 \\
&\quad y^{\min} \le \hat{y}(k+p \mid k) \le y^{\max} \qquad \text{dla} \quad p = 1,\ldots,N
\end{aligned} \tag{3.119}$$

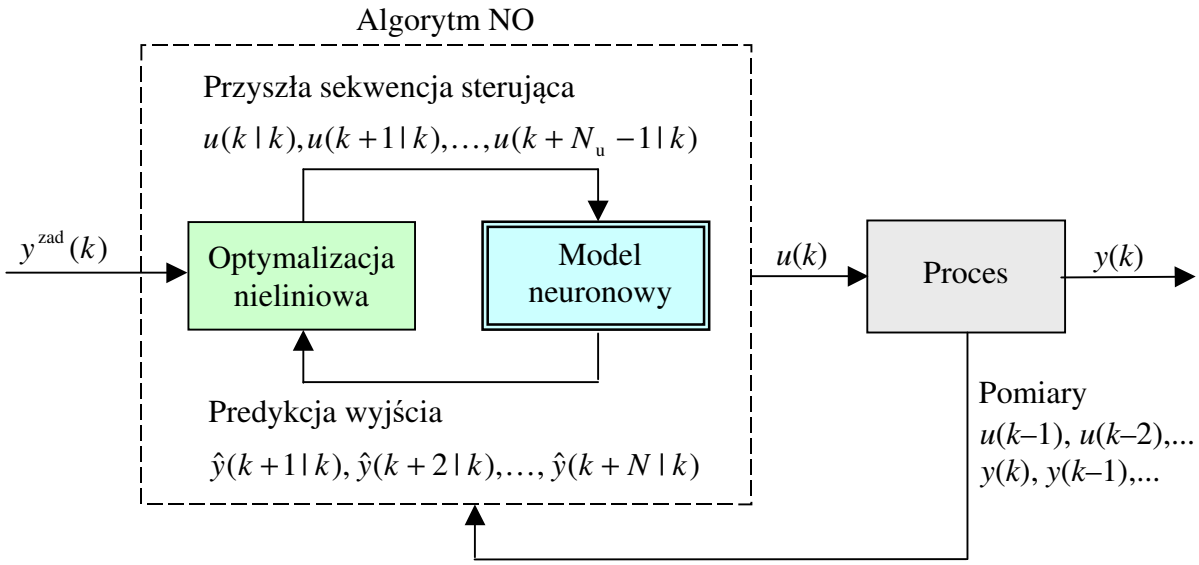
Warto podkreślić, że podane powyżej sformułowanie algorytmu regulacji predykcyjnej jest bardzo uniwersalne, nie ogranicza liczby wejść i wyjść procesu ani klasy modelu stosowanego do predykcji. Z tego też powodu, oraz dzięki możliwości uwzględnienia ograniczeń, uważa się, że algorytmy omawianej klasy są jedyną zaawansowaną techniką regulacji, która odniosła istotny sukces praktyczny [38].

Jeżeli do predykcji stosuje się model liniowy, otrzymany problem optymalizacji jest zadaniem programowania kwadratowego (kwadratowa funkcja celu, liniowe ograniczenia). Problemy tego typu, przy wykorzystaniu metod zbioru ograniczeń aktywnych lub punktu wewnętrznego, rozwiązywane są w bardzo efektywny sposób. Można określić czas obliczeń.

Oczywiste jest, że procesy nieliniowe nie mogą być w sposób dokładny opisane za pomocą modeli liniowych. W konsekwencji, w przypadku istotnie nieliniowych procesów zastosowanie algorytmu liniowego wykorzystującego do predykcji model liniowy może okazać się niewystarczające (w zależności od punktu pracy algorytm taki może pracować niestabilnie lub też bardzo wolno). Do regulacji silnie nieliniowych procesów należy zastosować algorytmy wykorzystujące modele nieliniowe. W ogólności, można wyróżnić dwie klasy nieliniowych algorytmów regulacji predykcyjnej:

a) algorytmy z nieliniową optymalizacją,

b) algorytmy z cykliczną linearyzacją modelu nieliniowego.



Rys. 3.38. Struktura układu regulacji z algorytmem z Nieliniową Optymalizacją (NO)

Najbardziej naturalnie koncepcyjnie rozwiązanie to zastosowanie nieliniowego modelu procesu do predykcji w algorytmie regulacji predykcyjnej. Algorytm ten nazywany będzie algorytmem typu NO (z Nieliniową Optymalizacja), jego struktura przedstawiona jest na rys. 3.38. Niech dynamiczny model procesu dany będzie przez następujące równanie różnicowe

$$y(k) = f(u(k-\tau), \ldots, u(k-n_B), y(k-1), \ldots, y(k-n_A)) \tag{3.120}$$

przy czym funkcja $f : \Re^{n_A+n_B-\tau+1} \to \Re$ realizowana jest przez sieć neuronową, np. typu MLP lub RBF. Dla sieci MLP równanie modelu ma postać

$$y(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i(k)) \tag{3.121}$$

gdzie suma sygnałów dochodzących do i-tego neuronu ukrytego jest następująca

$$z_i(k) = w_{1,0}^1 + \sum_{j=1}^{I_u} w_{i,j}^1 u(k-\tau+1-j) + \sum_{j=1}^{n_A} w_{i,I_u+j}^1 y(k-j) \tag{3.122}$$

gdzie $I_u = n_B - \tau + 1$. Wagi pierwszej warstwy oznaczone są przez $w_{i,j}^1$, gdzie $i = 1, \ldots, K$, $j = 0, \ldots, I_u + n_A$, natomiast wagi drugiej warstwy przez w_i^2, gdzie $i = 0, \ldots, K$.

Ogólne równanie predykcji dla chwili $k+p$ ($p = 1, \ldots, N$) ma postać

$$\hat{y}(k+p \mid k) = y(k+p \mid k) + d(k) \tag{3.123}$$

gdzie sygnał $y(k+p \mid k)$ jest wyjściem modelu dla chwili $k+p$ obliczonym na podstawie dostępnych pomiarów, $d(k)$ jest estymacją niemierzalnego zakłócenia. Przyjmuje się, że jest ono stałe na horyzoncie predykcji, a jego wartość jest obliczana jako różnica między zmierzonym sygnałem wyjściowym a sygnałem obliczonym na podstawie modelu przy użyciu sygnałów do chwili $k-1$ włącznie

$$d(k) = y(k) - y(k \mid k-1) \tag{3.124}$$

Uwzględnienie zakłócenia wprowadza do algorytmu działanie całkujące, co pozwala skompensować błędy modelowania i rzeczywiste zakłócenia.

Wykorzystując równanie predykcji (3.123) oraz model w postaci ogólnej, kolejne predykcje na horyzoncie predykcji ($p = 1,\ldots,N$) można wyrazić w następującej formie

$$\begin{aligned}\hat{y}(k+p\mid k) = f(&\underbrace{u(k-\tau+p\mid k),\ldots,u(k\mid k)}_{I_{\mathrm{uf}}(p)},\underbrace{u(k-1),\ldots,u(k-n_{\mathrm{B}}+p)}_{I_{\mathrm{u}}-I_{\mathrm{uf}}(p)},\\ &\underbrace{\hat{y}(k-1+p\mid k),\ldots,\hat{y}(k+1\mid k)}_{I_{\mathrm{yp}}(p)},\underbrace{y(k),\ldots,y(k-n_{\mathrm{A}}+p)}_{n_{\mathrm{A}}-I_{\mathrm{yp}}(p)})+d(k)\end{aligned} \tag{3.125}$$

Predykcja $\hat{y}(k+p\mid k)$ zależy od $I_{\mathrm{uf}}(p)=\max(\min(p-\tau+1,I_{\mathrm{u}}),0)$ przyszłych sygnałów sterujących (zmiennych decyzyjnych algorytmu regulacji), $I_{\mathrm{u}}-I_{\mathrm{uf}}(p)$ wartości sygnału sterującego, które zostały wykorzystane do sterowania w poprzednich iteracjach, $I_{\mathrm{yp}}(p)=\min(p-1,n_{\mathrm{A}})$ predykcji oraz $n_{\mathrm{A}}-I_{\mathrm{yp}}(p)$ wartości sygnału sterującego zmierzonych w poprzednich iteracjach. Na przykład, dla modelu rzędu drugiego postaci

$$y(k)=f(u(k-1),u(k-2),y(k-1),y(k-2)) \tag{3.126}$$

predykcja dla chwili *k*+1 (*p*=1) ma postać

$$\hat{y}(k+1\mid k)=f(u(k\mid k),u(k-1),y(k),y(k-1))+d(k) \tag{3.127}$$

gdzie estymacja zakłócenia ma wartość

$$\begin{aligned}d(k) &= y(k)-y(k\mid k-1)\\ &= y(k)-f(u(k-1),u(k-2),y(k-1),y(k-2))\end{aligned} \tag{3.128}$$

Predykcja dla chwili *k*+2 (*p*=2) ma postać

$$\hat{y}(k+2\mid k)=f(u(k+1\mid k),u(k\mid k),\hat{y}(k+1\mid k),y(k))+d(k) \tag{3.129}$$

Predykcja dla chwili *k*+3 (*p*=3) ma postać

$$\hat{y}(k+3\mid k)=f(u(k+2\mid k),u(k+1\mid k),\hat{y}(k+2\mid k),\hat{y}(k+1\mid k))+d(k) \tag{3.130}$$

Ogólnie, dla *p*=3,...,*N* obowiązuje

$$\hat{y}(k+p\mid k)=f(u(k-1+p\mid k),u(k-2+p\mid k),\hat{y}(k-1+p\mid k),\hat{y}(k-2+p\mid k))+d(k) \tag{3.131}$$

Uwzględniając strukturę sieci MLP, otrzymuje się

$$\hat{y}(k+p\mid k)=y(k+p\mid k)+d(k)=w_0^2+\sum_{i=1}^{K}w_i^2\varphi(z_i(k+p\mid k))+d(k) \tag{3.132}$$

gdzie

$$\begin{aligned}z_i(k+p\mid k)=w_{1,0}^1&+\sum_{j=1}^{I_{\mathrm{uf}}(p)}w_{i,j}^1u(k-\tau+1-j+p\mid k)+\sum_{j=I_{\mathrm{uf}}(p)+1}^{I_u}w_{i,j}^1u(k-\tau+1-j+p)\\ &+\sum_{j=1}^{I_{\mathrm{yp}}(p)}w_{i,I_{\mathrm{u}}+j}^1\hat{y}(k-j+p\mid k)+\sum_{j=I_{\mathrm{yp}}(p)+1}^{n_{\mathrm{A}}}w_{i,I_{\mathrm{u}}+j}^1y(k-j+p)\end{aligned} \tag{3.133}$$

natomiast ze wzoru (3.124) prognoza niemierzalnego zakłócenia ma wartość

$$d(k)=y(k)-\left(w_2(0)+\sum_{i=1}^{K}w_2(i)\varphi(z_i(k))\right) \tag{3.134}$$

Reasumując, algorytm regulacji predykcyjnej z nieliniową optymalizacją polega na zastosowaniu do predykcji pełnego, nieliniowego modelu procesu (w rozważanym przypadku jest to model neuronowy). Procedura optymalizacji tak dobiera wektor zmiennych decyzyjnych algorytmu (wektor przyszłych sygnałów sterujących $\boldsymbol{u}(k)$), aby rozwiązać zadanie optymalizacji (3.119). Z punktu widzenia złożoności obliczeniowej i zbieżności algorytmu NO, fundamentalne znaczenie ma wybór punktu początkowego $\boldsymbol{u}^0(k)$ zadania nieliniowej optymalizacji. W najprostszym przypadku można założyć, że punkt początkowy jest stały. Nie jest to rozwiązanie dobre, ponieważ oczywiste jest, że punkt początkowy powinien być w jakiś sposób zależny od aktualnego punktu pracy. Znacznie lepsze jest przyjęcie, że punkt początkowy wyznaczony jest przez wartość sygnału sterującego wykorzystanego w poprzedniej iteracji

$$\boldsymbol{u}^0(k) = \begin{bmatrix} u(k-1) & u(k-1) & u(k-1) & \dots & u(k-1) \end{bmatrix}^{\mathrm{T}} \tag{3.135}$$

Logicznym podejściem, możliwym gdy tylko horyzont sterowania $N_{\mathrm{u}} > 1$, jest przyjęcie $N_{\mathrm{u}} - 1$ wartości sygnału sterującego obliczonych w poprzedniej iteracji algorytmu, które nie zostały wykorzystane do sterowania

$$\boldsymbol{u}^0(k) = \begin{bmatrix} u(k \mid k-1) \\ \vdots \\ u(k+N_{\mathrm{u}}-3 \mid k-1) \\ u(k+N_{\mathrm{u}}-2 \mid k-1) \\ u(k+N_{\mathrm{u}}-2 \mid k-1) \end{bmatrix} \tag{3.136}$$

Pomimo prostoty koncepcji algorytmu NO, ma on dwie zasadnicze wady strukturalne, ponieważ trajektoria prognozowana jest nieliniową funkcją wektora zmiennych decyzyjnych. W rezultacie, problem optymalizacji jest nieliniowy, często niewypukły. Po pierwsze, pomimo ciągłego doskonalenia nieliniowych algorytmów optymalizacji nie istnieje algorytm, który byłby w stanie wyznaczyć minimum globalne nieliniowego zadania optymalizacji w możliwym do oszacowania czasie. Po drugie, złożoność obliczeniowa algorytmów nieliniowej optymalizacji jest zazwyczaj znaczna, co stawia pod znakiem zapytania możliwość praktycznej implementacji algorytmu NO.

3.8.7. Algorytmy regulacji predykcyjnej z linearyzacją

Uwzględniając wady algorytmu NO, naturalnym rozwiązaniem wydaje się opracowanie nieliniowego algorytmu regulacji predykcyjnej, bazującego na nieliniowym modelu procesu, ale o znacznie mniejszej złożoności obliczeniowej. Bardzo duże znaczenie praktyczne mają algorytmy z linearyzacją, w których cyklicznie dokonuje się linearyzacji oryginalnego nieliniowego modelu w aktualnym punkcie pracy. Uzyskana liniowa aproksymacja modelu zostaje następnie wykorzystana do predykcji, dzięki czemu rozwiązywane w każdej iteracji zadanie optymalizacji można sformułować jako zadanie optymalizacji kwadratowej. W dalszej części pracy zostanie omówiony algorytm z Nieliniową Predykcją i Linearyzacją (NPL).

Liniowa aproksymacja nieliniowego modelu (3.120) ma postać

$$y(k) = \sum_{l=1}^{n_{\mathrm{B}}} b_l(k) u(k-l) - \sum_{l=1}^{n_{\mathrm{A}}} a_l(k) y(k-l) \tag{3.137}$$

gdzie $a_l(k)$ oraz $b_l(k)$ są zależnymi od punktu pracy współczynnikami modelu

zlinearyzowanego. Stosując w sposób rekurencyjny model zlinearyzowany, z ogólnego równania predykcji (3.123) można obliczyć prognozowany sygnał wyjściowy modelu na horyzoncie predykcji

$$\begin{aligned}
\hat{y}(k+1\,|\,k) &= b_1(k)u(k\,|\,k) + b_2(k)u(k-1) + b_3(k)u(k-2) + \ldots + b_{n_\mathrm{B}}(k)u(k-n_\mathrm{B}+1) \\
&\quad -a_1(k)y(k) - a_2(k)y(k-1) - a_3(k)y(k-2) - \ldots - a_{n_\mathrm{A}}(k)y(k-n_\mathrm{A}+1) \\
&\quad +d(k) \\
\hat{y}(k+2\,|\,k) &= b_1(k)u(k+1\,|\,k) + b_2(k)u(k\,|\,k) + b_3(k)u(k-1) + \ldots + b_{n_\mathrm{B}}(k)u(k-n_\mathrm{B}+2) \\
&\quad -a_1(k)\hat{y}(k+1\,|\,k) - a_2(k)y(k) - a_3(k)y(k-1) - \ldots - a_{n_\mathrm{A}}(k)y(k-n_\mathrm{A}+2) \\
&\quad +d(k) \\
\hat{y}(k+3\,|\,k) &= b_1(k)u(k+2\,|\,k) + b_2(k)u(k+1\,|\,k) + b_3(k)u(k\,|\,k) + \ldots + b_{n_\mathrm{B}}(k)u(k-n_\mathrm{B}+3) \\
&\quad -a_1(k)\hat{y}(k+2\,|\,k) - a_2(k)\hat{y}(k+1\,|\,k) - a_3(k)y(k) - \ldots - a_{n_\mathrm{A}}(k)y(k-n_\mathrm{A}+3) \\
&\quad +d(k) \\
&\quad \vdots
\end{aligned} \tag{3.138}$$

Predykcje można w wyrazić jako funkcje przyrostów sygnału sterującego

$$\begin{aligned}
\hat{y}(k+1\,|\,k) &= s_1(k)\Delta u(k\,|\,k) + \ldots \\
\hat{y}(k+2\,|\,k) &= s_2(k)\Delta u(k\,|\,k) + s_1(k)\Delta u(k+1\,|\,k) + \ldots \\
\hat{y}(k+3\,|\,k) &= s_3(k)\Delta u(k\,|\,k) + s_2(k)\Delta u(k+1\,|\,k) + s_1(k)\Delta u(k+2\,|\,k) + \ldots \\
&\quad \vdots
\end{aligned} \tag{3.139}$$

Współczynniki odpowiedzi skokowej modelu zlinearyzowanego są wyznaczane ze wzoru

$$s_j(k) = \sum_{i=1}^{\min(j,n_\mathrm{B})} b_i(k) - \sum_{i=1}^{\min(j-1,n_\mathrm{A})} a_i(k)s_{j-i}(k) \tag{3.140}$$

Wykorzystując do predykcji liniową aproksymację modelu nieliniowego można zauważyć, że wektor sygnałów prognozowanych na horyzoncie predykcji

$$\hat{\boldsymbol{y}}(k) = \begin{bmatrix} \hat{y}(k+1\,|\,k) & \ldots & \hat{y}(k+N\,|\,k) \end{bmatrix}^\mathrm{T} \tag{3.141}$$

można przedstawić jako następującą sumę

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{M}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \tag{3.142}$$

Pierwszy składnik powyższej sumy zależy wyłącznie od przyszłości (tzn. od przyszłych przyrostów sygnału sterującego $\Delta\boldsymbol{u}(k)$), natomiast drugi składnik sumy

$$\boldsymbol{y}^0(k) = \begin{bmatrix} y^0(k+1\,|\,k) & \ldots & y^0(k+N\,|\,k) \end{bmatrix}^\mathrm{T} \tag{3.143}$$

jest tzw. trajektorią swobodną, która zależy wyłącznie od przeszłości. Macierz dynamiczna

$$\boldsymbol{M}(k) = \begin{bmatrix} s_1(k) & 0 & \ldots & 0 \\ s_2(k) & s_1(k) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_N(k) & s_{N-1}(k) & \ldots & s_{N-N_\mathrm{u}+1}(k) \end{bmatrix} \tag{3.144}$$

o wymiarowości $N \times N_\mathrm{u}$ zawiera współczynniki odpowiedzi swobodnej modelu

zlinearyzowanego w aktualnym punkcie pracy. Zarówno macierz dynamiczna, jak i trajektoria swobodna są obliczane przy wykorzystaniu nieliniowego (neuronowego) modelu procesu.

Można oczywiście zauważyć, że predykcje obliczone na podstawie równania predykcji (3.142) nie będą dokładnie równe predykcjom obliczonym na podstawie pełnego modelu nieliniowego, jak ma to miejsce w algorytmie NO (wzory (3.132) i (3.133)). Jest o prawda, stosowana w algorytmie NPL predykcja nosi nazwę *predykcji suboptymalnej*. Intuicyjnie, można spodziewać się, że różnica między predykcją nieliniową a suboptymalną będzie proporcjonalna do aktualnie wyznaczanego wektora zmiennych decyzyjnych $\Delta \boldsymbol{u}(k)$, czyli do zmiany punktu pracy. Z drugiej jednak strony, stosując suboptymalne równanie predykcji otrzymuje się następujące zadanie optymalizacji kwadratowej

$$\begin{aligned}
&\min_{\Delta \boldsymbol{u}(k)}\{J(k) = \left\| \boldsymbol{y}^{\text{zad}}(k) - \boldsymbol{M}(k)\Delta \boldsymbol{u}(k) - \boldsymbol{y}^{0}(k) \right\|^2 + \left\| \Delta \boldsymbol{u}(k) \right\|^2_{\boldsymbol{\Lambda}}\} \\
&\text{przy ograniczeniach :} \\
&\quad \boldsymbol{u}^{\min} \le \boldsymbol{J}\Delta \boldsymbol{u}(k) + \boldsymbol{u}(k-1) \le \boldsymbol{u}^{\max} \\
&\quad -\Delta \boldsymbol{u}^{\max} \le \Delta \boldsymbol{u}(k) \le \Delta \boldsymbol{u}^{\max} \\
&\quad \boldsymbol{y}^{\min} \le \boldsymbol{M}(k)\Delta \boldsymbol{u}(k) + \boldsymbol{y}^{0}(k) \le \boldsymbol{y}^{\max}
\end{aligned} \tag{3.145}$$

gdzie wektor wartości zadanej

$$\boldsymbol{y}^{\text{zad}}(k) = \left[y^{\text{zad}}(k) \quad \ldots \quad y^{\text{zad}}(k) \right]^{\text{T}} \tag{3.146}$$

oraz wektory definiujące ograniczenia sygnału wyjściowego

$$\boldsymbol{y}^{\min} = \left[y^{\min} \quad \ldots \quad y^{\min} \right]^{\text{T}}, \quad \boldsymbol{y}^{\max} = \left[y^{\max} \quad \ldots \quad y^{\max} \right]^{\text{T}} \tag{3.147}$$

mają długość *N*, wektory związane z sygnałem sterującym

$$\begin{aligned}
\boldsymbol{u}^{\min} &= \left[u^{\min} \quad \ldots \quad u^{\min} \right]^{\text{T}}, \quad \boldsymbol{u}(k-1) = \left[u(k-1) \quad \ldots \quad u(k-1) \right]^{\text{T}} \\
\boldsymbol{u}^{\max} &= \left[u^{\max} \quad \ldots \quad u^{\max} \right]^{\text{T}}, \quad \Delta \boldsymbol{u}^{\max} = \left[\Delta u^{\max} \quad \ldots \quad \Delta u^{\max} \right]^{\text{T}}
\end{aligned} \tag{3.148}$$

mają długość N_{u}, natomiast macierze o wymiarowości $N_{\text{u}} \times N_{\text{u}}$ mają postać

$$\boldsymbol{J} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \ldots & 1 \end{bmatrix}, \quad \boldsymbol{\Lambda} = \lambda \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{bmatrix} \tag{3.149}$$

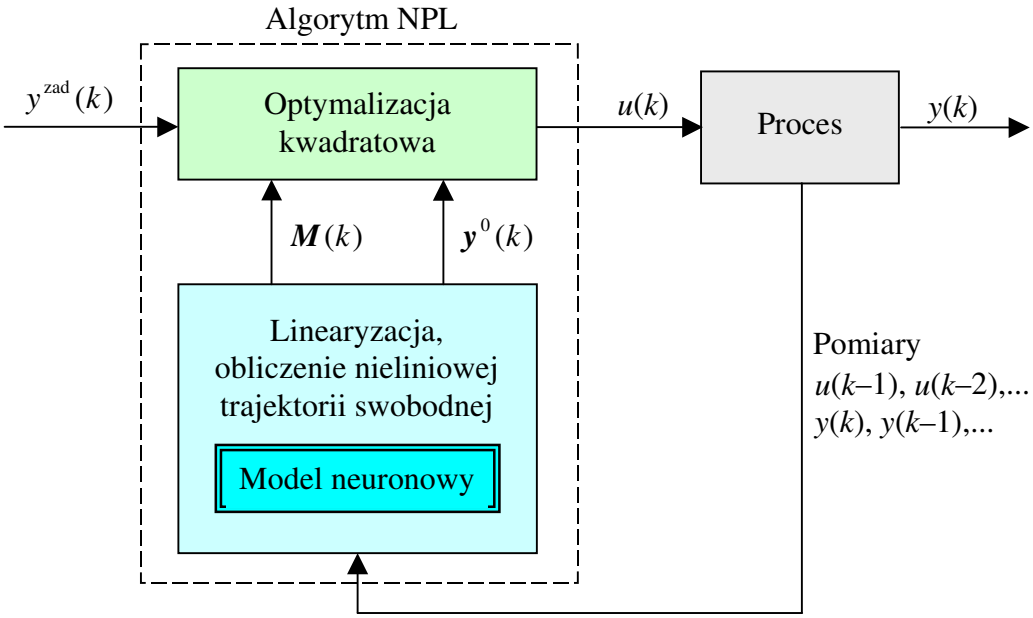
W zadaniu optymalizacji algorytmu NPL (3.145) zastosowano tzw. twarde ograniczenia wyjścia. Ograniczenia takie mogą okazać się w praktyce niemożliwe do spełnienia, np. po wystąpieniu zakłócenia. Lepiej zastosować tzw. *miękkie ograniczenia wyjścia*, które, w razie potrzeby, mogą być czasowo przekraczane. Zadanie optymalizacji algorytmu NPL ma postać

$$\min_{\Delta \boldsymbol{u}(k),\boldsymbol{\varepsilon}^{\min},\boldsymbol{\varepsilon}^{\max}} \{J(k) = \left\| \boldsymbol{y}^{\text{zad}}(k) - \boldsymbol{M}(k)\Delta \boldsymbol{u}(k) - \boldsymbol{y}^{0}(k) \right\|^2 + \left\| \Delta \boldsymbol{u}(k) \right\|_{\boldsymbol{\Lambda}}^2 + \rho^{\min} \left\| \boldsymbol{\varepsilon}^{\min} \right\|^2 + \rho^{\max} \left\| \boldsymbol{\varepsilon}^{\max} \right\|^2 \}$$

przy ograniczeniach : (3.150)

$$\boldsymbol{u}^{\min} \le \boldsymbol{J}\Delta \boldsymbol{u}(k) + \boldsymbol{u}(k-1) \le \boldsymbol{u}^{\max}$$
$$-\Delta \boldsymbol{u}^{\max} \le \Delta \boldsymbol{u}(k) \le \Delta \boldsymbol{u}^{\max}$$
$$\boldsymbol{y}^{\min} - \boldsymbol{\varepsilon}^{\min} \le \boldsymbol{M}(k)\Delta \boldsymbol{u}(k) + \boldsymbol{y}^{0}(k) \le \boldsymbol{y}^{\max} + \boldsymbol{\varepsilon}^{\max}$$
$$\boldsymbol{\varepsilon}^{\min} \ge 0, \quad \boldsymbol{\varepsilon}^{\max} \ge 0$$

Dodatkowe zmienne optymalizacji to wektory $\boldsymbol{\varepsilon}^{\min}$ oraz $\boldsymbol{\varepsilon}^{\max}$ o długości *N*. W funkcji kryterialnej stosuje się dodatkowo dwa człony kary, których celem jest minimalizacja stopnia przekroczenia ograniczeń sygnału wyjściowego, $\rho^{\min}$, $\rho^{\max}$ są dodatnimi wagami.



Rys. 3.39. Struktura układu regulacji z algorytmem z Nieliniową Predykcją i Linearyzacją (NPL)

Struktura algorytmu NPL została szczegółowo przedstawiona na rys. 3.39. W każdej iteracji algorytmu (dyskretnej chwili *k*) zostają powtórzone następujące czynności:

1. Na podstawie nieliniowego (neuronowego) modelu procesu wyznacza się jego aproksymację liniową w aktualnym punkcie pracy (3.137). Model zlinearyzowany służy do obliczenia współczynników odpowiedzi skokowej $s_j(k)$ ze wzoru (3.140), które określają z kolei macierz dynamiczną $\boldsymbol{M}(k)$ (3.144).
2. Nieliniowy model procesu zostaje także wykorzystany do wyznaczenia nieliniowej trajektorii swobodnej $\boldsymbol{y}^{0}(k)$.
3. Rozwiązane zostaje zadanie programowania kwadratowego (3.150), w wyniku czego oblicza się wektor przyszłych przyrostów sygnałów sterujących $\Delta \boldsymbol{u}(k)$.
4. Pierwszy element wyznaczonej sekwencji zostaje wykorzystany do bieżącego sterowania, tzn. $u(k) = \Delta u(k \mid k) + u(k-1)$.
5. W następnej iteracji ($k := k+1$) następuje przejście do pierwszego kroku algorytmu.

Jeżeli stopień nieliniowości procesu nie jest znaczny, lub gdy zmiany punktu pracy nie są szybkie i częste, linearyzacja może być wykonywana nie w każdej linearyzacji, ale rzadziej.

Fundamentalną zaletą algorytmu NPL jest fakt, że do bieżącej optymalizacji stosuje się procedurę programowania kwadratowego. Oznacza to, że unika się nieliniowej optymalizacji stosowanej w algorytmie NO. Rozwiązanie takie ma dwie zasadnicze zalety: jakościową i ilościową. Po pierwsze, znalezienie globalnego rozwiązania kwadratowego zadania optymalizacji jest możliwe w każdej iteracji, nie ma ryzyka wyznaczenia minimum lokalnego. Po drugiej, nakład obliczeń algorytmu programowania kwadratowego jest zwykle bez porównania mniejszy niż algorytmu nieliniowej optymalizacji z ograniczeniami. Oczywiście należy pamiętać, że w algorytmie NPL stosuje się do predykcji aproksymację liniową oryginalnego modelu nieliniowego. Można się więc spodziewać różnic między działaniem algorytmu NO i NPL. Jak wykazują dotychczasowe badania, różnice jakości regulacji nie są jednak duże, natomiast różnica złożoności obliczeniowej jest kolosalna.

Sformułowany powyżej algorytm NPL jest ogólny, nie ogranicza klasy zastosowanych modeli dynamicznych. Z uwagi na prostą, regularną strukturę jednokierunkowych perceptronowych sieci neuronowych (MLP) oraz uwzględniając, że pozwalają one modelować silnie nieliniowe procesy z dużą dokładnością przy umiarkowanej liczbie parametrów (wag), warto podać szczegóły implementacji algorytmu dla tych najpopularniejszych modeli neuronowych.

Uwzględniając model neuronowy określony wzorami (3.121) i (3.122), współczynniki modelu zlinearyzowanego (3.137) są obliczane analitycznie jako

$$a_l(k) = -\frac{\partial f(\bar{\boldsymbol{x}}(k))}{\partial y(k-l)} = -\sum_{i=1}^{K} w_i^2 \frac{\partial \varphi(z_i(\bar{\boldsymbol{x}}(k)))}{\partial z_i(\bar{\boldsymbol{x}}(k))} w_{i,l-\tau+1}^1 \tag{3.151}$$

dla wszystkich $l = 1,\ldots,n_{\mathrm{A}}$, oraz

$$b_l(k) = \frac{\partial f(\bar{\boldsymbol{x}}(k))}{\partial u(k-l)} = \begin{cases} 0 & l = 1,\ldots,\tau-1 \\ \sum_{i=1}^{K} w_i^2 \frac{\partial \varphi(z_i(\bar{\boldsymbol{x}}(k)))}{\partial z_i(\bar{\boldsymbol{x}}(k))} w_{i,l-\tau+1}^1 & l = \tau,\ldots,n_{\mathrm{B}} \end{cases} \tag{3.152}$$

Punkt linearyzacji określony jest przez sygnały sterujące wykorzystane w poprzednich iteracjach oraz sygnały wyjściowe zmierzone w przeszłości. Sygnały te odpowiadają argumentom modelu nieliniowego (3.120)

$$\bar{\boldsymbol{x}}(k) = \left[u(k-\tau) \quad \ldots \quad u(k-n_{\mathrm{B}}) \quad y(k-1) \quad \ldots \quad y(k-n_{\mathrm{A}})\right]^{\mathrm{T}} \tag{3.153}$$

Jeżeli neurony warstwy ukrytej mają funkcję aktywacji φ typu tangens hiperboliczny zachodzi

$$\frac{\partial \varphi(z_i(\bar{\boldsymbol{x}}(k)))}{\partial z_i(\bar{\boldsymbol{x}}(k))} = 1 - \mathrm{tgh}^2(z_i(\bar{\boldsymbol{x}}(k))) \tag{3.154}$$

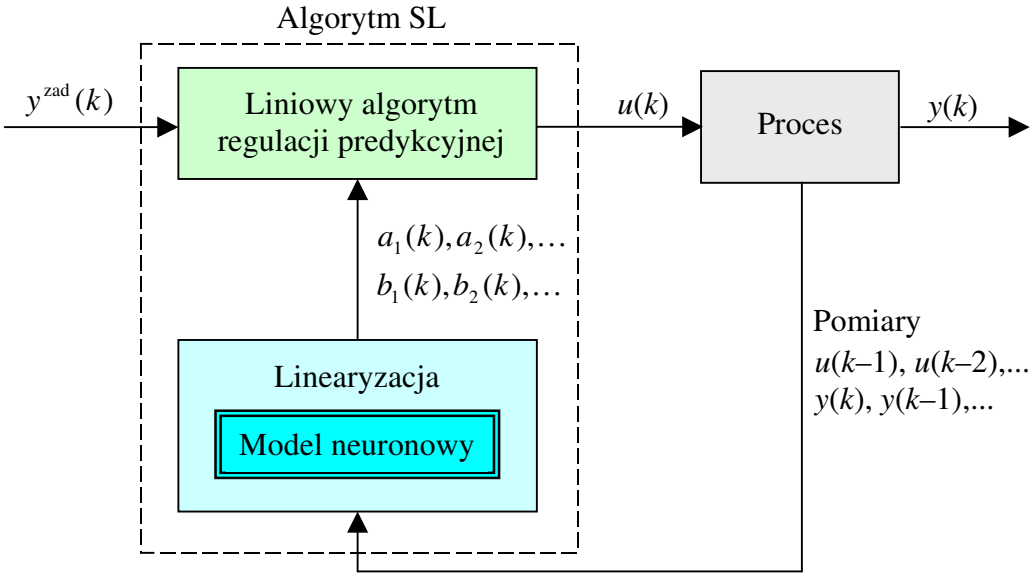
Nieliniowa trajektoria swobodna obliczana jest rekurencyjnie (dla $p = 1,\ldots,N$) na podstawie nieliniowego równania predykcji (stosowanego w algorytmie NO), ale przy uwzględnieniu wyłącznie wpływu przeszłości

$$y^0(k+p \mid k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(z_i^0(k+p \mid k)) + d(k) \tag{3.155}$$

Korzystając ze wzoru (3.133) i uwzględniając jedynie wpływ przeszłości otrzymuje się

$$z_i^0(k+p\,|\,k) = w_{1,0}^1 + \sum_{j=1}^{I_{\mathrm{uf}}(p)} w_{i,j}^1 u(k-1) + \sum_{j=I_{\mathrm{uf}}(p)+1}^{I_u} w_{i,j}^1 u(k-\tau+1-j+p) + \sum_{j=1}^{I_{\mathrm{yp}}(p)} w_{i,I_{\mathrm{u}}+j}^1 y^0(k-j+p\,|\,k) + \sum_{j=I_{\mathrm{yp}}(p)+1}^{n_{\mathrm{A}}} w_{i,I_{\mathrm{u}}+j}^1 y(k-j+p) \tag{3.156}$$

W porównaniu ze wzorem (3.133) na potrzeby obliczania trajektorii swobodnej przyjmuje się, że przyrosty sterowania od chwili k są zerowe, czyli $u(k\,|\,k) = u(k+1\,|\,k) = \ldots = u(k-1)$. Przedostatnia suma powyższego wzoru nie zawiera również nieliniowych predykcji $\hat{y}(k+1\,|\,k), \hat{y}(k+2\,|\,k), \ldots$, zostają one zastąpione przez elementy trajektorii swobodnej. Oszacowanie niemierzalnego zakłócenia oblicza się na podstawie modelu neuronowego, tak samo jak w algorytmie NO – wzór (3.134).



Rys. 3.40. Struktura układu regulacji z algorytmem z Sukcesywną Linearyzacją (SL)

Można zauważyć, że stosowane w algorytmie NPL suboptymalne równanie predykcji (3.142) jest bardzo podobne do równania stosowanego w algorytmach liniowych, wykorzystujących do predykcji modele liniowe o stałych parametrach. Różnice są dwie: w algorytmie liniowym stosuje się stałą, niezależną od punktu pracy macierz dynamiczną $\boldsymbol{M}$ oraz, co naturalne, model liniowy jest stosowany do obliczana trajektorii swobodnej. Można przyjąć, że model zlinearyzowany służy nie tylko do obliczenia współczynników odpowiedzi skokowych tworzących macierz dynamiczną, ale również do wyznaczenia trajektorii swobodnej (a tym samym oszacowania niemierzalnego zakłócenia). Podejście takie znane jest jako algorytm z Sukcesywną Linearyzacją (SL). Jego struktura została przedstawiona na rys. 3.40.

W algorytmie SL trajektoria swobodna może być w najprostszy sposób obliczana jest rekurencyjnie (dla $p = 1, \ldots, N$) ze wzoru

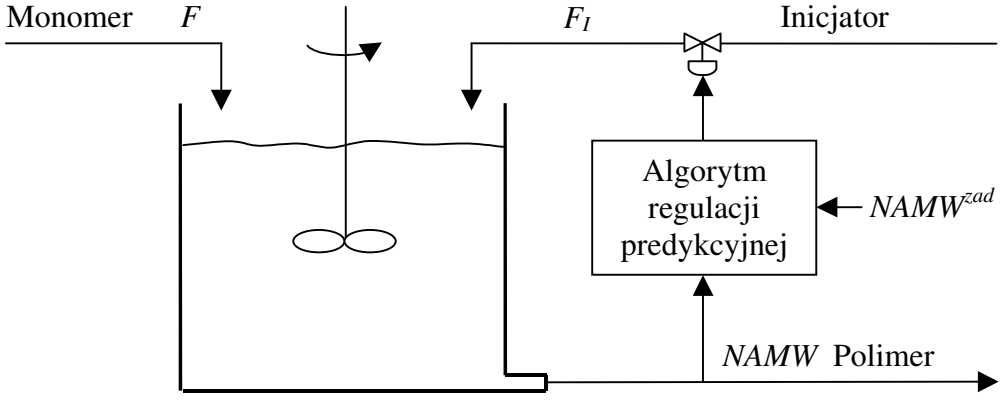
$$y^0(k+p\,|\,k) = \sum_{j=1}^{I_{\mathrm{uf}}(p)} b_j(k) u(k-1) + \sum_{j=I_{\mathrm{uf}}(p)+1}^{I_u} b_j(k) u(k-\tau+1-j+p) - \sum_{j=1}^{I_{\mathrm{yp}}(p)} a_{I_{\mathrm{u}}+j}(k) y^0(k-j+p\,|\,k) - \sum_{j=I_{\mathrm{yp}}(p)+1}^{n_{\mathrm{A}}} a_{I_{\mathrm{u}}+j}(k) y(k-j+p) + d(k) \tag{3.157}$$

gdzie

$$d(k) = y(k) - y(k \mid k-1)$$
$$y(k) - \left(\sum_{l=1}^{n_B} b_l(k) u(k-l) - \sum_{l=1}^{n_A} a_l(k) y(k-l) \right) \tag{3.158}$$

Przykład

Rozważanym obiektem regulacji jest reaktor polimeryzacji szczegółowo opisany w pracy [5]. Reaktor został schematycznie przedstawiony na rys. 3.41. Wielkością sterującą jest natężenie przepływu strumienia inicjatora F_I (wyrażone w m^3/h), wielkością wyjściową jest natomiast średnia masa cząsteczkowa *NAMW* (ang. Number Average Molecular Weight) (w kg/kmol). Zmiany natężenia przepływu strumienia monomeru *F* są zakłóceniami proces.



Rys. 3.41. Schemat reaktora polimeryzacji

Na podstawie znajomości zjawisk zachodzących w reaktorze polimeryzacji można sformułować jego model fizykochemiczny procesu. Model składa się z czterech zwyczajnych równań różniczkowych

$$\frac{\mathrm{d}C_{\mathrm{m}}(t)}{\mathrm{d}(t)} = -\left[Z_{\mathrm{P}} \exp\left(\frac{-E_{\mathrm{P}}}{RT} \right) + Z_{\mathrm{f_m}} \exp\left(\frac{-E_{\mathrm{f_m}}}{RT} \right) \right] C_{\mathrm{m}}(t) P_0(t) - \frac{F(t) C_{\mathrm{m}}(t)}{V} + \frac{F(t) C_{\mathrm{m_{in}}}}{V}$$

$$\frac{\mathrm{d}C_{\mathrm{I}}(t)}{\mathrm{d}(t)} = -Z_{\mathrm{I}} \exp\left(\frac{-E_{\mathrm{I}}}{RT} \right) C_{\mathrm{I}}(t) - \frac{F(t) C_{\mathrm{I}}(t)}{V} + \frac{F(t) C_{\mathrm{I_{in}}}}{V}$$

$$\frac{\mathrm{d}D_0(t)}{\mathrm{d}(t)} = \left[0{,}5 Z_{\mathrm{T_c}} \exp\left(\frac{-E_{\mathrm{T_c}}}{RT} \right) + Z_{\mathrm{T_d}} \exp\left(\frac{-E_{\mathrm{T_d}}}{RT} \right) \right] P_0^2(t) \tag{3.159}$$
$$+ Z_{\mathrm{f_m}} \exp\left(\frac{-E_{\mathrm{f_m}}}{RT} \right) C_{\mathrm{m}}(t) P_0(t) - \frac{F(t) D_0(t)}{V}$$

$$\frac{\mathrm{d}D_{\mathrm{I}}(t)}{\mathrm{d}(t)} = M_{\mathrm{m}} \left[Z_{\mathrm{P}} \exp\left(\frac{-E_{\mathrm{P}}}{RT} \right) + Z_{\mathrm{f_m}} \exp\left(\frac{-E_{\mathrm{f_m}}}{RT} \right) \right] C_{\mathrm{m}}(t) P_0(t) - \frac{F(t) D_l(t)}{V}$$

gdzie

$$P_0(t) = \sqrt{\frac{2f^* C_{\mathrm{I}}(t) Z_{\mathrm{I}} \exp\left(\frac{-E_{\mathrm{I}}}{RT}\right)}{Z_{\mathrm{T_d}} \exp\left(\frac{-E_{\mathrm{T_d}}}{RT}\right) + Z_{\mathrm{T_c}} \exp\left(\frac{-E_{\mathrm{T_c}}}{RT}\right)}} \tag{3.160}$$

oraz równania wyjścia

$$NAMW(t) = \frac{D_{\mathrm{I}}(t)}{D_0(t)} \tag{3.161}$$

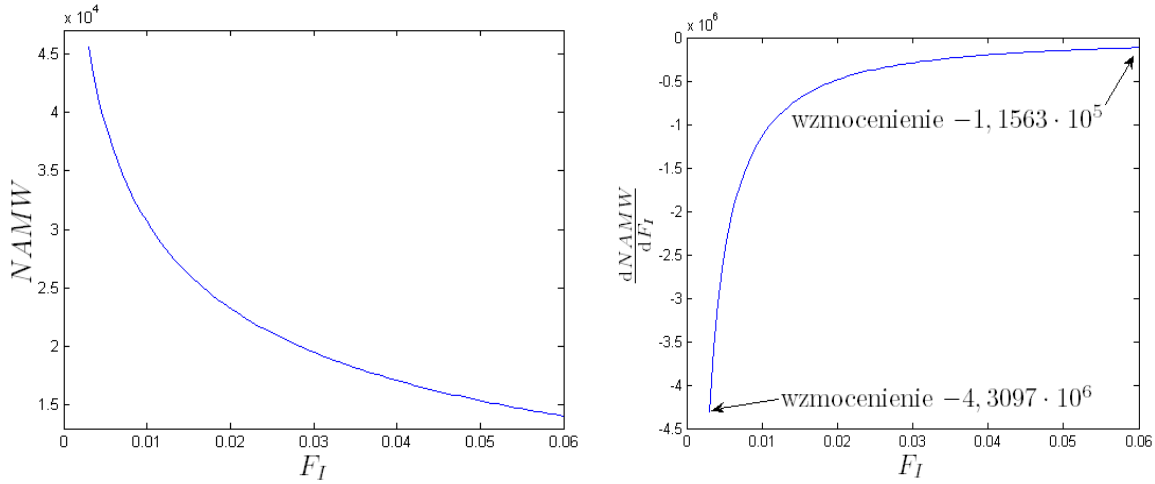
Parametry modelu podane są w tabeli 3.5.

W wybranym punkcie pracy sygnał sterujący ma wartość $F_{\mathrm{I}} = 0{,}028328\ \mathrm{m^3\ h^{-1}}$, zakłócenie $F = 1\ \mathrm{m^3\ h^{-1}}$, sygnał wyjściowy $NAMW = 20000\ \mathrm{kg\ kmol^{-1}}$, natomiast wartości czterech zmiennych stanu modelu są następujące: $C_{\mathrm{m}} = 5{,}3745\ \mathrm{kmol\ m^{-3}}$, $C_{\mathrm{I}} = 2{,}2433 \cdot 10^{-1}\ \mathrm{kmol\ m^{-3}}$, $D_0 = 3{,}1308 \cdot 10^{-3}\ \mathrm{kmol\ m^{-3}}$, $D_{\mathrm{I}} = 6{,}2616 \cdot 10^{-1}\ \mathrm{kmol\ m^{-3}}$.

Tabela 3.5. Parametry modelu fizykochemicznego reaktora polimeryzacji

$C_{\mathrm{I_{in}}} = 8\ \mathrm{kmol\ m^{-3}}$	$R = 8{,}314\ \mathrm{kJ\ kmol^{-1}\ K^{-1}}$
$C_{\mathrm{m_{in}}} = 6\ \mathrm{kmol\ m^{-3}}$	$T = 335\ \mathrm{K}$
$E_{\mathrm{T_c}} = 2{,}9442 \cdot 10^3\ \mathrm{kJ\ kmol^{-1}}$	$Z_{\mathrm{T_c}} = 3{,}8223 \cdot 10^{10}\ \mathrm{m^3\ kmol^{-1}\ h^{-1}}$
$E_{\mathrm{T_d}} = 2{,}9442 \cdot 10^3\ \mathrm{kJ\ kmol^{-1}}$	$Z_{\mathrm{T_d}} = 3{,}1457 \cdot 10^{11}\ \mathrm{m^3\ kmol^{-1}\ h^{-1}}$
$E_{\mathrm{f_m}} = 7{,}4478 \cdot 10^4\ \mathrm{kJ\ kmol^{-1}}$	$Z_{\mathrm{f_m}} = 1{,}0067 \cdot 10^{15}\ \mathrm{m^3\ kmol^{-1}\ h^{-1}}$
$E_{\mathrm{I}} = 1{,}2550 \cdot 10^5\ \mathrm{kJ\ kmol^{-1}}$	$Z_{\mathrm{I}} = 3{,}7920 \cdot 10^{18}\ \mathrm{h^{-1}}$
$E_{\mathrm{P}} = 1{,}8283 \cdot 10^4\ \mathrm{kJ\ kmol^{-1}}$	$Z_{\mathrm{P}} = 1{,}7700 \cdot 10^9\ \mathrm{m^3\ kmol^{-1}\ h^{-1}}$
$f^* = 0{,}58$	$V = 0{,}1\ \mathrm{m^3}$
$M_{\mathrm{m}} = 100{,}12\ \mathrm{kg\ kmol^{-1}}$	

a) b)



Rys. 3.42. a) Charakterystyka statyczna reaktora polimeryzacji, b) zależność wzmocnienia statycznego od punktu pracy

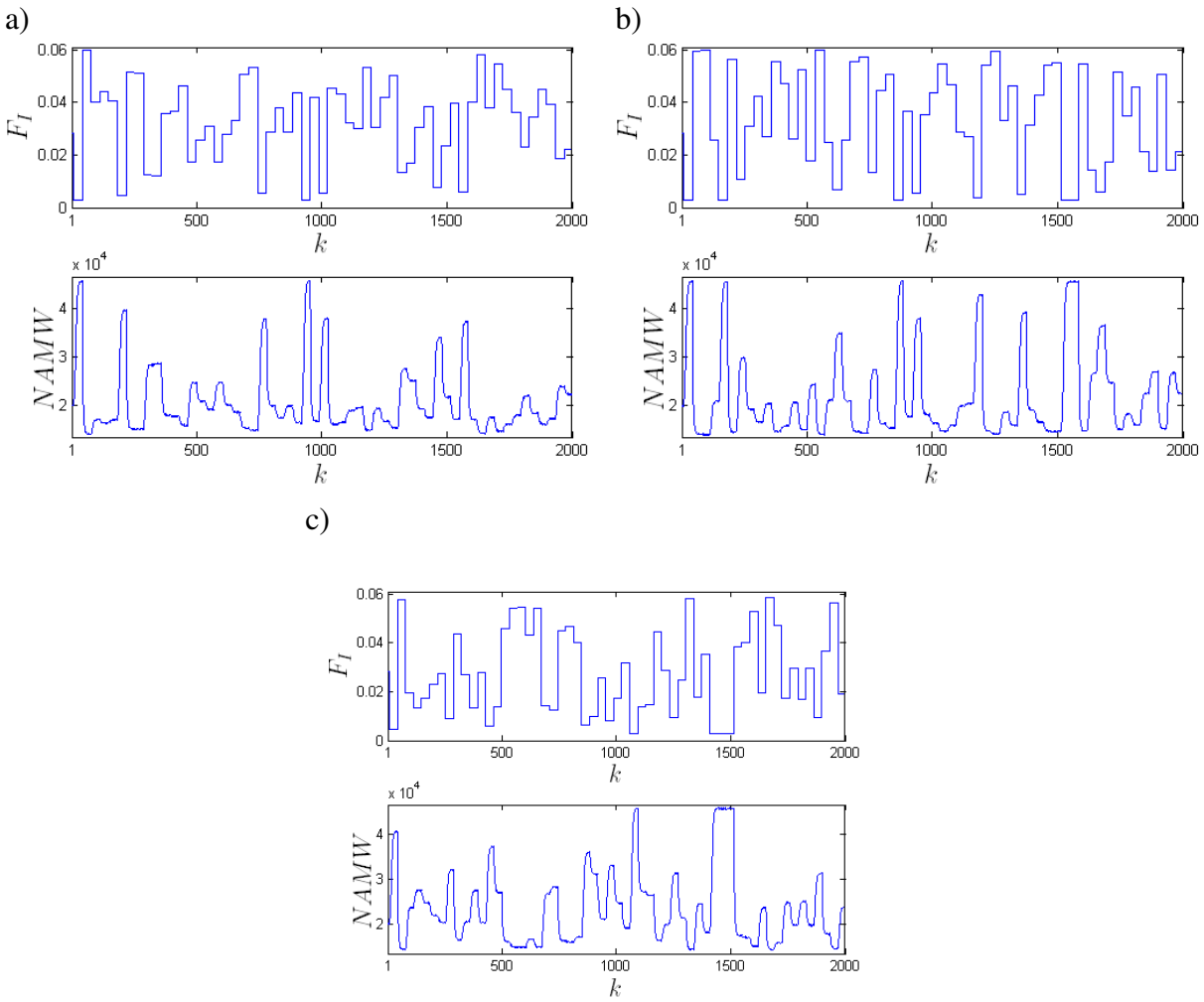
Reaktor polimeryzacji ma silnie nieliniowe właściwości, bardzo często skuteczność

różnych modeli i algorytmów regulacji jest testowana na tym procesie. Charakterystykę statyczną procesu pokazano na rys. 3.42. Sygnał sterujący zmienia się w granicach od $F_{\mathrm{I}}^{\min} = 0{,}003\ \mathrm{m^3\ h^{-1}}$ (czemu odpowiada sygnał wyjściowy $NAMW = 45524\ \mathrm{kg\ kmol^{-1}}$) do $F_{\mathrm{I}}^{\max} = 0{,}06\ \mathrm{m^3\ h^{-1}}$ (czemu odpowiada $NAMW = 14069\ \mathrm{kg\ kmol^{-1}}$), przyjęto stałe zakłócenie $F = 1\ \mathrm{m^3\ h^{-1}}$. Wzmocnienie statyczne procesu, czyli nachylenie charakterystyki statycznej ($\frac{\mathrm{d}NAMW}{\mathrm{d}F_{\mathrm{I}}}$) zmienia się ponad 37 razy: od wartości $-4{,}3097 \cdot 10^6$ do $-1{,}1563 \cdot 10^5$.

Model fizykochemiczny określony równaniami (3.159), (3.160) i (3.161) jest stosowany jako symulowany proces. Na podstawie symulacji tego modelu są wygenerowane 3 zbiory danych: dane uczące, weryfikujące i testowe pokazane na rys. 3.43. Wszystkie zbiory liczą 2000 próbek, okres próbkowania wynosi 1,8 min. Do sygnału wyjściowego dodano szum, który imituje niedokładność pomiaru. Ponieważ sygnał wejścia i wyjścia mają zupełnie inne rzędy wielkości, przed modelowaniem wszystkie zbiory danych zostały przeskalowane

$$u = 100(F_{\mathrm{I}} - F_{\mathrm{I0}}), \quad y = 0{,}0001(NAMW - NAMW_0) \tag{3.162}$$

gdzie wielkości $F_{\mathrm{I0}} = 0{,}028328\ \mathrm{m^3\ h^{-1}}$ i $NAMW_0 = 20000\ \mathrm{kg\ kmol^{-1}}$ odpowiadają nominalnemu (początkowemu) punktowi pracy.



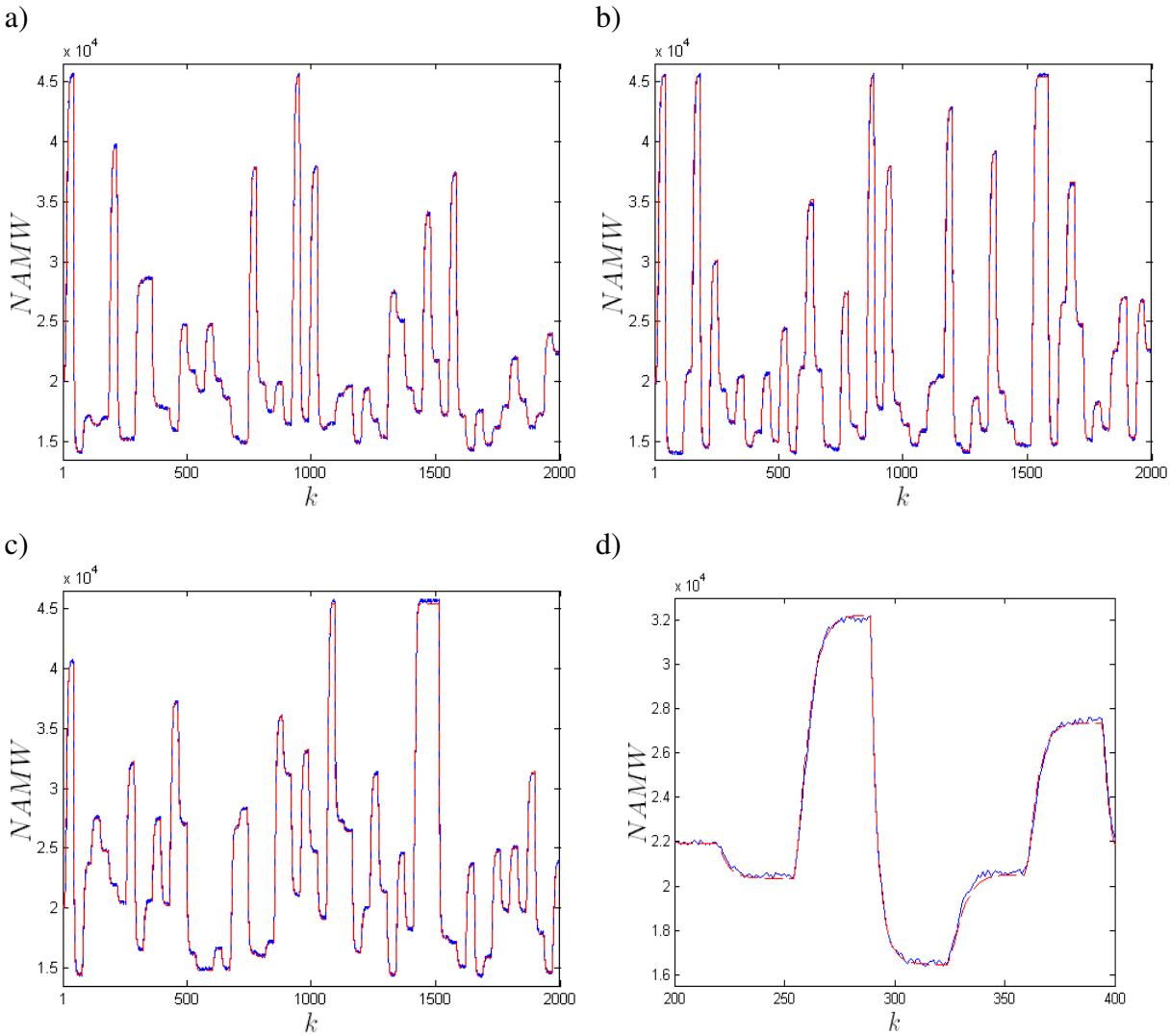
Rys. 3.43. Zbiory danych: a) zbiór uczący, b) zbiór weryfikujący, c) zbiór testowy

Pierwszym etapem prac jest dobór odpowiednio dokładnego modelu procesu. Wykorzystano dwuwarstwową jednokierunkową sieć neuronową (MLP), w warstwie ukrytej

zastosowano neurony z funkcją aktywacji $\varphi = \mathrm{tgh}$. Do uczenia sieci (minimalizacji błędu dla zbioru uczącego) zastosowano efektywny algorytm zmiennej metryki typu BFGS. Przetestowano kilkanaście różnych modeli, różniących się rzędem dynamiki (od pierwszego do trzeciego) i liczbą neuronów ukrytych. Dla każdej konfiguracji sieci uczenie powtórzono 10 razy. Metodą eksperymentalną stwierdzono, że model powinien mieć dynamikę drugiego rzędu oraz posiadać opóźnienie

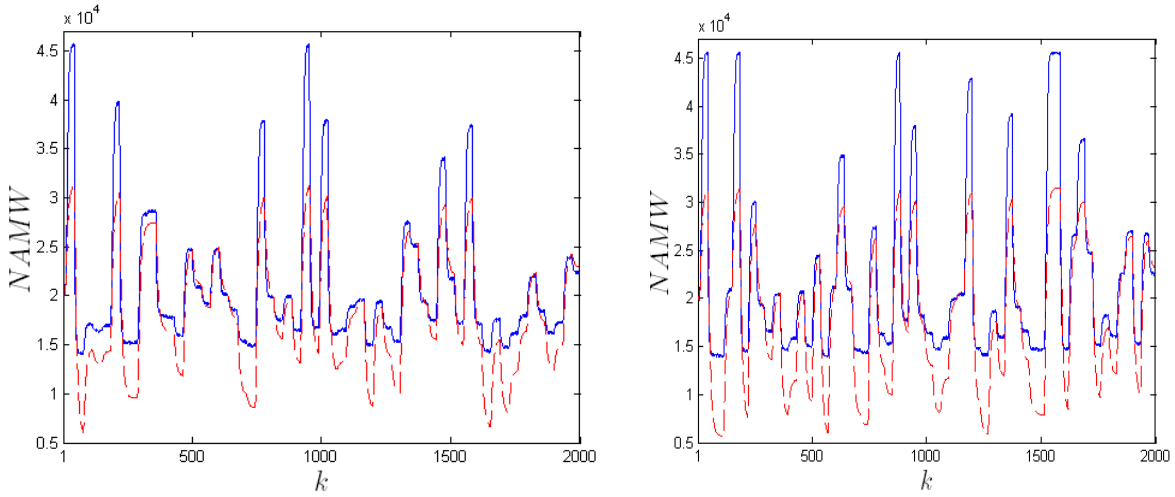
$$y(k) = f(u(k-2), y(k-1), y(k-2)) \tag{3.163}$$

Argumenty modelu określone są więc przez stałe $\tau = n_A = n_B = 2$. Model ma 3 wejścia ($u(k-2)$, $y(k-1)$, $y(k-2)$), jedno wyjście ($y(k)$) i $K = 6$ neuronów ukrytych. Dla zbioru uczącego błąd wybranego modelu wynosi $SSE = 5{,}559159 \cdot 10^{-1}$, dla zbioru weryfikującego $SSE = 1{,}190907 \cdot 10^{0}$, natomiast dla zbioru testowego $SSE = 1{,}039309 \cdot 10^{0}$. Porównanie rzeczywistego sygnału wyjściowego procesu i wyjścia modelu neuronowego dla trzech zbiorów danych przedstawiono na rys. 3.44. Widać, że model jest bardzo dokładny, dla obu zbiorów prawidłowo oddaje charakter procesu w różnych punktach pracy.



Rys. 3.44. Wyjście procesu oraz wyjście modelu neuronowego: a) zbiór danych uczących, b) zbiór danych weryfikujących, c) zbiór danych testowych, d) powiększony fragment zbioru danych testowych

a) b)



Rys. 3.45. Wyjście procesu oraz wyjście modelu liniowego: a) zbiór danych uczących, b) zbiór danych weryfikujących

Na podstawie tego samego zbioru danych uczących wyznaczono również model liniowy

$$y(k) = b_2 u(k-2) - a_1 y(k-1) - a_2 y(k-2)) \tag{3.164}$$

Model liniowy ma takie same wejścia jak model neuronowy. Niestety, na skutek nieliniowości procesu model liniowy jest bardzo niedokładny. Dla zbioru uczącego $SSE = 3{,}402260 \cdot 10^2$, dla zbioru testowego $SSE = 6.255822 \cdot 10^2$. W przeciwieństwie do modelu neuronowego, błąd dla zbioru testowego nie jest liczony ponieważ model liniowy nie jest zaakceptowany do zastosowania w algorytmach regulacji. Na rys. 3.45 porównano wyjście procesu i wyjście modelu liniowego dla zbioru uczącego i weryfikującego. W tabeli 3.6 porównano dokładność modelu neuronowego i liniowego.

Tabela 3.6. Porównanie dokładności (błędu) modelu neuronowego i liniowego

	Model neuronowy	Model liniowy
Zbiór uczący	$5{,}559159 \cdot 10^{-1}$	$3{,}402260 \cdot 10^2$
Zbiór weryfikujący	$1{,}190907 \cdot 10^0$	$6.255822 \cdot 10^2$
Zbiór testowy	$1{,}039309 \cdot 10^0$	–

Zasymulowano następujące trzy algorytmy regulacji predykcyjnej:

a) algorytm liniowy bazujący na modelu liniowym,

b) nieliniowy algorytm NO z bieżącą nieliniową optymalizacją powtarzaną w każdej iteracji,

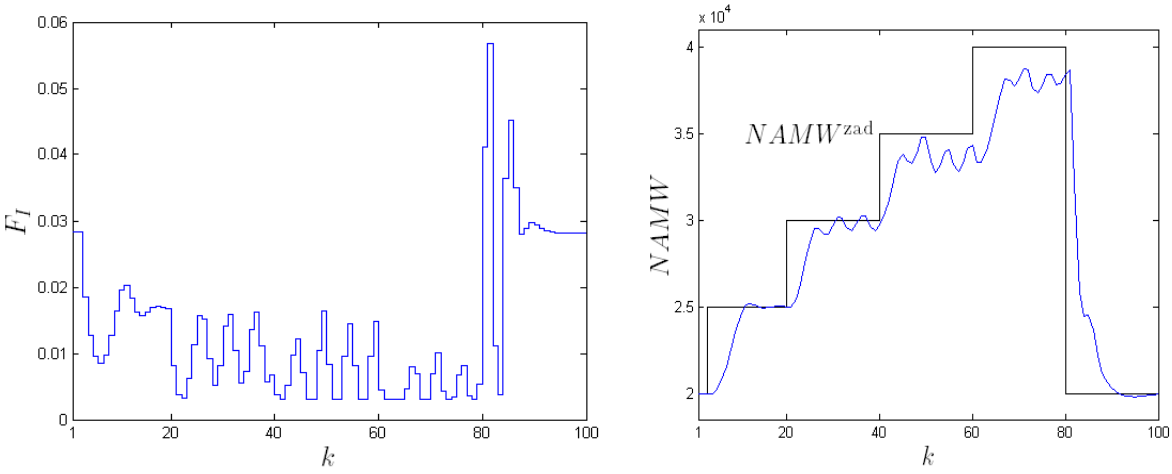
c) nieliniowy algorytm NPL z cykliczną linearyzacją modelu i optymalizacją kwadratową.

Dwa nieliniowe algorytmy wykorzystują ten sam neuronowy model procesu. Wszystkie algorytmy zaimplementowano w pakiecie Matlab. Do optymalizacji w algorytmie NO zastosowano algorytm optymalizacji nieliniowej SQP (ang. Sequential Quadratic Programming), jako punkt początkowy optymalizacji zastosowano $N_u - 1$ sygnałów sterujących obliczonych w poprzedniej iteracji i nie wykorzystanych do sterowania. Parametry trzech porównywanych algorytmów są takie same: $N = 10$, $N_u = 3$, $\lambda = 0{,}2$. Sygnał sterujący jest ograniczony: $F_I^{\min} = 0{,}003\ \mathrm{m^3\ h^{-1}}$, $F_I^{\max} = 0{,}06\ \mathrm{m^3\ h^{-1}}$. Okres próbkowania algorytmów, analogicznie jak okres próbkowania modeli, wynosi 1,8 min.

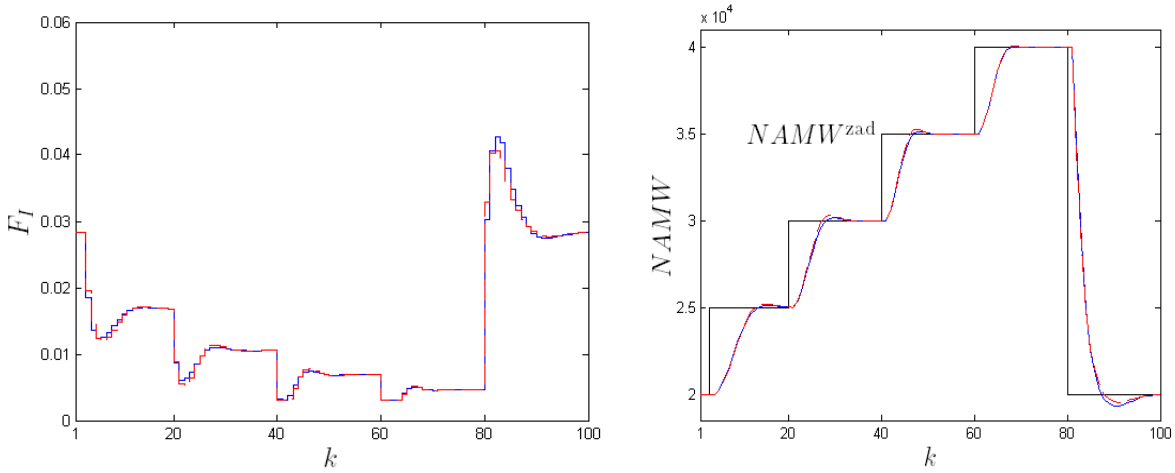
Wyniki symulacji algorytmu regulacji bazującego na modelu liniowym przedstawiono na rys. 3.46. Trajektoria zadania ($NAMW^{\text{zad}}$) ma postać pięciu skoków, ich zakres pokrywa cały zakres pracy reaktora. Ponieważ model liniowy jest bardzo niedokładny, dlatego też wykorzystujący go algorytm regulacji działa zadowalająco dla pierwszego, najmniejszego skoku wartości zadanej. W dalszej części symulacji algorytm jest niestabilny.

Na rys. 3.47 pokazano wyniki symulacji nieliniowych algorytmów regulacji predykcyjnej (NO i NPL) bazujących na modelu neuronowym. Oba algorytmy pracują stabilnie. Z łatwością można zauważyć, że z punktu widzenia przebiegów sygnału wejściowego i wyjściowego oba algorytmy zachowują się bardzo podobnie. Należy jednak podkreślić, że algorytm NO w każdej iteracji korzysta z nieliniowej optymalizacji, natomiast algorytm NPL z efektywnej obliczeniowo procedury programowania kwadratowego.

Na rys. 3.48 przedstawiono wyniki symulacji nieliniowych algorytmów regulacji predykcyjnej bazujących na modelu neuronowym z uwzględnieniem dodatkowego ograniczenia szybkości narastania sygnału sterującego $\Delta F_{\text{I}}^{\max} = 0{,}005\ \text{m}^3\ \text{h}^{-1}$. Przyjęcie takiego ograniczenia jest konieczne aby wyeliminować duże zmiany sygnału sterującego towarzyszące skokowym zmianom wartości zadanej. Szybkie i duże zmiany sygnału sterującego są niekorzystne, mogą doprowadzić do uszkodzenia elementów wykonawczych. W porównaniu z algorytmem, w którym nie uwzględniono dodatkowego ograniczenia (rys. 3.47), otrzymana trajektoria sygnału wyjściowego jest nieco wolniejsza, ale istniejące ograniczenia sygnału sterującego są zachowane. Ograniczenie niekorzystnych skoków sygnału sterującego można również osiągną zwiększając współczynnik kary λ, ale nie gwarantuje to dokładnego spełnienia ograniczenia, jak ma to miejsce wówczas, gdy jest ono uwzględnione w zadaniu optymalizacji. Co więcej, zwiększanie współczynnika kary może prowadzić do niepotrzebnego spowolnienia przebiegu wyjściowego. Analogicznie jak poprzednio, wyniki symulacji algorytmu NPL są bardzo podobne do wyników algorytmu NO.

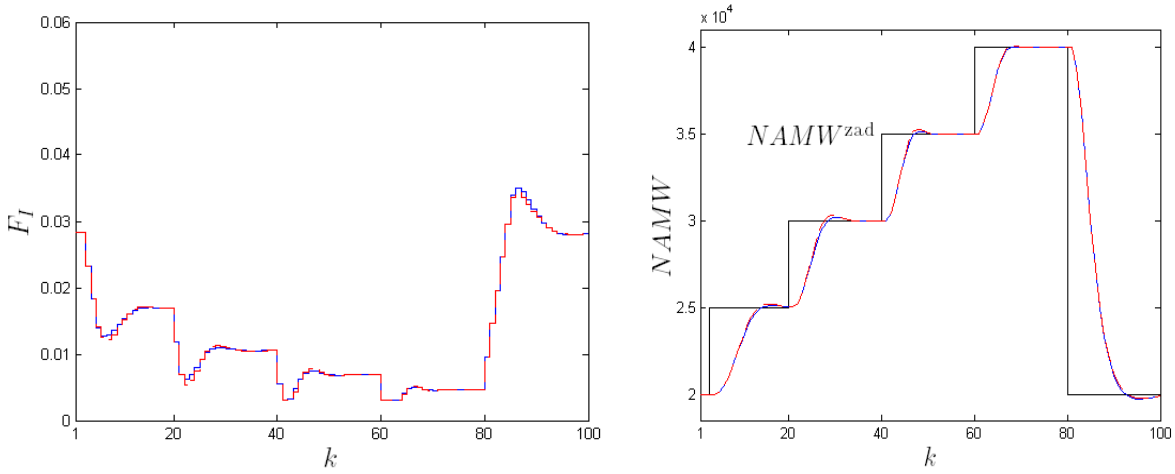


Rys. 3.46. Wyniki symulacji algorytmu regulacji predykcyjnej bazującego na modelu liniowym

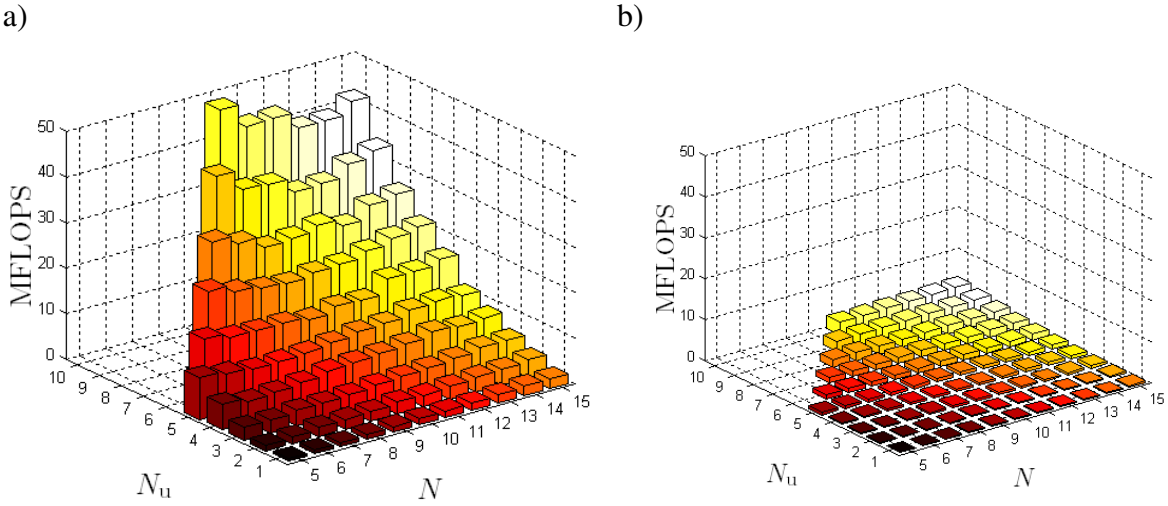


Rys. 3.47. Wyniki symulacji algorytmów regulacji predykcyjnej bazujących na modelu neuronowym: algorytm NPL oraz algorytm NO

Ponieważ trajektorie otrzymane w obydwu nieliniowych algorytmach regulacji predykcyjnej bazujących na tym samym modelu neuronowym są bardzo podobne, warto porównać ich złożoność obliczeniową. Dla horyzontów $N = 10$, $N_u = 3$ symulacja algorytmu NO wymaga nakładu obliczeń 4,110 MFLOPS (milionów operacji zmiennoprzecinkowych), podczas gdy symulacja algorytmu NPL wymaga tylko 0,405 MFLOPS. Na rys 3.49 porównano złożoność obliczeniową obu nieliniowych algorytmów regulacji dla różnych kombinacji horyzontów predykcji i sterowania ($N = 5,\ldots,15$, $N_u = 1,\ldots,10$). Algorytm NPL jest wielokrotnie mniej złożony obliczeniowo od algorytmu NO. W najgorszym przypadku stosunek nakładu obliczeń wynosi 4,28, w najlepszym 15,37, wartość średnia wynosi 10,59. Oprócz zademonstrowanej ilościowej efektywności obliczeniowej algorytmu NPL nie można zapominać, że procedura optymalizacji kwadratowej zawsze znajduje minimum globalne, czas obliczeń można przewidzieć, natomiast procedura nieliniowej optymalizacji może znaleźć płytkie minimum lokalne, jej czas obliczeń nie może być z góry znany.



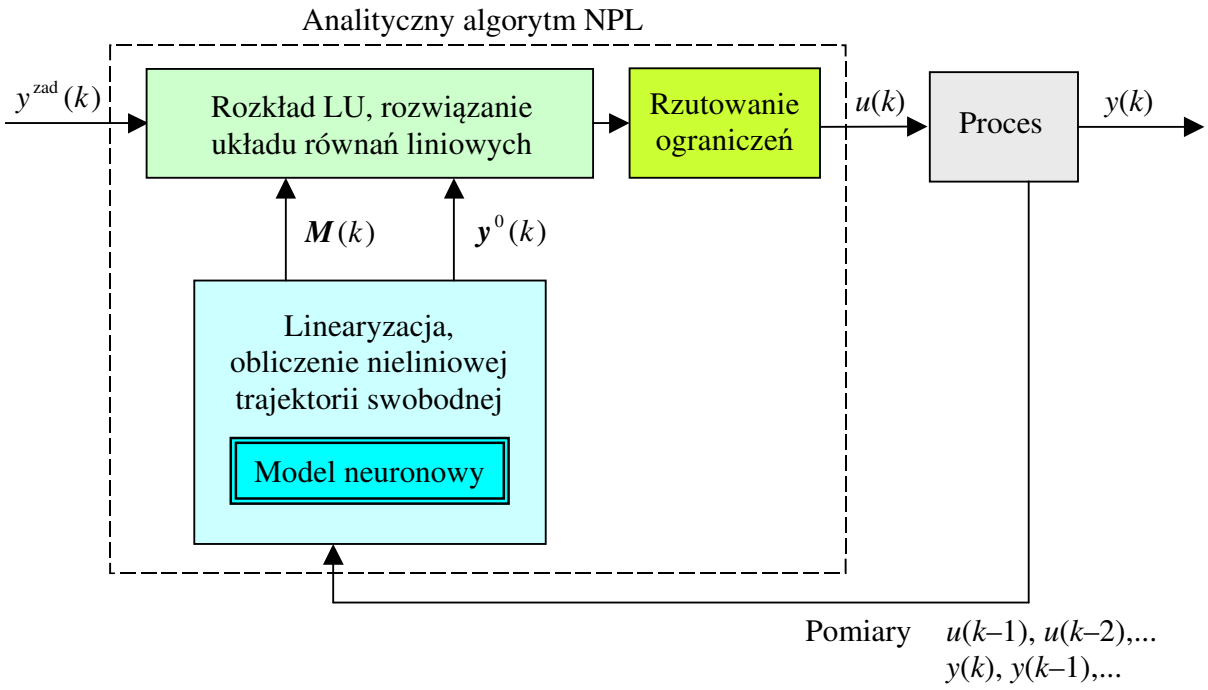
Rys. 3.48. Wyniki symulacji algorytmów regulacji predykcyjnej z ograniczeniem szybkości narastania sygnału sterującego $\Delta F_I^{max} = 0{,}005\ \mathrm{m}^3\ \mathrm{h}^{-1}$ bazujących na modelu neuronowym: algorytm NPL oraz algorytm NO



Rys. 3.49. Porównanie złożoności obliczeniowej nieliniowych algorytmów regulacji predykcyjnej bazujących na tym samym modelu neuronowym: a) algorytm NPL, b) algorytm NO

3.8.8. Analityczne algorytmy regulacji predykcyjnej z linearyzacją

Aby ograniczyć nakład obliczeń można zastosować analityczny algorytm NPL o strukturze przedstawionej na rys. 3.50. W miejsce stosowanej w klasycznym (numerycznym) algorytmie NPL optymalizacji kwadratowej stosuje się rozkład (faktoryzację) LU, a następnie rozwiązuje się prosty układ równań liniowych. Sygnał sterujący obliczany jest w sposób analityczny (bez optymalizacji, a więc bez ograniczeń), ale uzyskane rozwiązanie jest rzutowanie na zbiór dopuszczalny, który jest określony przez ograniczenia. Oczywiście, możliwe jest również zastosowanie analitycznego algorytmu SL, jedyna różnica polega na użyciu do obliczania trajektorii swobodnej zlinearyzowanego, a nie nieliniowego, modelu procesu.



Rys. 3.50. Struktura układu regulacji z analitycznym algorytmem NPL

Minimalizowana funkcja kryterialna ma klasyczną postać

$$J(k) = \left\| \boldsymbol{y}^{\text{zad}}(k) - \hat{\boldsymbol{y}}(k) \right\|^2 + \left\| \Delta \boldsymbol{u}(k) \right\|_{\boldsymbol{\Lambda}}^2 \tag{3.165}$$

Analogicznie jak w suboptymalnych algorytmach SL i NPL wykorzystuje się zasadę superpozycji (3.142), czyli zadanie optymalizacji ma postać

$$\min_{\Delta \boldsymbol{u}(k)} \{ J(k) = \left\| \boldsymbol{y}^{\text{zad}}(k) - \boldsymbol{M}(k) \Delta \boldsymbol{u}(k) - \boldsymbol{y}^0(k) \right\|^2 + \left\| \Delta \boldsymbol{u}(k) \right\|_{\boldsymbol{\Lambda}}^2 \} \tag{3.166}$$

Dzięki zastosowaniu do predykcji modelu zlinearyzowanego funkcja kryterialna jest kwadratowa. Przyrównując do zera (wektorowego) jej gradient

$$\frac{\mathrm{d}J(k)}{\mathrm{d}\Delta \boldsymbol{u}(k)} = -2\boldsymbol{M}^{\mathrm{T}}(k)(\boldsymbol{y}^{\text{zad}}(k) - \boldsymbol{M}(k)\Delta \boldsymbol{u}(k) - \boldsymbol{y}^0(k)) + 2\boldsymbol{\Lambda}\Delta \boldsymbol{u}(k) \tag{3.167}$$

otrzymuje się optymalny wektor przyrostów sygnału sterującego

$$\Delta \boldsymbol{u}(k) = \boldsymbol{K}(k)(\boldsymbol{y}^{\text{zad}}(k) - \boldsymbol{y}^0(k)) \tag{3.168}$$

gdzie

$$\boldsymbol{K}(k) = (\boldsymbol{M}^{\mathrm{T}}(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda})^{-1}\boldsymbol{M}^{\mathrm{T}}(k) \tag{3.169}$$

jest macierzą o wymiarowości $N_{\mathrm{u}} \times N$. Jest ona zależna od macierzy dynamicznej $\boldsymbol{M}(k)$ obliczanej w każdej iteracji algorytmu na podstawie linearyzacji modelu neuronowego. Oznacza to, że również macierz $\boldsymbol{K}(k)$ musi być obliczana w każdej iteracji algorytmu. Łatwo pokazać, że uzyskane rozwiązanie jest minimum globalnym zadania (3.166), ponieważ macierz drugich pochodnych (hesjan) funkcji kryterialnej $\frac{\mathrm{d}^2 J(k)}{\mathrm{d}\Delta \boldsymbol{u}^2(k)} = 2(\boldsymbol{M}^T(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda})$ jest dodatnio określona dla dodatnio określonej macierzy $\boldsymbol{\Lambda}$.

Do obliczania odwrotności macierzy $\boldsymbol{M}^T(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda}$ stosuje się rozkład LU tej macierzy, a następnie rozwiązuje się zestaw równań liniowych. Wyznacza się mianowicie macierze $\boldsymbol{P}(k)$, $\boldsymbol{L}(k)$ oraz $\boldsymbol{U}(k)$, dla których

$$\boldsymbol{L}(k)\boldsymbol{U}(k) = \boldsymbol{P}(k)\left(\boldsymbol{M}^T(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda}\right) \tag{3.170}$$

przy czym $\boldsymbol{L}(k)$ jest macierzą trójkątną dolną (z jedynkami na przekątnej), $\boldsymbol{U}(k)$ jest macierzą trójkątną górną, natomiast $\boldsymbol{P}(k)$ jest macierzą przestawień w eliminacji Gaussa, w każdym wierszu i kolumnie zawiera ona tylko jeden element równy 1, pozostałe elementy są zerowe. Obliczenie macierzy odwrotnej na podstawie rozkładu LU jest klasycznym problemem znanym z metod numerycznych i nie będzie tutaj omawiane.

Jeżeli ograniczenia sygnałów sterujących nie mogą być pominięte, pierwszy element wyznaczonego wektora przyszłych przyrostów sygnału sterującego ($\Delta u(k \mid k)$) jest rzutowany na zbiór rozwiązań dopuszczalnych. Uwzględnia się ograniczenie szybkości narastania aktualnie obliczanego sygnału sterującego

$$-\Delta u^{\max} \le \Delta u(k \mid k) \le \Delta u^{\max} \tag{3.171}$$

oraz ograniczenie wartości tego sygnału

$$u^{\min} \le u(k \mid k) \le u^{\max} \tag{3.172}$$

Procedura rzutowania obliczonego analitycznie przyrostu $\Delta u(k \mid k)$ na zbiór rozwiązań dopuszczalnych jest następująca

$$
\begin{aligned}
&\text{jeżeli } \Delta u(k \mid k) < -\Delta u^{\max} \quad \Delta u(k \mid k) = -\Delta u^{\max} \\
&\text{jeżeli } \Delta u(k \mid k) > \Delta u^{\max} \quad \Delta u(k \mid k) = \Delta u^{\max} \\
&u(k \mid k) = \Delta u(k \mid k) + u(k-1) \\
&\text{jeżeli } u(k \mid k) < u^{\min} \quad u(k \mid k) = u^{\min} \\
&\text{jeżeli } u(k \mid k) > u^{\max} \quad u(k \mid k) = u^{\max} \\
&u(k) = u(k \mid k)
\end{aligned}
\tag{3.173}
$$

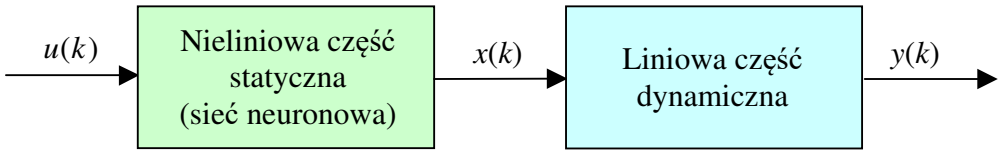
W każdej iteracji algorytmu (dyskretnej chwili k) zostają powtórzone następujące czynności:

1. Na podstawie nieliniowego (neuronowego) modelu procesu wyznacza się jego aproksymację liniową w aktualnym punkcie pracy (3.137). Model zlinearyzowany służy do obliczenia współczynników odpowiedzi skokowej $s_j(k)$ ze wzoru (3.140), które określają z kolei macierz dynamiczną $\boldsymbol{M}(k)$ (3.144).
2. Nieliniowy model procesu zostaje także wykorzystany do wyznaczenia nieliniowej trajektorii swobodnej $\boldsymbol{y}^0(k)$.
3. W wyniku rozkładu LU i rozwiązaniu układu równań liniowych oblicza się wektor przyszłych przyrostów sygnałów sterujących $\Delta\boldsymbol{u}(k)$.
4. Pierwszy element uzyskanego rozwiązania (przyrost $\Delta u(k \mid k)$) jest rzutowany na zbiór rozwiązań dopuszczalnych zgodnie ze wzorami (3.173).
5. Sygnał $\Delta u(k \mid k)$ zostaje wykorzystany do bieżącego sterowania, tzn. $u(k) = \Delta u(k \mid k) + u(k-1)$.
6. W następnej iteracji ($k := k+1$) następuje przejście do pierwszego kroku algorytmu.

3.8.9. Modele neuronowe w algorytmach regulacji predykcyjnej

W opisanych dotychczas algorytmach NO, NPL i SL wykorzystano model neuronowy typu MPL. Nic nie stoi jednak na przeszkodzie, aby wykorzystać model innego typu, na przykład RBF lub też model neuronowy w przestrzeni stanu. Wyprowadzenie odpowiednich zależności pozostawmy Czytelnikowi. Ich złożoność jest porównywalna ze stopniem skomplikowania wzorów dla sieci MLP.

W algorytmach regulacji predykcyjnej można właściwie wykorzystać większość znanych modeli neuronowych, zastosowanie konkretnego modelu może być podyktowane dostępnością algorytmu uczenia, zaletami modelu (dokładnością i małą liczbą parametrów), lub też po prostu preferencją projektanta. Warto jednak wspomnieć o neuronowych modelach Hammersteina i Wienera z wydzieloną częścią statyczną i dynamiczną. Na rys. 3.51 przedstawiono model Hammersteina, w którym statyczny blok nieliniowy, realizowany przez sieć neuronową, poprzedza liniowy blok dynamiczny. W modelu występuje sygnał pośredni x, nie jest on obecny w rzeczywistym procesie. W modelu Wienera kolejność obu bloków jest odwrotna. Modele Hammersteina i Wienera można z powodzeniem zastosować do modelowania wielu procesów technologicznych. Bardzo często bowiem nieliniowość procesu ma charakter statyczny, np. jest związana z nasyceniem zaworu.



Rys. 3.51. Struktura neuronowego modelu Hammersteina

Nieliniowa część statyczna jest opisana siecią neuronową typu MLP o jednym wejściu $u(k)$ i jednym wyjściu $x(k)$

$$x(k) = w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k)) \tag{3.174}$$

natomiast liniowa część dynamiczna równaniem

$$y(k) = \sum_{l=\tau}^{n_B} b_l x(k-l) - \sum_{l=1}^{n_A} a_l y(k-l) \tag{3.175}$$

Korzystając z powyższych wzorów otrzymuje się równanie sygnału wyjściowego modelu, dodatkowy sygnał *x* zostaje wyeliminowany

$$y(k) = f(\boldsymbol{x}(k)) = \sum_{l=1}^{n_B} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-l)) \right] - \sum_{l=1}^{n_A} a_l y(k-l) \tag{3.176}$$

Neuronowy model Hammersteina może być oczywiście wykorzystany w regulacji predykcyjnej. Warto podkreślić, że dzięki prostocie modelu, obliczenia są jeszcze mniej skomplikowane niż w przypadku klasycznego modelu MLP. Na przykład, współczynniki modelu zlinearyzowanego (3.137), to znaczy modelu $y(k) = \sum_{l=1}^{n_B} b_l(k) u(k-l) - \sum_{l=1}^{n_A} a_l(k) y(k-l)$, który jest stosowany w algorytmach SL i NPL, są obliczane następująco

$$a_l(k) = a_l \quad l = 1, \ldots, n_A \tag{3.177}$$

oraz

$$b_l(k) = \frac{\partial f(\overline{\boldsymbol{x}}(k))}{\partial u(k-l)} = \begin{cases} 0 & l = 1, \ldots, \tau - 1 \\ b_l \sum_{i=1}^{K} w_i^2 \dfrac{\partial \varphi(z_i(k-l))}{\partial z_i(k-l)} w_{i,1}^1 & l = \tau, \ldots, n_B \end{cases} \tag{3.178}$$

gdzie $z_i(k-l) = w_{i,0}^1 + w_{i,1}^1 u(k-l)$. Jeżeli neurony warstwy ukrytej mają funkcję aktywacji φ typu tangens hiperboliczny zachodzi

$$\frac{\partial \varphi(z_i(k-l))}{\partial z_i(k-l)} = 1 - \mathrm{tgh}^2(z_i(k-l)) \tag{3.179}$$

Na potrzeby wyprowadzenia nieliniowej trajektorii swobodnej stosowanej w algorytmie NPL należy najpierw napisać równanie predykcji, analogiczne do równania predykcji (3.125) neuronowego modelu MPL. Korzystając z ogólnego równania predykcji (3.123) otrzymuje się

$$\begin{aligned} \hat{y}(k+p \mid k) = & \sum_{l=1}^{I_{uf}(p)} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-\tau+1-l+p \mid k)) \right] \\ & + \sum_{l=I_{uf}(p)+1}^{I_u} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-\tau+1-l+p)) \right] \\ & - \sum_{l=1}^{I_{yp}(p)} a_l \hat{y}(k-l+p \mid k) - \sum_{l=I_{yp}(p)+1}^{n_A} a_l y(k-l+p) + d(k) \end{aligned} \tag{3.180}$$

Na potrzeby obliczania trajektorii swobodnej przyjmuje się, że przyrosty sterowania od chwili k są zerowe, czyli $u(k \mid k) = u(k+1 \mid k) = \ldots = u(k-1)$ oraz predykcje $\hat{y}(k+1 \mid k), \hat{y}(k+2 \mid k), \ldots$ zostają zastąpione przez elementy trajektorii swobodnej. Trajektoria swobodna obliczana jest więc rekurencyjnie (dla $p = 1, \ldots, N$) ze wzoru

$$\begin{aligned} y^0(k+p \mid k) = & \sum_{l=1}^{I_{\mathrm{uf}}(p)} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-1)) \right] \\ & + \sum_{l=I_{\mathrm{uf}}(p)+1}^{I_{\mathrm{u}}} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-\tau+1-l+p)) \right] \\ & - \sum_{l=1}^{I_{\mathrm{yp}}(p)} a_l y^0(k-l+p \mid k) - \sum_{l=I_{\mathrm{yp}}(p)+1}^{n_{\mathrm{A}}} a_l y(k-l+p) + d(k) \end{aligned} \tag{3.181}$$

Oszacowanie niemierzalnego zakłócenia oblicza się z uniwersalnego wzoru (3.124) i z równania modelu Hammersteina (3.176)

$$d(k) = y(k) - \left(\sum_{l=1}^{n_{\mathrm{B}}} b_l \left[w_0^2 + \sum_{i=1}^{K} w_i^2 \varphi(w_{i,0}^1 + w_{i,1}^1 u(k-l)) \right] - \sum_{l=1}^{n_{\mathrm{A}}} a_l y(k-l) \right) \tag{3.182}$$

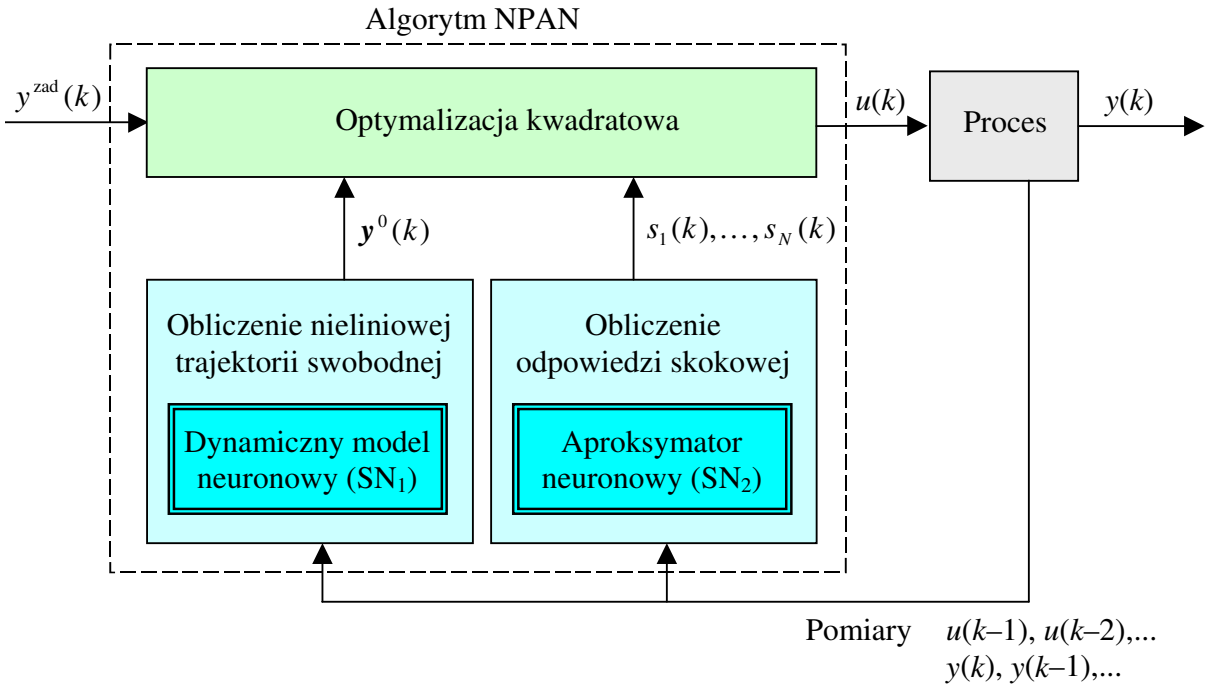
3.8.10. Algorytmy regulacji predykcyjnej z aproksymacją neuronową

W suboptymalnych algorytmach NPL oraz SL nieliniowy, na przykład neuronowy model

$$y(k) = f(u(k-\tau), \ldots, u(k-n_{\mathrm{B}}), y(k-1), \ldots, y(k-n_{\mathrm{A}})) \tag{3.183}$$

jest na bieżąco linearyzowany w aktualnym punkcie pracy. Uzyskane przybliżenie liniowe modelu nieliniowego jest stosowane do predykcji. Korzystając z wyprowadzenia (3.138), 3.139), dzięki cyklicznej linearyzacji suboptymalne równanie predykcji ma postać (3.142), czyli $\hat{\boldsymbol{y}}(k) = \boldsymbol{M}(k)\Delta\boldsymbol{u}(k) + \boldsymbol{y}^0(k)$, przy czym macierz dynamiczna $\boldsymbol{M}(k)$ jest obliczana na podstawie modelu zlinearyzowanego, natomiast trajektoria swobodna $\boldsymbol{y}^0(k)$ jest obliczana na podstawie pełnego nieliniowego modelu procesu (algorytm NPL) lub modelu zlinearyzowanego (algorytm SL).

Tematem bieżącego podrozdziału są algorytmy regulacji predykcyjnej z aproksymacją neuronową. Omówiono dwie wersje algorytmu z Nieliniową Predykcją i Aproksymacją Neuronową (NPAN): w pierwszej wersji ograniczenia są uwzględniane w sposób klasyczny (są częścią zadania optymalizacji) [15], w drugiej wersji sygnał sterujący obliczany jest w sposób analityczny (bez ograniczeń), ale uzyskane rozwiązanie jest następnie rzutowanie na zbiór dopuszczalny ograniczeń [17]. W obu wersjach algorytmu NPAN wykorzystuje się modele neuronowe zaprojektowane specjalnie do regulacji predykcyjnej.



Rys. 3.52. Struktura układu regulacji z algorytmem z Nieliniową Predykcją i Aproksymacją Neuronową (NPAN)

Model zastosowany w algorytmie NPAN oraz omawiany algorytm regulacji pokazano na rys. 3.52. Struktura modelu została zainspirowana suboptymalnym równaniem predykcji stosowanym w algorytmie NPL. W klasycznym algorytmie NPL stosuje się jeden model (3.183), natomiast model użyty w algorytmie NPAN zawiera dwie sieci. Pierwsza z nich (SN_1) jest zwykłym dynamicznym modelem neuronowym, takim samym jak model stosowany w algorytmie NPL. Model ten jest stosowany wyłącznie do obliczania nieliniowej trajektorii swobodnej $\boldsymbol{y}^0(k)$. Druga sieć neuronowa (SN_2) na bieżąco wyznacza współczynniki odpowiedzi skokowej $s_1(k),\ldots,s_N(k)$ przybliżenia liniowego modelu dynamicznego (SN_1). Współczynniki te zależą od aktualnego punktu pracy. W algorytmie NPAN stosuje się to samo równanie predykcji (3.183) co w algorytmie NPL, ale dzięki zastosowaniu drugiej sieci neuronowej nie ma potrzeby cyklicznej bieżącej linearyzacji modelu i obliczania współczynników odpowiedzi skokowej. Sieć neuronowa oblicza (aproksymuje) te współczynniki dla aktualnego punktu pracy bez jawnej linearyzacji. W algorytmie APAN rozwiązuje się kwadratowe zadanie optymalizacji algorytmu NPL (3.150), elementy macierzy dynamicznej $\boldsymbol{M}(k)$ aproksymowane są na bieżąco przez sieć SN_2. Jak pokazano w pracy [15] omawiany algorytm APAN charakteryzuje się niższą złożonością obliczeniową niż klasyczny algorytm NPL.

W każdej iteracji algorytmu (dyskretnej chwili k) zostają powtórzone następujące czynności:

1. Na podstawie neuronowego modelu procesu (sieć SN_1) wyznacza się nieliniową trajektorię swobodną $\boldsymbol{y}^0(k)$.
2. Aproksymator neuronowy (sieć SN_2) wyznacza współczynniki odpowiedzi skokowej $s_1(k),\ldots,s_N(k)$ modelu zlinearyzowanego, które określają macierz dynamiczną $\boldsymbol{M}(k)$ (3.144).
3. Rozwiązane zostaje zadanie programowania kwadratowego (3.150), w wyniku czego oblicza się wektor przyszłych przyrostów sygnałów sterujących $\Delta\boldsymbol{u}(k)$.

4. Pierwszy element wyznaczonej sekwencji zostaje wykorzystany do bieżącego sterowania, tzn. $u(k) = \Delta u(k \mid k) + u(k-1)$.
5. W następnej iteracji ($k := k+1$) następuje przejście do pierwszego kroku algorytmu i cała procedura zostaje powtórzona.

Nieco miejsca należy poświęcić uczeniu aproksymatora neuronowego. Dla aktualnego punktu pracy sieć SN_2 wyznacza współczynniki odpowiedzi skokowej

$$\begin{aligned} s_1(k) &= g_1(\boldsymbol{x}(k)) \\ &\vdots \\ s_N(k) &= g_N(\boldsymbol{x}(k)) \end{aligned} \tag{3.184}$$

modelu zlinearyzowanego. Punkt pracy określony jest przez wektor sygnałów

$$\boldsymbol{x}(k) = \left[u(k-1) \quad \ldots \quad u(k-\overline{n}_\mathrm{B}) \quad y(k) \quad \ldots \quad y(k-\overline{n}_\mathrm{A})\right]^\mathrm{T} \tag{3.185}$$

Najprostszym rozwiązaniem jest przyjęcie $\overline{n}_\mathrm{A} = n_\mathrm{A}$ i $\overline{n}_\mathrm{B} = n_\mathrm{B}$ przy czym stałe n_A i n_B określają rząd dynamiki dynamicznego modelu neuronowego (SN_1). Funkcja $g : \Re^{\overline{n}_\mathrm{A}+\overline{n}_\mathrm{B}+1} \to \Re^N$ jest realizowana przez sieć neuronową, np. typu MLP lub RBF.

Opóźnione sygnały wejściowe i wyjściowe definiujące aktualny punkt pracy (3.185) są wejściami aproksymatora neuronowego, natomiast jego wyjściami są współczynniki odpowiedzi skokowej modelu zlinearyzowanego. Można podać co najmniej trzy metody generacji danych potrzebnych do uczenia, weryfikacji i testowania sieci SN_2:

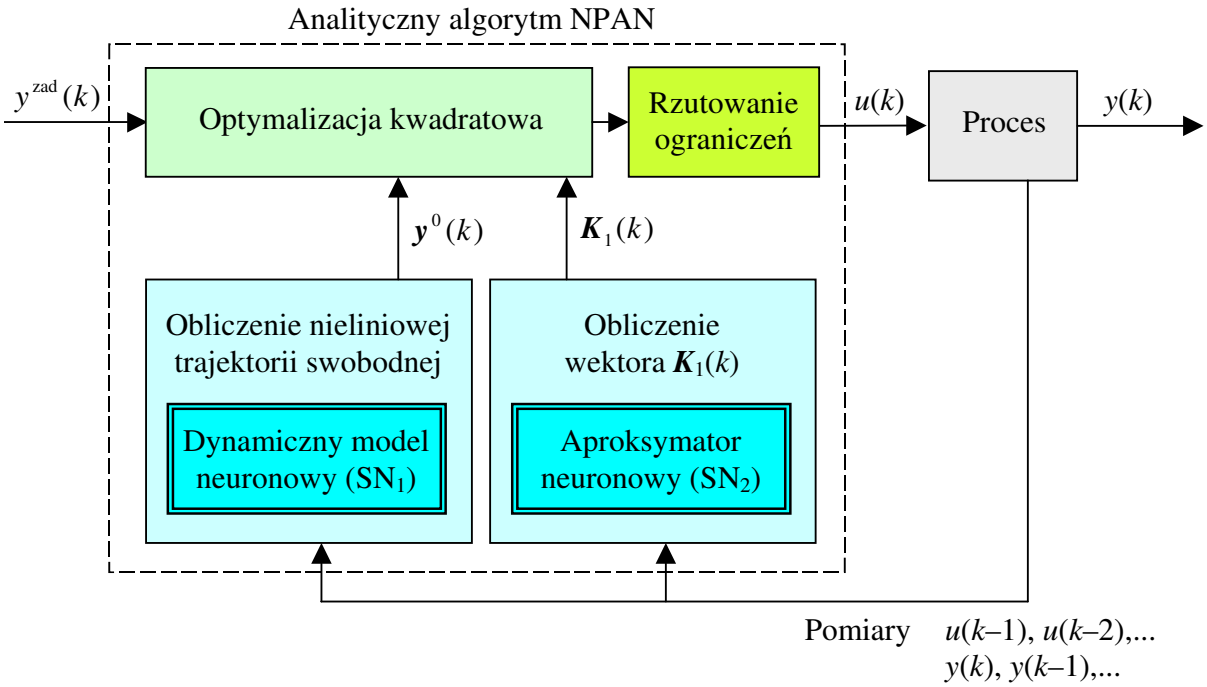
1. Dynamiczny model neuronowy SN_1 jest symulowany w pętli otwartej (bez algorytmu regulacji), jako pobudzenie wykorzystuje się sygnały użyte podczas uczenia lub podobne. W trakcie symulacji model jest sukcesywnie linearyzowany dla aktualnego punktu pracy, czemu towarzyszy obliczanie współczynników odpowiedzi skokowej modelu zlinearyzowanego.
2. Dynamiczny model neuronowy SN_1 jest wykorzystany w algorytmie NPL. Algorytm ten jest symulowany dla serii skoków wartości zadanej. Podczas symulacji algorytmu model jest sukcesywnie linearyzowany dla różnych punktów pracy, wyznaczane są współczynniki odpowiedzi skokowej modelu zlinearyzowanego.
3. Dla różnych punktów pracy rejestruje się odpowiedzi skokowe procesu [18]. W takiej sytuacji punkt pracy może być zdefiniowany w najprostszy sposób, za pomocą sygnałów $u(k-1)$ i (lub) $y(k)$.

Gdy projektowany jest algorytm dla rzeczywistego procesu, symulacje modelu (metoda 1) lub algorytmu (metoda 2) zostają zastąpione odpowiedziami rzeczywistego procesu.

W drugiej wersji algorytmu NPAN sygnał sterujący obliczany jest w sposób analityczny (bez optymalizacji, a więc bez ograniczeń), ale uzyskane rozwiązanie jest następnie rzutowanie na zbiór dopuszczalny ograniczeń. Omawiany analityczny algorytm NPAN odpowiada więc analitycznemu algorytmowi NPL, w którym optymalny (bez uwzględnienia ograniczeń) wektor przyrostów sygnału sterującego wyraża się wzorem (3.168), tzn. $\Delta\boldsymbol{u}(k) = \boldsymbol{K}(k)(\boldsymbol{y}^{\mathrm{zad}}(k) - \boldsymbol{y}^0(k))$, gdzie macierz $\boldsymbol{K}(k) = (\boldsymbol{M}^\mathrm{T}(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda})^{-1}\boldsymbol{M}^\mathrm{T}(k)$ zależy od macierzy dynamicznej $\boldsymbol{M}(k)$ modelu zlinearyzowanego w aktualnym punkcie pracy. W bieżącej iteracji analitycznego algorytmu APAN oblicza się wyłącznie pierwszy przyrost sygnału sterującego (a nie przyrosty na całym horyzoncie sterowania)

$$\Delta u(k \mid k) = \boldsymbol{K}_1(k)(\boldsymbol{y}^{\mathrm{zad}}(k) - \boldsymbol{y}^0(k)) \tag{3.186}$$

gdzie $\boldsymbol{K}_1(k)$ jest pierwszym wierszem macierzy $\boldsymbol{K}(k)$. Wyznaczany przyrost sygnału sterującego zależy od różnicy między trajektorią zadaną a swobodną. Wektor $\boldsymbol{K}_1(k)$ zależy od aktualnego punktu pracy, ponieważ jest on funkcją macierzy dynamicznej $\boldsymbol{M}(k)$ zawierającej współczynniki odpowiedzi skokowej modelu zlinearyzowanego.



Rys. 3.53. Struktura układu regulacji z analitycznym algorytmem NPAN w wersji z rzutowaniem ograniczeń

W każdej iteracji klasycznego analitycznego algorytmu NPL (lub SL), model neuronowy jest linearyzowany w aktualnym punkcie pracy, na podstawie tej linearyzacji oblicza się współczynniki odpowiedzi skokowej określające macierz dynamiczną. Następnie należy obliczyć odwrotność macierzy $(\boldsymbol{M}^{\mathrm{T}}(k)\boldsymbol{M}(k)+\boldsymbol{\Lambda})^{-1}$, do czego można zastosować efektywną obliczeniowo metodę rozkładu LU. W omawianym algorytmie NPAN, którego struktura została przedstawiona na rys. 3.53, stosuje się dwie sieci neuronowe: pierwsza z nich (model dynamiczny SN_1) służy do obliczenia trajektorii swobodnej, natomiast druga sieć (aproksymator neuronowy SN_2) oblicza dla aktualnego punktu pracy wektor $\boldsymbol{K}_1(k)$, bez potrzeby jawnej linearyzacji modelu, obliczania jego odpowiedzi skokowej i odwracania macierzy. Dzięki zastosowaniu modelu neuronowego o specjalnej strukturze (dwie sieci) omawiany algorytm APAN w wersji z rzutowaniem ograniczeń jest bardziej efektywny obliczeniowo niż klasyczny algorytm analityczny z linearyzacją, liczeniem odpowiedzi skokowej i odwracaniem macierzy.

W każdej iteracji algorytmu (dyskretnej chwili k) zostają powtórzone następujące czynności:

1. Na podstawie neuronowego modelu procesu (sieć SN_1) wyznacza się nieliniową trajektorię swobodną $\boldsymbol{y}^0(k)$.
2. Aproksymator neuronowy (sieć SN_2) wyznacza elementy wektora $\boldsymbol{K}_1(k)$.
3. Aktualny przyrost sygnału sterującego ($\Delta u(k \mid k)$) zostaje obliczony ze wzoru (3.186).
4. Sygnał $\Delta u(k \mid k)$ zostaje rzutowany na zbiór rozwiązań dopuszczalnych (3.173).
5. Wyznaczone rozwiązanie zostaje wykorzystane do bieżącego sterowania, tzn. $u(k) = \Delta u(k \mid k) + u(k-1)$.

6. W następnej iteracji ($k := k+1$) następuje przejście do pierwszego kroku algorytmu i cała procedura zostaje powtórzona.

Aproksymator neuronowy (SN_2) wyznacza elementy wektora $\boldsymbol{K}_1(k) = \left[k_{1,1}(k) \quad \ldots \quad k_{1,N}(k)\right]^{\mathrm{T}}$ na podstawie aktualnego punktu pracy

$$\begin{aligned} k_{1,1}(k) &= g_1(\boldsymbol{x}(k)) \\ &\vdots \\ k_{1,N}(k) &= g_N(\boldsymbol{x}(k)) \end{aligned} \tag{3.187}$$

gdzie punkt pracy określony jest przez (3.185), analogicznie jak w algorytmie NPAN w wersji z ograniczeniami. Funkcja $g : \Re^{\bar{n}_{\mathrm{A}}+\bar{n}_{\mathrm{B}}+1} \to \Re^N$ jest realizowana przez sieć neuronową, np. typu MLP lub RBF.

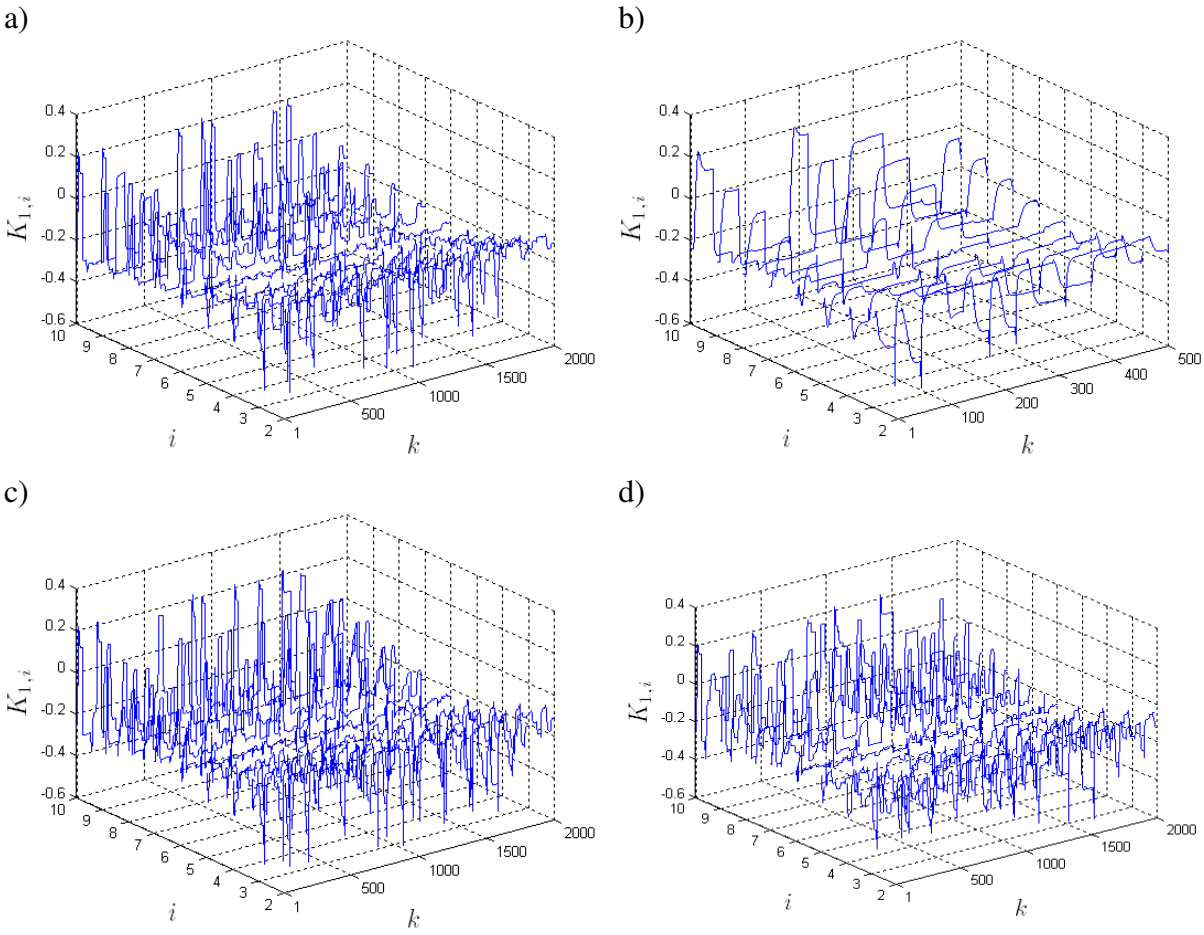
Opóźnione sygnały wejściowe i wyjściowe definiujące aktualny punkt pracy (3.185) są wejściami aproksymatora neuronowego, natomiast jego wyjściami są elementy wektora $\boldsymbol{K}_1(k)$. Można podać co najmniej dwie metody generacji danych potrzebnych do uczenia, weryfikacji i testowania sieci SN_2 (analogiczne do metod omówionych na potrzeby aproksymatora neuronowego zastosowanego w algorytmie NPAN z ograniczeniami):

1. Dynamiczny model neuronowy SN_1 jest symulowany w pętli otwartej (bez algorytmu regulacji), jako pobudzenie wykorzystuje się sygnały użyte podczas uczenia lub podobne. W trakcie symulacji model jest sukcesywnie linearyzowany dla aktualnego punktu pracy, obliczane są współczynniki odpowiedzi skokowej modelu zlinearyzowanego i wektor $\boldsymbol{K}_1(k)$.
2. Dynamiczny model neuronowy SN_1 jest wykorzystany w algorytmie NPL. Algorytm ten jest symulowany dla serii skoków wartości zadanej. Podczas symulacji algorytmu model jest sukcesywnie linearyzowany dla różnych punktów pracy, wyznaczane są współczynniki odpowiedzi skokowej modelu zlinearyzowanego, wykonywany jest rozkład LU macierzy $\boldsymbol{M}^{\mathrm{T}}(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda}$ i rozwiązywany jest układ równań liniowych w wyniku czego wyznacza się macierz odwrotną $(\boldsymbol{M}^{\mathrm{T}}(k)\boldsymbol{M}(k) + \boldsymbol{\Lambda})^{-1}$ i zależny od niej wektor $\boldsymbol{K}_1(k)$.

Oczywiście, gdy dostępny jest rzeczywisty proces, symulacje modelu lub algorytmu zostają zastąpione odpowiedziami rzeczywistego procesu.

Przykład

Rozważa się analityczny algorytm APAN w zastosowaniu do reaktora polimeryzacji. Aby wygenerować dane (elementy wektora $\boldsymbol{K}_1(k)$) potrzebne do uczenia aproksymatora neuronowego wykorzystano pierwszą metodę (symulacja modelu SN_1). Dla zbiorów danych wejściowych i wyjściowych z rys. 3.43 otrzymano zbiory danych pokazane na rys. 3.54.

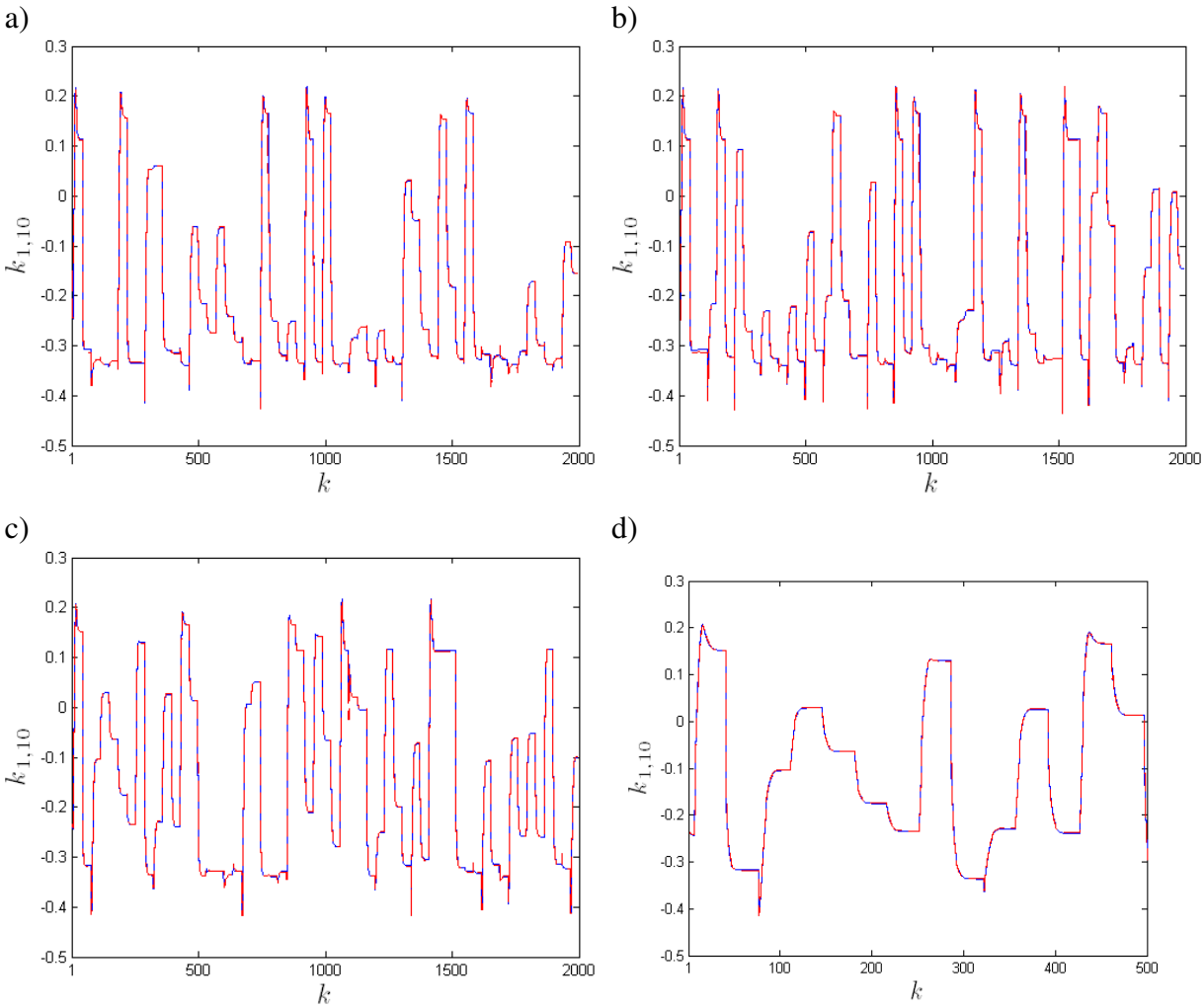


Rys. 3.54. a) Zbiór danych uczących, b) 500 pierwszych próbek zbioru danych uczących, c) zbiór danych weryfikujących, d) zbiór danych testowych

Jako aproksymator neuronowy zastosowano dwuwarstwową jednokierunkową sieć neuronową (MLP), w warstwie ukrytej zastosowano neurony z funkcją aktywacji $\varphi = \mathrm{tgh}$. Do uczenia zastosowano algorytm BFGS. Sieć neuronowa ma 5 wejść, punkt pracy określony jest przez sygnały $u(k-1), u(k-2), y(k), y(k-1), y(k-2)$. Z uwagi na opóźnienie procesu pierwszy współczynnik odpowiedzi skokowej ($s_1(k)$) jest zawsze zerowy, a więc pierwszy element wektora $\boldsymbol{K}_1(k)$ jest zerowy. Dlatego też sieć neuronowa ma 9 wyjść i realizuje odwzorowanie

$$\begin{aligned} k_{1,2}(k) &= g_2(u(k-1), u(k-2), y(k), y(k-1), y(k-2)) \\ &\vdots \\ k_{1,N}(k) &= g_N(u(k-1), u(k-2), y(k), y(k-1), y(k-2)) \end{aligned} \tag{3.188}$$

Stwierdzono, że sieć SN_2 powinna mieć 8 neuronów ukrytych. Na rys. 3.55 pokazano zmiany przykładowego, dziesiątego elementu wektora $\boldsymbol{K}_1(k)$, oraz jego aproksymację neuronową dla trzech zbiorów danych. Dla zbioru uczącego błąd wynosi $SSE = 9{,}195816 \cdot 10^{-2}$, dla zbioru weryfikującego $SSE = 2{,}300266 \cdot 10^{-1}$, dla zbioru testowego $SSE = 2{,}645983 \cdot 10^{-1}$.



Rys. 3.55. Przykładowy dziesiąty element ($k_{1,10}$) wektora $\boldsymbol{K}_1(k)$ oraz jego aproksymacja neuronowa: a) zbiór danych uczących, b) zbiór danych weryfikujących, c) zbiór danych testowych, d) powiększony fragment zbioru danych testowych

Zasymulowano następujące algorytmy regulacji predykcyjnej:

a) algorytm NO z nieliniową optymalizacją,

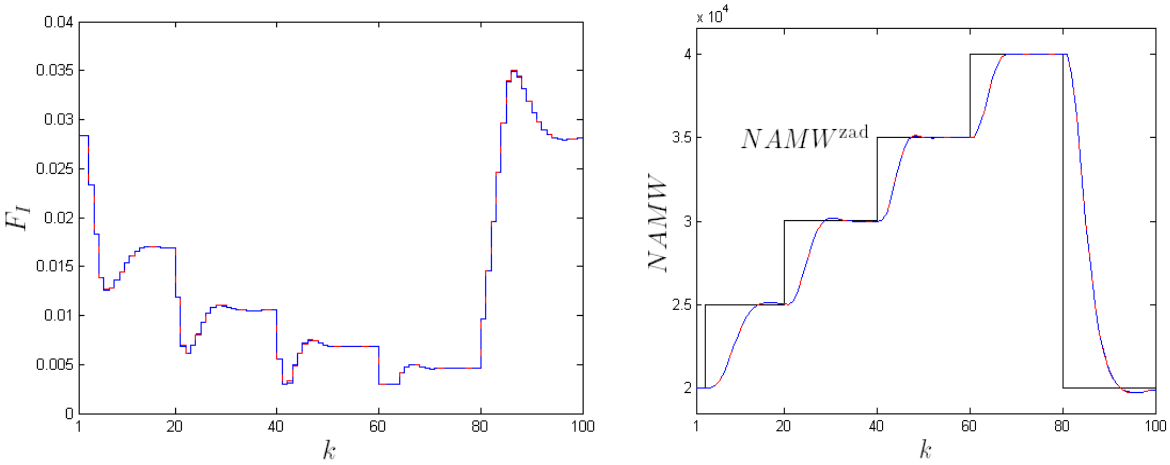
b) klasyczny algorytm NPL z bieżącą linearyzacją i programowaniem kwadratowym,

c) analityczny algorytm NPL z bieżącą linearyzacją i rozkładem LU,

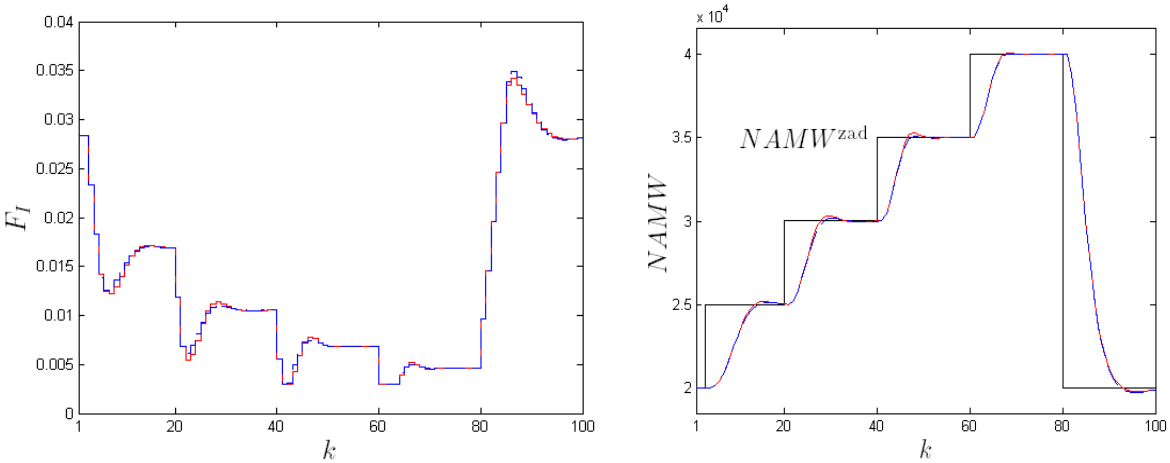
d) analityczny algorytm NPAN z aproksymacją neuronową i rzutowaniem ograniczeń.

Pierwsze trzy algorytmy wykorzystują ten sam model neuronowy, natomiast czwarty algorytm wykorzystuje ten model tylko do obliczania trajektorii swobodnej, elementy wektora $\boldsymbol{K}_1(k)$ są wyznaczane przez drugą sieć. Parametry wszystkich algorytmów są takie same, analogiczne jak w poprzednich eksperymentach.

Na rys. 3.56 porównano wyniki symulacji algorytmu NPL z optymalizacją kwadratową bazującego na klasycznym modelu neuronowym (jedna sieć) oraz wyniki analitycznego algorytmu NPAN z rzutowaniem ograniczeń. W algorytmie NPAN bieżący przyrost sygnału sterującego wyznacza się bez potrzeby optymalizacji, w sposób analityczny. Pomimo tego, trajektorie uzyskane w obu algorytmach są praktycznie takie same. Nie pokazano trajektorii uzyskanych w analitycznym algorytmie NPL, gdyż są one bardzo podobne. Nieco większe, choć wciąż małe, są różnice między trajektoriami otrzymanymi w analitycznym algorytmie NPAN i w algorytmie NO z nieliniową optymalizacją co pokazano na rys. 3.57.



Rys. 3.56. Wyniki symulacji algorytmu NPL z optymalizacją kwadratową bazującego na klasycznym modelu neuronowym oraz wyniki symulacji analitycznego algorytmu NPAN z rzutowaniem ograniczeń



Rys. 3.57. Wyniki symulacji algorytmu NO z nieliniową optymalizacją bazującego na klasycznym modelu neuronowym oraz wyniki symulacji analitycznego algorytmu NPAN z rzutowaniem ograniczeń

W tabeli 3.7 porównano jakość regulacji (sumę kwadratów błędów) oraz złożoność obliczeniową (MFLOPS) wszystkich czterech badanych algorytmów. Jakość regulacji wszystkich algorytmów jest zbliżona, ale różnią się one znacznie nakładem obliczeń. Złożoność obliczeniowa klasycznego algorytmu NPL jest przeszło 10 razy mniejszy od algorytmu NO. W analitycznym algorytmie NPL, dzięki zastosowania w miejsce optymalizacji kwadratowej rozkładu LU, udaje się jeszcze około dwukrotnie zmniejszyć nakład obliczeń. Dalsze zmniejszenie złożoności obliczeniowej (o około 17%) możliwe jest w analitycznym algorytmie NPAN.

Tabela 3.7. Porównanie jakości regulacji (*SSE*) oraz złożoności obliczeniowej

Algorytm	*SSE*	Złożoność obliczeniowa
NO	$2{,}210627 \cdot 10^9$	4,110
NPL	$2{,}211703 \cdot 10^9$	0,405
Analityczny NPL	$2{,}211703 \cdot 10^9$	0,223
Analityczny NPAN	$2{,}211651 \cdot 10^9$	0,185

Tabela 3.8. Wpływ horyzontu sterowania na złożoność obliczeniową

Algorytm	$N_u = 3$	$N_u = 5$	$N_u = 10$
NO	4,110	8,923	48,365
NPL	0,405	0,814	3,208
Analityczny NPL	0,223	0,267	0,502
Analityczny NPAN	0,185	0,185	0,185

W tabeli 3.8 pokazano wpływ horyzontu sterowania na złożoną obliczeniową algorytmów regulacji. Złożoność analitycznego algorytmu NPAN jest praktycznie niezależna od horyzontu (zależy ona głównie od horyzontu predykcji), natomiast złożoność analitycznego algorytmu NPL, choć też wyznacza się w nim tylko pierwszy przyrost sygnału sterującego, jest znacznie większa.

Warto podkreślić, że niska złożoność obliczeniowa analitycznego algorytmu APAN ma dwojaki charakter: nie tylko ilościowy, co pokazano w tabeli 3.7 i 3.8, ale przede wszystkim jakościowy. Algorytm nie wymaga linearyzacji, optymalizacji, rozkładu LU i rozwiązania układu równań liniowych.

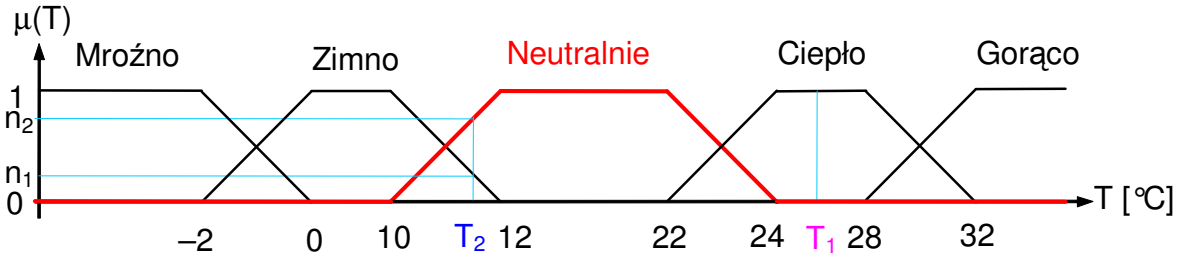
Na podstawie przeprowadzonych badań można sformułować dwie ważne obserwacje dotyczące opisywanych algorytmów NPAN z aproksymacją neuronową. Po pierwsze, sieci neuronowe pełniące funkcje aproksymatora odpowiedzi swobodnej i wektora $\boldsymbol{K}_1(k)$ dość łatwo się uczą. Nie ma potrzeby uczenia rekurencyjnego, niekiedy stosowanego przy uczeniu klasycznych dynamicznych modeli neuronowych (wejściami sieci SN_2 nie są jej wyjścia z poprzednich iteracji). Po drugie, algorytm NPAN jest bardzo mało wrażliwy na niedokładność aproksymatora SN_2, ewentualne niedoskonałości aproksymacji są niwelowane przez sprzężenie zwrotne algorytmu regulacji. W omawianym przypadku reaktora polimeryzacji można zastosować znacznie prostszy aproksymator, np. z pięcioma neuronami ukrytymi. Można również znacznie uprościć sposób określenia aktualnego punktu pracy, zbiór wejść aproksymatora neuronowego można ograniczyć do sygnałów $u(k-1), y(k)$. Taka sieć nie zapewnia tak doskonałej jakości jak sieć z ośmioma neuronami ukrytymi i pięcioma wejściami (dla zbioru uczącego otrzymuje się $SSE = 8{,}694486 \cdot 10^{-1}$, dla zbioru weryfikującego $SSE = 1{,}401156 \cdot 10^{0}$, dla zbioru testowego $SSE = 1{,}115099 \cdot 10^{0}$), ale wykorzystujący ją algorytm NPAN działa praktycznie tak samo jak algorytm, którego wyniki symulacji pokazano na rys. 3.57 i daje mniejszy nakład obliczeń (0,169). Oznacza to, że w stosunku do klasycznego algorytmu NPL uzyskuje się redukcję nakładu obliczeń o 24%.

4. Systemy rozmyte

Zacznijmy nasze rozważania od następującego eksperymentu myślowego. Załóżmy, że chcemy, aby pewna grupa osób („ekspertów") określiła, jaka wartość temperatury na dworze należy do zbioru wartości temperatury, przy których jest według nich: mroźno, zimno, neutralnie, ciepło, gorąco. Zauważmy, że w przypadku jednej osoby poszczególne wartości temperatury zostaną zapewne przypisane do poszczególnych pięciu zbiorów jednoznacznie, tzn. tak, jak to ma miejsce w zwykłej, klasycznej logice, w której każdy element należy do danego zbioru lub nie, co odpowiada wartościom logicznym 1 lub 0.

Zastanówmy się teraz, co się stanie, jeżeli spróbujemy zebrać opinie wszystkich „ekspertów". Prawdopodobnie różne osoby dokonają nieco różnego przyporządkowania poszczególnych wartości temperatury do poszczególnych zbiorów. W związku z tym część wartości temperatury zostanie przypisana przez wszystkie osoby do tych samych zbiorów, ale część do różnych. Co możemy w takiej sytuacji zrobić? Warto byłoby znaleźć sposób na opisanie takiej sytuacji. Sposób ten daje nam logika rozmyta, w której element nie tylko może należeć do danego zbioru lub nie, ale może należeć do tego zbioru w pewnym stopniu. Przynależność do zbioru opisują wtedy liczby z przedziału [0, 1]. Z kolei stopień przynależności danej wartości zmiennej (w naszym przykładzie – temperatury) do zbioru może być opisany pewną funkcją. Funkcje takie są nazywane w logice rozmytej **funkcjami przynależności** (ang. membership functions).

Wróćmy teraz do naszego przykładu. Załóżmy, że w wyniku przeprowadzenia powyższego eksperymentu w rzeczywistości i uśrednienia wyników otrzymaliśmy rezultat przedstawiony na rys. 4.1.



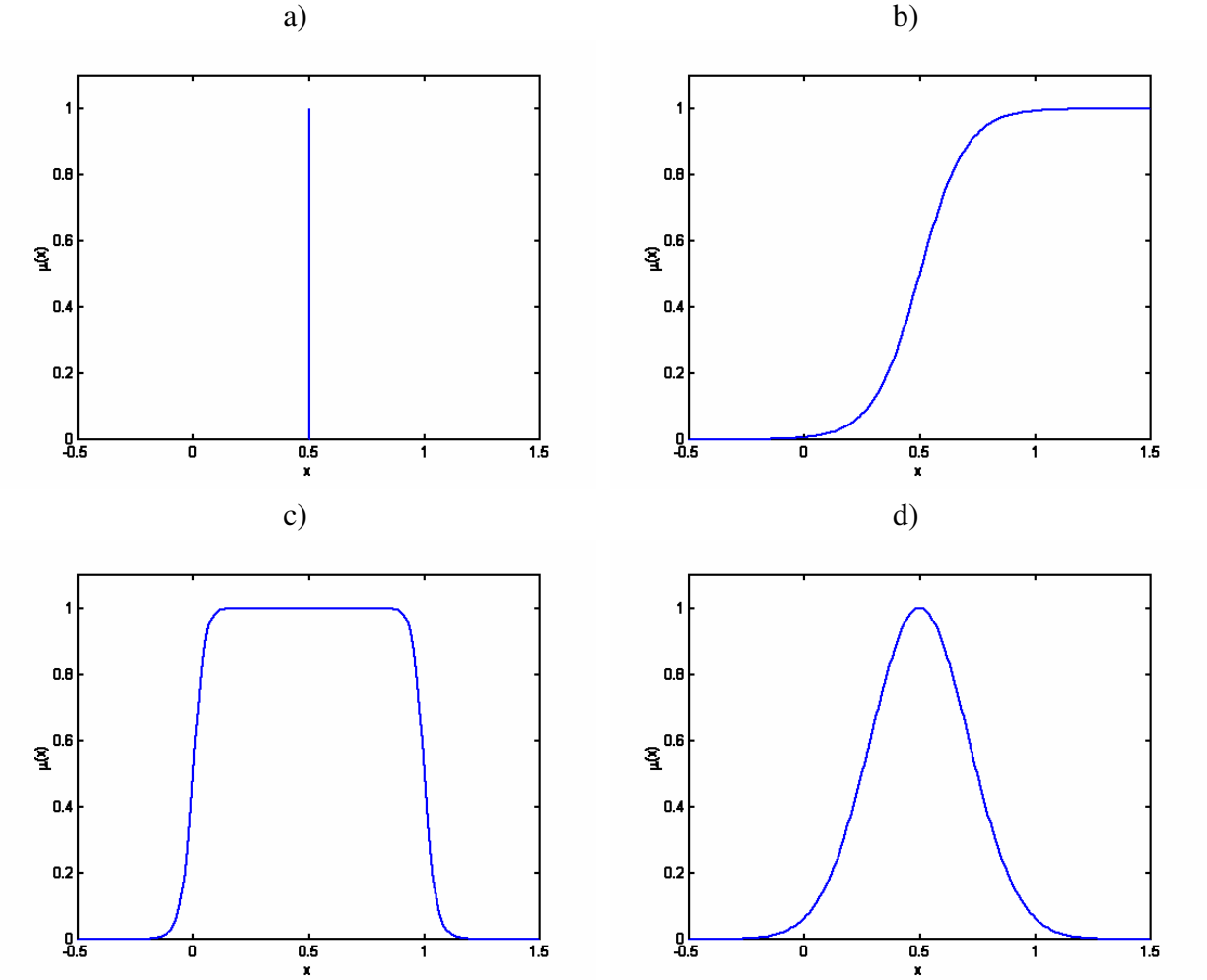
Rys. 4.1. Trapezowe funkcje przynależności

Zauważmy, że w sytuacji pokazanej na rysunku wartość temperatury T_1=26°C należy do zbioru wartości temperatur „ciepłych" ze stopniem przynależności 1 i tylko do tego zbioru (stopnie przynależności do pozostałych zbiorów są równe 0). Sytuację tę można interpretować, jako zgodność wszystkich „ekspertów" co do przyporządkowania tej wartości temperatury do danego zbioru. Sprawdźmy teraz, co się stanie dla wartości temperatury T_2=11,5°C. Należy ona do zbioru wartości temperatur „zimnych" ze stopniem przynależności n_1=0,25, do zbioru temperatur „neutralnych" ze stopniem przynależności n_2=0,75, a do pozostałych zbiorów – ze stopniem przynależności 0.

Zwróćmy uwagę na to, że na rys. 4.1 przedstawiono od razu pięć funkcji przynależności (dla pięciu zbiorów rozmytych). Wszystkie te funkcje są funkcjami trapezowymi, a przykładową funkcję przynależności do zbioru „neutralnych" wartości temperatury (oznaczona na rysunku kolorem czerwonym) można opisać następującym wzorem:

$$\mu_n(T) = \begin{cases} 0 & \text{dla} \quad T \le 10 \\ \dfrac{T-10}{12-10} & \text{dla } 10 < T < 12 \\ 1 & \text{dla } 12 \le T \le 22 \\ \dfrac{24-T}{24-22} & \text{dla } 22 < T < 24 \\ 0 & \text{dla} \quad T \ge 24. \end{cases} \tag{4.1}$$

Analogicznie można opisać wzorami wszystkie pozostałe funkcje przynależności przedstawione na rys. 4.1. Funkcje trapezowe nie są jedynymi, których można użyć.



Rys. 4.2. Różne rodzaje funkcji przynależności:
a) singleton, b) funkcja sigmoidalna, c) funkcja dzwonowa, d) funkcja Gaussa

Do częściej wykorzystywanych funkcji przynależności należą:

— singleton (często wykorzystywany w następnikach modeli rozmytych)

$$\mu(x) = \begin{cases} 0 \text{ dla } x < c, \\ 1 \text{ dla } x = c, \\ 0 \text{ dla } x > c, \end{cases} \tag{4.2}$$

gdzie c jest parametrem odpowiadającym za położenie singletonu;

— funkcja sigmoidalna

$$\mu(x) = \frac{1}{1+e^{-a(x-c)}} \tag{4.3}$$

gdzie parametr *c* decyduje o położeniu funkcji (wartość argumentu, dla której wartość funkcji wynosi 0,5), a parametr *a* decyduje o nachyleniu funkcji;

— funkcja dzwonowa

$$\mu(x) = \frac{1}{1+\left(\frac{x-c}{b}\right)^{2a}} \tag{4.4}$$

gdzie parametr *c* decyduje o położeniu funkcji (środek symetrii), a parametry *a* i *b* decydują o kształcie funkcji;

—funkcja Gaussa

$$\mu(x) = e^{-\left(\frac{x-c}{a}\right)^2} \tag{4.5}$$

gdzie parametr *c* decyduje o położeniu funkcji (środek symetrii), a parametr *a* decyduje o kształcie funkcji. Wygląd różnych typów funkcji przynależności został przedstawiony na rys. 4.2. Różniczkowalne funkcje przynależności są często stosowane w rozmytych sztucznych sieciach neuronowych. Sieci te można bowiem wówczas uczyć metodami gradientowymi.

4.1. Modele rozmyte Mamdaniego

Modele rozmyte mają najczęściej postać zestawów reguł:

Reguła *i*: jeśli $\underbrace{x_1 \text{ jest } X_1^i \text{ i} \ldots \text{i } x_n \text{ jest } X_n^i}_{\text{poprzednik}}$, to

$$\underbrace{y \text{ jest } Y^i}_{\text{następnik}}, \tag{4.6}$$

gdzie x_j, y – zmienne, X_j^i, Y^i – zbiory rozmyte. Powyższy model ma zarówno poprzedniki jak i następniki reguł rozmyte. Modele takie często są otrzymywane bazując na wiedzy eksperta [19]. Dzięki takiemu podejściu, można w stosunkowo łatwy sposób opracować regulator rozmyty, pod warunkiem znalezienia eksperta na podstawie wiedzy którego opracowana zostanie baza reguł. W celu obliczenia wartości wyjścia takiego modelu należy kolejno:

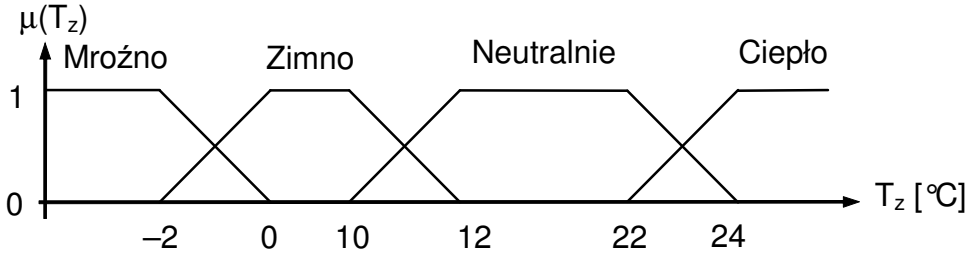
1. Obliczyć stopień przynależności wartości poszczególnych zmiennych do odpowiednich zbiorów rozmytych. Krok ten jest nazywany **fuzyfikacją** lub **rozmywaniem**.
2. Następnie na podstawie obliczonych stopni przynależności należy dla każdej reguły wyznaczyć **poziom aktywacji reguły** zwany również **siłą odpalenia reguły**.
3. Później określa się postać wynikowej funkcji przynależności zmiennej wyjściowej.
4. Na podstawie kształtu otrzymanej funkcji przynależności zmiennej wyjściowej dokonuje się **defuzyfikacji** (**wyostrzania**), w wyniku której otrzymana zostanie wartość wyjścia modelu.

Następniki reguł nie muszą być rozmyte. Często wykorzystywane do modelowania obiektów dynamicznych są modele z następnikami w postaci funkcji liniowych, w których

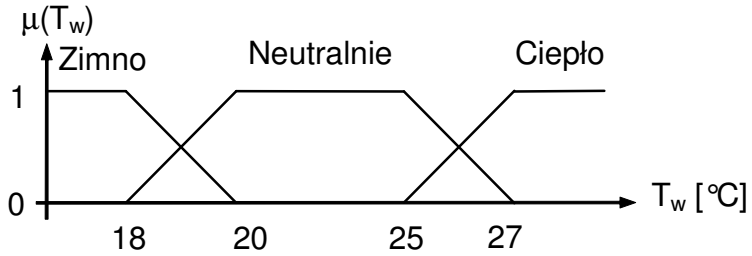
wyznaczanie wartości wyjścia modelu znacznie się upraszcza, zostały one opisane w rozdz. 4.3.

Przykład 4.1

Rozpatrzmy przypadek, gdy naszym zadaniem jest opracowanie regulatora temperatury wewnątrz budynku gospodarczego w gospodarstwie rolnym. Pod uwagę weźmiemy wartość temperatury na zewnątrz budynku T_z oraz wartość temperatury wewnątrz budynku T_w. Załóżmy, że wartość temperatury na zewnątrz może należeć do czterech zbiorów: wartości „Mroźnych", „Zimnych", „Neutralnych" i „Ciepłych" a funkcje przynależności do tych zbiorów są pokazane na rys. 4.3. Przyjmijmy także, że temperatura wewnątrz budynku może należeć do trzech zbiorów kiedy jest: „Zimno", „Neutralnie" i „Ciepło" a funkcje przynależności są pokazane na rys. 4.4.

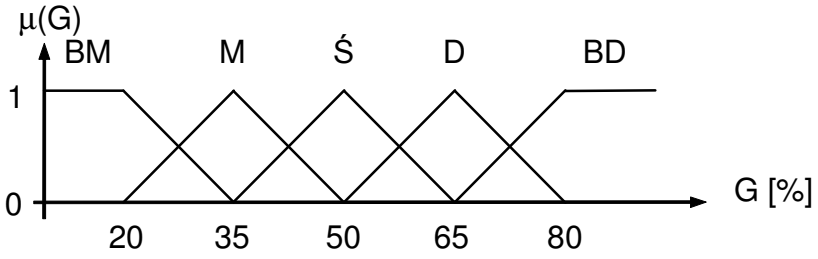


Rys. 4.3. Funkcje przynależności dla temperatury na zewnątrz budynku T_z



Rys. 4.4. Funkcje przynależności dla temperatury wewnątrz budynku T_w

Spróbujmy teraz sformułować zestaw reguł regulatora temperatury, przy czym załóżmy, że sterowanie G (grzanie) jest wyrażone w procentach, tzn. 100% oznacza maksymalny dopływ czynnika grzewczego. Przyjmijmy ponadto, że wartość grzania może należeć do pięciu zbiorów wartości: „Bardzo małych", „Małych", „Średnich", „Dużych" i „Bardzo dużych"; funkcje przynależności zostały pokazane na rys. 4.5.



Rys. 4.5. Funkcje przynależności dla grzania; zbiory wartości BM – bardzo małych, M – małych, Ś – średnich, D – dużych, BD – bardzo dużych

Częstą metodą przedstawiania bazy reguł jest **tablica decyzyjna**. Tego podejścia teraz użyjemy. Wystąpimy przy tym w roli eksperta, formułując zestaw reguł. Zostały one zebrane w tabl. 4.1. Każda komórka tej tablicy odpowiada jednej regule. Przyjmijmy na przykład, że na dworze jest zimno, a w budynku neutralnie. Odpowiednia reguła będzie więc następująca

(jest ona zaznaczona w tabl. 4.1):

Jeśli na dworze jest zimno i w budynku jest neutralnie, to grzanie jest duże.

Zauważmy, że reguła ta jest sformułowana w języku potocznym. Niemniej jednak wiadomo, jakie jest jej znaczenie. Bardziej formalnie można tę regułę zapisać jako:

Jeśli wartość T_z należy do zbioru „Zimno" i wartość T_w należy do zbioru „Neutralnie", to G należy do zbioru „Duże".

Wszystkie dwanaście reguł z tabl. 4.1 tworzy bazę reguł rozmytego regulatora temperatury. W celu łatwego odwoływania się do tych reguł, przyjmijmy, że będą one oznaczane pierwszymi literami nazw zbiorów kolejno temperatury T_z i T_w. Na przykład reguła podana powyżej będzie nazwana jako Reguła ZN.

Tablica 4.1. Tablica decyzyjna regulatora temperatury w budynku gospodarczym

$T_w \rightarrow$ / $T_z \downarrow$	Z	N	C
M	BD	BD	D
Z	BD	D	Ś
N	D	Ś	M
C	Ś	M	BM

Sposób przeprowadzania fuzyfikacji został omówiony na przykładzie w pierwszej części rozdziału. Przejdźmy więc do dokładniejszego omówienia drugiego punktu procedury obliczania wyjścia modelu rozmytego, tzn. wyznaczenia poziomów aktywacji reguł. Jeśli przesłanka reguły ma najprostszą postać, czyli:

$$Jeśli\ x_1\ jest\ X_1\,, \tag{4.7}$$

wtedy poziom aktywacji reguły jest równy wartości funkcji przynależności zmiennej x_1 do zbioru X_1, czyli

$$w_i = \mu_{X_1}(x_1). \tag{4.8}$$

Często jednak przesłanki są bardziej skomplikowane. Najczęściej, tak, jak we wzorze (4.6) lub w przykładzie 4.1, w przesłankach używa się operatora koniunkcji. Tego typu przesłanka złożona z dwóch przesłanek prostych jest następująca:

$$Jeśli\ x_1\ jest\ X_1\ i\ x_2\ jest\ X_2\,. \tag{4.9}$$

W takiej sytuacji poziom aktywacji reguły można obliczyć używając różnych metod. Ich wspólną cechą jest jednak to, że jeśli któraś z wartości funkcji przynależności jest równa 0, wtedy poziom aktywacji reguły jest równy $w_i = 0$. Natomiast, gdy wszystkie wartości funkcji przynależności są równe 1, to poziom aktywacji reguły jest równy $w_i = 1$. Najczęściej, w przypadku konkluzji z koniunkcją, poziom aktywacji reguły wyznacza się używając funkcji minimum i wtedy:

$$w_i = \min\{\mu_{X_1}(x_1), \mu_{X_2}(x_2)\} \tag{4.10}$$

lub oblicza się ją jako iloczyn wartości funkcji przynależności, wówczas:

$$w_i = \mu_{X_1}(x_1) \cdot \mu_{X_2}(x_2). \tag{4.11}$$

Inne metody obliczania poziomów aktywacji reguł w przypadku użycia operatora koniunkcji można znaleźć np. w pracy [31].

W przypadku użycia operatora alternatywy, co zdarza się rzadko, przesłanka złożona z dwóch przesłanek prostych jest następująca:

$$Jeśli\ x_1\ jest\ X_1\ lub\ x_2\ jest\ X_2 \tag{4.12}$$

W takim przypadku, poziom aktywacji reguły oblicza się najczęściej używając funkcji maksimum, czyli:

$$w_i = \max\left\{\mu_{X_1}(x_1), \mu_{X_2}(x_2)\right\}. \tag{4.13}$$

Zauważmy, że w takiej sytuacji, poziom aktywacji reguły może być równy $w_i = 0$ tylko, jeśli wszystkie wartości funkcji przynależności są równe 0 a będzie równy $w_i = 1$ jeśli przynajmniej jedna z wartości funkcji przynależności jest równa 1.

Przykład 4.2

Prześledźmy teraz metodę wyznaczania poziomów aktywacji reguł na przykładzie regulatora temperatury z przykł. 4.1. Rozpatrzymy trzy przypadki.

Przypadek I

Najpierw załóżmy, że temperatura na zewnątrz budynku T_{z1} = 5 °C a temperatura wewnątrz wynosi T_{w1} = 21 °C. W takiej sytuacji, wartość T_{z1} należy tylko do zbioru wartości zimnych ze stopniem przynależności $\mu_Z(T_{z1}) = 1$. Wartość T_{w1} – tylko do zbioru wartości neutralnych ze stopniem przynależności $\mu_N(T_{w1}) = 1$. W takim razie aktywna (poziom aktywacji większy od 0) będzie tylko jedna reguła (Reguła ZN) a poziom aktywacji tej reguły wyniesie w_{ZN} = 1 (dla obydwu przedstawionych sposobów przeprowadzania koniunkcji).

Przypadek II

Tym razem przyjmijmy, że temperatura na zewnątrz budynku wynosi T_{z2} = 11 °C a temperatura wewnątrz wynosi tyle, co poprzednio, czyli T_{w1} = 21 °C. W takim razie, wartość temperatury T_{z2} należy do zbioru wartości zimnych ze stopniem przynależności $\mu_Z(T_{z2}) = 0{,}5$ i do zbioru wartości neutralnych ze stopniem przynależności $\mu_N(T_{z2}) = 0{,}5$ (a do pozostałych zbiorów wartości ze stopniami przynależności $\mu_M(T_{z2}) = \mu_C(T_{z2}) = 0$). Tym razem więc, aktywne będą dwie reguły (ZN oraz NN) a poziomy ich aktywacji będą wynosiły (zarówno przy wykorzystaniu funkcji minimum jak i iloczynu do obliczenia koniunkcji) w_{ZN} = 0,5 oraz w_{NN} = 0,5.

Przypadek III

Tym razem przyjmijmy, że temperatura na zewnątrz budynku wynosi T_{z2} = 11 °C (jak w przypadku II) ale temperatura wewnątrz wynosi T_{w2} = 25,5 °C. W takim razie, wartość temperatury T_{w2} należy do zbioru wartości neutralnych ze stopniem przynależności $\mu_N(T_{w2}) = 0{,}75$ i do zbioru wartości ciepłych ze stopniem przynależności $\mu_C(T_{w2}) = 0{,}25$ (a do zbioru wartości zimnych ze stopniem przynależności $\mu_Z(T_{w2}) = 0$). W takim razie, w tym przypadku aktywne będą cztery reguły (ZN, ZC, NN i NC). Poziomy aktywacji tych reguł będą wynosiły przy wykorzystaniu funkcji minimum:

a) w_{ZN} = 0,5, w_{ZC} = 0,25, w_{NN} = 0,5 i w_{NC} = 0,25;

a w przypadku użycia iloczynu do obliczenia koniunkcji:

b) w_{ZN} = 0,375, w_{ZC} = 0,125, w_{NN} = 0,375 i w_{NC} = 0,125.

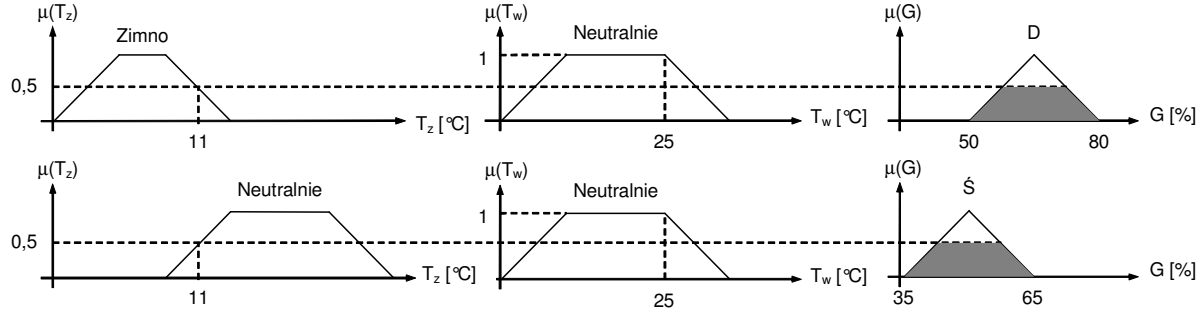
Kolejnym etapem przy wyznaczaniu wartości wyjścia modelu rozmytego jest otrzymanie wynikowej postaci funkcji przynależności zmiennej wyjściowej. Najczęściej czyni się to używając **operatora implikacji Mamdaniego**. Wówczas wynikowa funkcja przynależności następnika (otrzymana przy uwzględnieniu poziomu aktywacji reguły) jest opisana wzorem:

$$\mu^{*}_{X^{i}_{n+1}}(x_{n+1}) = \min\left\{w_i, \mu_{X^{i}_{n+1}}(x_{n+1})\right\} . \tag{4.14}$$

Najprościej wyjaśnić działanie operatora implikacji Mamdaniego na przykładzie.

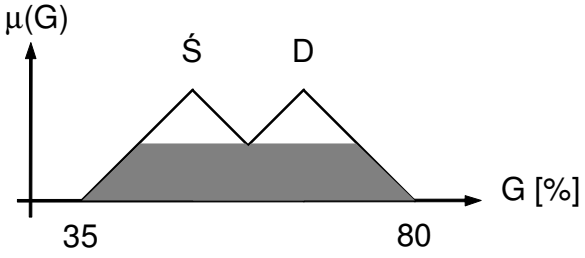
Przykład 4.3

Powróćmy do przypadku II z przykładu 4.2. Prześledźmy działanie operatora implikacji Mamdaniego na rys. 4.6 (oraz funkcji minimum w roli operatora koniunkcji). Otrzymane w wyniku przeprowadzonej operacji zmodyfikowane funkcje przynależności następników aktywnych reguł oznaczone zostały szarym kolorem.



Rys. 4.6. Wyznaczanie wynikowej postaci funkcji przynależności zmiennej wyjściowej przy użyciu operatora implikacji Mamdaniego

Następnym krokiem jest **akumulacja** otrzymanych zmodyfikowanych funkcji przynależności następników aktywnych reguł do wynikowej funkcji przynależności bazy reguł, co w rozważanym przykładzie da wynik przedstawiony na rys. 4.7.

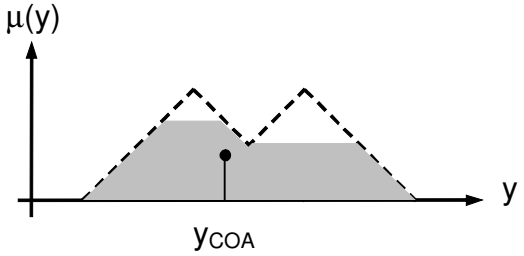


Rys. 4.7. Wynikowa funkcja przynależności bazy reguł

Ostatnim etapem procedury otrzymywania wartości wyjścia modelu rozmytego jest defuzyfikacja. W wyniku jej wykonania otrzymuje się szukaną wartość wyjścia modelu rozmytego. Sposobów wykonywania defuzyfikacji jest wiele. Najczęściej stosowane z tych sposobów, które zostaną omówione szczegółowo dalej to:

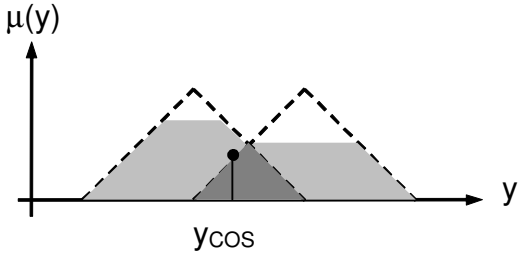
1. metoda środka przedziału,
2. metoda środka sum,
3. metoda środka maksimum,
4. metoda pierwszego maksimum,
5. metoda ostatniego maksimum.

Ad. 1. Metoda środka obszaru (Center of Area – COA) nazywana też metodą środka ciężkości (ang. Center of Gravity – COG) polega na wyznaczeniu środka ciężkości powierzchni pod wynikową funkcją przynależności bazy reguł. Sposób jej użycia został schematycznie pokazany na rys. 4.8.



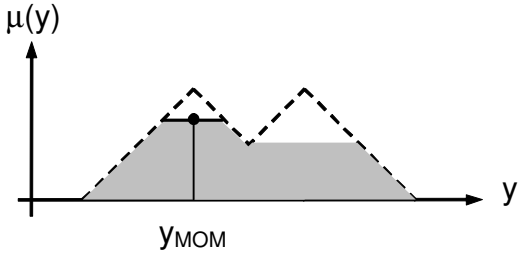
Rys. 4.8. Defuzyfikacja metodą środka obszaru (COA)

Ad. 2. Metoda środka sum (ang. Center of Sums – COS) jest podobna do poprzedniej i polega na wyznaczeniu środka ciężkości powierzchni pod wynikową funkcją przynależności bazy reguł ale z uwzględnieniem obszarów wspólnych tyle razy, ile razy występują. Na rys. 4.9 przedstawiono schematycznie sposób jej użycia. Obszar uwzględniany podwójnie został oznaczony kolorem ciemnoszarym. Ze względu na konieczność jego podwójnego uwzględnienia, wartość y_{COS} jest przesunięta na prawo względem wartości y_{COA} otrzymanej poprzednio omówioną metodą.



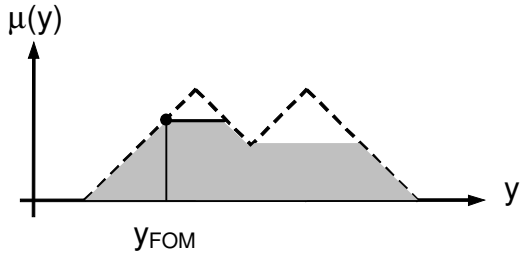
Rys. 4.9. Defuzyfikacja metodą środka sum (COS)

Ad. 3. Metoda środka maksimum (ang. Mean of Maxima – MOM) polega na wybraniu wartości ze środka przedziału wartości, dla których wartość wynikowej funkcji przynależności bazy reguł przyjmuje największe wartości. Działanie tej metody ilustruje rys. 4.10.



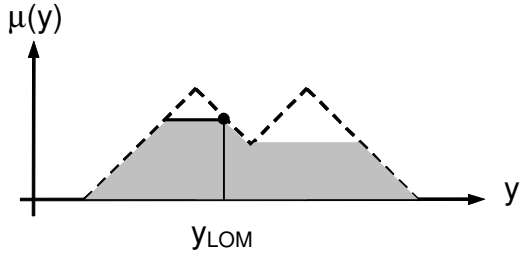
Rys. 4.10. Defuzyfikacja metodą środka maksimum (MOM)

Ad. 4. Metoda pierwszego maksimum (ang. First of Maxima – FOM) polega na wybraniu wartości z początku przedziału wartości, dla których wartość wynikowej funkcji przynależności bazy reguł przyjmuje największe wartości. Działanie tej metody ilustruje rys. 4.11.



Rys. 4.11. Defuzyfikacja metodą pierwszego maksimum (FOM)

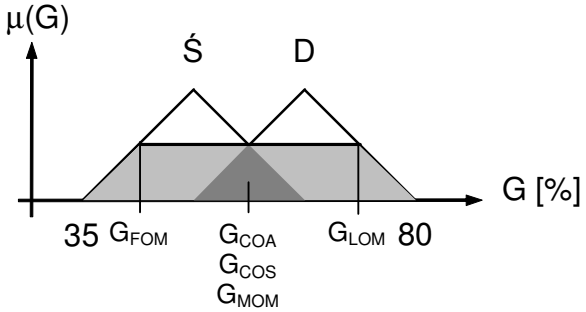
Ad. 5. Metoda ostatniego maksimum (ang. Last of Maxima – LOM) polega na wybraniu wartości z końca przedziału wartości, dla których wartość wynikowej funkcji przynależności bazy reguł przyjmuje największe wartości. Działanie tej metody ilustruje rys. 4.12.



Rys. 4.12. Defuzyfikacja metodą ostatniego maksimum (LOM)

Przykład 4.4

Powróćmy do przykładu 4.3 i przeprowadźmy defuzyfikację dla tam otrzymanej wynikowej funkcji przynależności bazy reguł. Wynik tego zabiegu przedstawiono na rys. 4.13. Mamy tu do czynienia z interesującym przypadkiem, ponieważ otrzymana wynikowa funkcja przynależności bazy reguł jest symetryczna. W związku z tym wartości wyjścia otrzymane pierwszymi trzema metodami defuzyfikacji są takie same i wynoszą $G_{COA}=G_{COS}=G_{MOM}=57{,}5$ %. Różnica pojawia się w przypadku zastosowania metody pierwszego maksimum ($G_{FOM}=42{,}5$ %) oraz metody ostatniego maksimum ($G_{LOM}=72{,}5$ %).



Rys. 4.13. Wynikowa funkcja przynależności bazy reguł w przykładzie 4.4

4.2. Regulatory rozmyte typu Mamdaniego

Równanie regulatora, w ogólnej postaci można zapisać jako:

$$u(k) = f\left(x_1,\ldots,x_n\right), \tag{4.15}$$

gdzie $u(k)$ jest szukanym sterowaniem zaś zmienne x_i można dobrać stosownie do potrzeb i mogą być one w szczególności: zmiennymi stanu procesu, uchybem w bieżącej i przeszłych chwilach (e_k, e_{k-1}, e_{k-2}, …), sterowaniem w przeszłych chwilach (u_{k-1}, u_{k-2}, …). W przypadku regulatora w postaci przyrostowej (generującego przyrost sterowania), otrzymamy:

$$\Delta u(k) = g(x_1,\ldots,x_n). \tag{4.16}$$

Jeśli funkcja f: $R^n \rightarrow R$ (lub g: $R^n \rightarrow R$) jest liniowa, wówczas mamy do czynienia z regulatorem liniowym. Przykłady takich regulatorów zostały opisane w rozdz. 2. W celu poprawy jakości regulacji możemy także założyć, że funkcja ta jest nieliniowa i opisana modelem rozmytym typu Mamdaniego [6, 42]. Zauważmy, że tak właśnie uczyniono w przykładzie 4.1 i przyjęto, że $G(k) = f\left(T_z(k), T_w(k)\right)$. Rozważmy teraz, na zasadzie analogii do ich wersji liniowych, odpowiedniki rozmyte typu Mamdaniego regulatorów P, PI, PD oraz PID przypomnianych w zarysie w rozdz. 2.

Regulator rozmyty P

Przypomnijmy, że w przypadku proporcjonalnego regulatora liniowego, prawo regulacji ma postać:

$$u(k) = f_{L\_P}(e(k)) = K \cdot e(k), \tag{4.17}$$

gdzie K jest parametrem dostrajalnym regulatora a f_{L_P}:$R \rightarrow R$ jest funkcją liniową zależną od bieżącej wartości uchybu regulacji. W takim razie równanie rozmytego regulatora typu P będzie miało postać:

$$u(k) = f_P(e(k)), \tag{4.18}$$

gdzie f_P:$R \rightarrow R$ jest funkcją nieliniową o właściwościach zależnych od kształtu funkcji przynależności przyjętych przez projektanta regulatora oraz od bazy reguł, która będzie złożona z reguł o następującej postaci:

Reguła i: jeśli $e(k)$ jest E_i, to

$$u(k) \text{ jest } U_i, \tag{4.19}$$

gdzie E_i, U_i (i = 1,…, l) są zbiorami rozmytymi. Regulator ten jest stosunkowo mało skomplikowany. Zauważmy bowiem, że w przypadku proporcjonalnego regulatora rozmytego, liczba reguł będzie równa liczbie zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego uchybu $e(k)$. Niestety w przypadku rozważanych regulatorów rozmytych, każde rozbudowanie regulatora polegające na dodaniu kolejnej zmiennej pociąga za sobą znaczne rozbudowanie bazy reguł oraz komplikację samych reguł, co zostanie za chwilę zademonstrowane.

Regulator rozmyty PI

W przypadku liniowego regulatora PI, prawo regulacji ma postać:

$$u(k) = f_{L\_PI}(e(k), e(k-1), u(k-1)) = u(k-1) + r_1 \cdot e(k) + r_2 \cdot e(k-1), \tag{4.20}$$

gdzie f_{L_PI}:$R^3 \rightarrow R$ jest funkcją liniową a r_1 i r_2 są parametrami dostrajalnymi regulatora. Powyższe prawo regulacji można zapisać w postaci przyrostowej jako:

$$\Delta u(k) = r_1^p \cdot e(k) + r_2^p \cdot \Delta e(k), \tag{4.21}$$

gdzie r_1^p, r_2^p są parametrami regulatora przyrostowego PI, a $\Delta e(k) = e(k) - e(k-1)$. W takim razie, równanie rozmytego regulatora typu PI będzie następujące:

$$\Delta u(k) = g_{PI}(e(k), \Delta e(k)), \tag{4.22}$$

gdzie g_{PI}:$R^2 \rightarrow R$ jest funkcją nieliniową. Baza reguł regulatora, będzie więc złożona z następujących reguł:

Reguła *i*: jeśli $e(k)$ jest E_i i $\Delta e(k)$ jest D_i, to

$$\Delta u(k) \text{ jest } U_i, \tag{4.23}$$

gdzie E_i, D_i, U_i (i = 1,…, *l*) są zbiorami rozmytymi. Ze względu na to, że regulator generuje przyrosty sterowania zamiast jego wartości, w celu wyznaczenia tejże wartości należy skorzystać ze wzoru:

$$u(k) = u(k-1) + \Delta u(k). \tag{4.24}$$

Porównajmy otrzymany rozmyty regulator PI z rozmytym regulatorem proporcjonalnym. Zauważmy, że komplikacji uległy poprzedniki reguł. Ponadto liczba tych reguł będzie tym razem równa iloczynowi liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego uchybu $e(k)$ oraz liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego przyrostu uchybu $\Delta e(k)$.

Regulator rozmyty PD

Podobnie skomplikowany, co rozmyty regulator PI, jest rozmyty regulator PD. W liniowym regulatorze PD, prawo regulacji ma postać:

$$u(k) = f_{L\_PD}(e(k), \Delta e(k)) = r_1 \cdot e(k) + r_2 \cdot \Delta e(k), \tag{4.25}$$

gdzie f_{L_PD}:$R^2 \rightarrow R$ jest funkcją liniową a r_1 i r_2 są parametrami dostrajalnymi regulatora. Równanie rozmytego regulatora typu PD będzie więc opisane wzorem:

$$u(k) = f_{PD}(e(k), \Delta e(k)), \tag{4.26}$$

gdzie f_{PD}:$R^2 \rightarrow R$ jest funkcją nieliniową. Baza reguł rozmytego regulatora PD, będzie złożona z reguł podobnych do tych z rozmytego regulatora PI z tą różnicą, że w następniku każdej reguły wystąpi wartość sterowania zamiast jego przyrostu. Każda z reguł będzie więc miała następującą postać:

Reguła *i*: jeśli $e(k)$ jest E_i i $\Delta e(k)$ jest D_i, to

$$u(k) \text{ jest } U_i, \tag{4.27}$$

gdzie E_i, D_i, U_i (i = 1,…, *l*) są zbiorami rozmytymi. W rozmytym regulatorze PD liczba reguł będzie więc równa iloczynowi liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego uchybu $e(k)$ oraz liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego przyrostu uchybu $\Delta e(k)$; tak samo, jak w rozmytym regulatorze PI. Niestety chęć zastosowania bardziej zaawansowanego, rozmytego regulatora typu PID pociąga za sobą konieczność dalszego znacznego skomplikowania regulatora.

Regulator rozmyty PID

Prawo regulacji liniowego regulatora PID w wersji przyrostowej jest następujące:

$$\Delta u(k) = g_{L\_PID}(e(k), e(k-1), e(k-2)) = r_1 \cdot e(k) + r_2 \cdot e(k-1) + r_3 \cdot e(k-2), \qquad (4.28)$$

gdzie g_{L_PID}:$R^3 \rightarrow R$ jest funkcją liniową a r_1, r_2 i r_3 są parametrami regulatora. W takim razie, równanie rozmytego regulatora typu PID w wersji przyrostowej będzie następujące:

$$\Delta u(k) = g_{PID}(e(k), e(k-1), e(k-2)), \qquad (4.29)$$

gdzie g_{PID}:$R^3 \rightarrow R$ jest funkcją nieliniową. Baza reguł rozmytego regulatora PID, będzie więc złożona z następujących reguł:

Reguła *i*: jeśli $e(k)$ jest $E0_i$ i $e(k-1)$ jest $E1_i$ i $e(k-2)$ jest $E2_i$, to

$$\Delta u(k) \text{ jest } U_i, \qquad (4.30)$$

gdzie $E0_i$, $E1_i$, $E2_i$, U_i (i = 1,…, *l*) są zbiorami rozmytymi. W przedstawionym rozmytym regulatorze PID liczba reguł będzie więc równa iloczynowi liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności bieżącego uchybu $e(k)$, liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności uchybu w chwili poprzedniej $e(k–1)$, oraz liczby zbiorów rozmytych, na które podzielono przestrzeń zmienności uchybu sprzed dwóch chwil próbkowania $e(k–2)$.

Możliwe jest także opracowanie regulatora PID bezpośrednio generującego wartość (a nie przyrost) sterowania. Rozpatrzmy liniowy regulator PID w następującej postaci:

$$u(k) = f_{L\_PID}(e(k), \sum e(k), \Delta e(k)) = k_p \cdot \left(e(k) + \frac{1}{T_i} \cdot \sum e(k) + T_d \cdot \Delta e(k) \right), \qquad (4.31)$$

gdzie f_{L_PID}:$R^3 \rightarrow R$ jest funkcją liniową a k_p, T_i i T_d są parametrami regulatora, $\sum e(k)$ jest sumą przeszłych uchybów, która jest traktowana jako kolejna zmienna. W takim razie prawo regulacji rozmytego regulatora PID będzie miało postać:

$$u(k) = f_{PID}(e(k), \sum e(k), \Delta e(k)), \qquad (4.32)$$

gdzie f_{PID}:$R^3 \rightarrow R$ jest funkcją nieliniową. Baza reguł rozmytego regulatora PID, będzie tym razem złożona z następujących reguł:

Reguła *i*: jeśli $e(k)$ jest E_i i $\sum e(k)$ jest S_i i $\Delta e(k)$ jest D_i, to

$$u(k) \text{ jest } U_i, \qquad (4.33)$$

gdzie E_i, S_i, D_i, U_i (i = 1,…, *l*) są zbiorami rozmytymi. Zauważmy, że zmianie w stosunku do wersji przyrostowej uległy zmienne wykorzystane w poprzednikach reguł a w następniku występuje wartość sterowania.

Możliwe jest dalsze rozbudowywanie regulatorów rozmytych przez wykorzystanie kolejnych zmiennych. Niestety komplikacji ulega wówczas proces doboru parametrów regulatora rozmytego i tak już bardzo skomplikowany w przypadku rozmytego regulatora PID. Zauważmy bowiem, że jeśli przestrzeń każdej ze zmiennych użytych w przesłankach reguł będzie podzielona na trzy zbiory rozmyte, wtedy reguł w regulatorze pojawi się $l = 3^3 = 27$. Na szczęście skutków tego przekleństwa wymiarowości można w znacznym stopniu uniknąć korzystając z innego typu modeli rozmytych – modeli Takagi–Sugeno.

4.3. Modele rozmyte Takagi–Sugeno

Jedną z odmian modeli rozmytych, które okazały się bardzo skuteczne w modelowaniu obiektów regulacji są modele typu Takagi–Sugeno [37] (czasami nazywane także modelami Takagi–Sugeno–Kanga). Modele te są złożone z reguł, w następnikach których używa się funkcji. Ich zaletą jest możliwość opisania zachowania obiektu za pomocą stosunkowo niewielkiej liczby reguł. Ogólna postać modeli Takagi–Sugeno jest więc następująca:

Reguła *i*: jeśli $\underbrace{x_1 \text{ jest } X_1^i \text{ i} \ldots \text{i } x_n \text{ jest } X_n^i}_{\text{poprzednik}}$, to

$$\underbrace{y^i = f^i(x_1,\ldots,x_n)}_{\text{następnik}}, \tag{4.34}$$

gdzie y^i są wyjściami następników. Najczęściej w następnikach używa się funkcji liniowych. Mają więc one postać:

$$y^i = \sum_{j=1}^{n} a_j^i \cdot x_j + a_0^i, \tag{4.35}$$

gdzie a_j^i ($j = 0,\ldots,n$, $i = 1,\ldots,l$) są parametrami modelu, l jest liczbą reguł z których złożony jest model rozmyty. Zastosowanie następników liniowych upraszcza model. Co więcej, łatwo jest dokonać identyfikacji ich parametrów korzystając z dobrze znanych metod. Następniki reguł, ponieważ mogą być interpretowane jako modele opisujące zachowanie obiektu wokół pewnych punktów pracy, są nazywane modelami lokalnymi.

W celu obliczenia wartości wyjściowej modelu rozmytego Takagi–Sugeno, należy skorzystać z następującego wzoru:

$$y = \frac{\sum_{i=1}^{l} w_i \cdot y^i}{\sum_{i=1}^{l} w_i}, \tag{4.36}$$

gdzie w_i ($i = 1,\ldots,l$) są siłami odpalenia poszczególnych reguł. Wyjście modelu jest więc sumą ważoną wyjść poszczególnych modeli lokalnych. Obliczenie wartości wyjścia modelu typu Takagi–Sugeno jest więc prostsze niż w przypadku modelu z rozmytymi następnikami.

Przykład 4.5

Rozpatrzmy przykład rozmytego modelu charakterystyki statycznej zaworu przeznaczonego do regulacji przepływu cieczy. Charakterystyka ta jest opisana następującym wzorem [1]:

$$y = \frac{0{,}3163 \cdot u}{\sqrt{0{,}1 + 0{,}9 \cdot u^2}}, \tag{4.37}$$

gdzie wyjście y jest przepływem przez zawór, a wejście u jest pozycją trzpienia zaworu. Charakterystyka ta jest przedstawiona na rys. 4.14 linią różową. Zauważmy, że kształt tej charakterystyki przypomina funkcję sigmoidalną. Spróbujmy użyć modelu rozmytego do zamodelowania tej charakterystyki. Można przy tym posłużyć się doborem parametrów modelu wspomaganym komputerowo. Po takim zabiegu, otrzymano funkcje przynależności pokazane na rys. 4.15. Model rozmyty jest złożony z następujących dwóch reguł [22]:

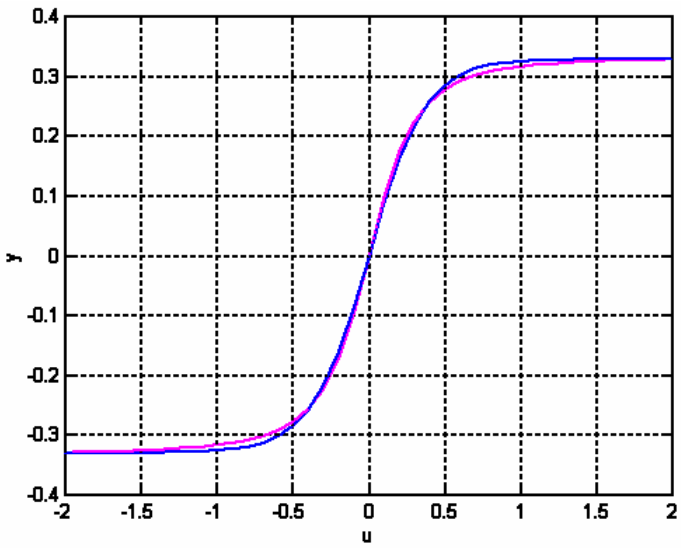
Reguła 1: jeśli u jest R_1, to

$$y = -0{,}3289,$$

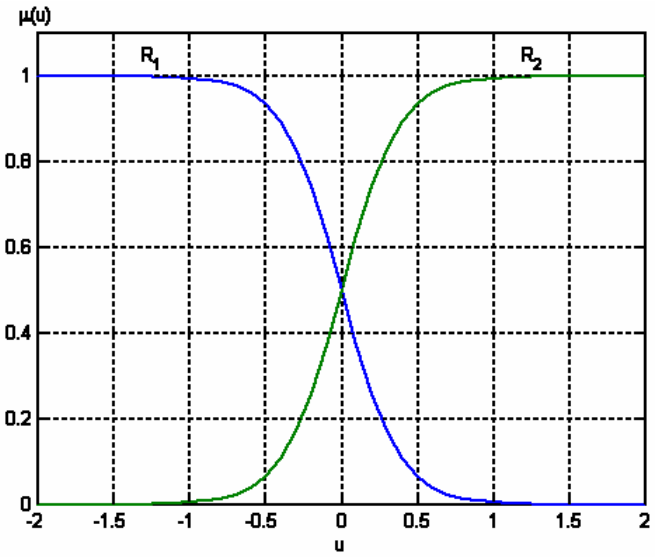
Reguła 2: jeśli u jest R_2, to

$$y = 0{,}3289.$$

Model ten choć prosty, zaskakująco dobrze oddaje modelowaną nieliniowość (linia niebieska na rys. 4.14).



Rys. 4.14. Charakterystyka statyczna zaworu regulacyjnego; oryginalna – linia różowa, z modelu rozmytego – linia niebieska



Rys. 4.15. Funkcje przynależności w modelu rozmytym charakterystyki statycznej zaworu regulacyjnego

Przykład 4.6

Rozpatrzmy teraz przypadek dynamicznego modelu typu Takagi–Sugeno. Załóżmy, że jest to model dyskretny o jednym wejściu i jednym wyjściu, wykorzystujący wartości sygnałów wejściowego i wyjściowego procesu w przeszłości oraz równania różnicowe w następnikach. W takim razie model ten jest złożony z następujących reguł:

Reguła *i*: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = b_1^i \cdot y_k + \ldots + b_n^i \cdot y_{k-n+1} + c_1^i \cdot u_k + \ldots + c_m^i \cdot u_{k-m+1}, \tag{4.38}$$

gdzie $b_1^i,\ldots,b_n^i,c_1^i,\ldots,c_m^i$ ($i = 1,\ldots,l$) są współczynnikami modeli lokalnych, y_k jest wartością wyjścia obiektu regulacji w chwili k, u_k jest wartością wejścia w chwili k, $B_1^i,\ldots,B_n^i,C_1^i,\ldots,C_m^i$ są zbiorami rozmytymi. Zauważmy, że modele lokalne są liniowe (najczęściej takie są używane w praktyce) i mogą być zidentyfikowane na podstawie próbek zarejestrowanych w okolicach kilku punktów pracy podczas eksperymentów prowadzonych na realnym obiekcie (często stosowane podejście).

Powróćmy teraz do ogólnej postaci modeli typu Takagi–Sugeno. Zwróćmy uwagę na to, że zastosowanie wzoru (4.36) jest równoważne obliczaniu następującej sumy:

$$y = \sum_{j=1}^{n} \tilde{a}_j \cdot x_j + \tilde{a}_0 , \tag{4.39}$$

$$\tilde{a}_j = \frac{\sum_{i=1}^{l} w_i \cdot a_j^i}{\sum_{i=1}^{l} w_i},$$

gdzie $\tilde{a}_j$ jest sumą ważoną odpowiednich parametrów modeli lokalnych. W celu uproszczenia zapisu zwykle wprowadza się wagi znormalizowane, tzn.:

$$\tilde{w}_i = \frac{w_i}{\sum_{i=1}^{l} w_i}. \tag{4.40}$$

Przykład 4.7

Wróćmy do poprzedniego przykładu. Wyjście rozmytego modelu dynamicznego będzie w takim razie opisane następującą zależnością:

$$y_{k+1} = \tilde{b}_1 \cdot y_k + \ldots + \tilde{b}_n \cdot y_{k-n+1} + \tilde{c}_1 \cdot u_k + \ldots + \tilde{c}_m \cdot u_{k-m+1}, \tag{4.41}$$

gdzie

$$\tilde{b}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot b_j^i , \; \tilde{c}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot c_j^i .$$

Tego typu model można więc traktować jako model liniowy z parametrami zmiennymi w czasie. Dlatego też modele Takagi–Sugeno nazywa się czasem modelami quasi–liniowymi.

4.3.1. Przedstawienie modelu Takagi–Sugeno w postaci sieci neuronowej

Modele rozmyte można przedstawić w postaci rozmytych sieci neuronowych (ang. Fuzzy Neural Networks – FNN), zob. np. [29, 31, 38]. Można z nich skorzystać w celu identyfikacji modeli rozmytych. Przypomnijmy ogólną postać modeli rozmytych typu Takagi–Sugeno. Model takiego typu jest złożony z zestawu następujących reguł (przy założeniu następników opisanych funkcją liniową):

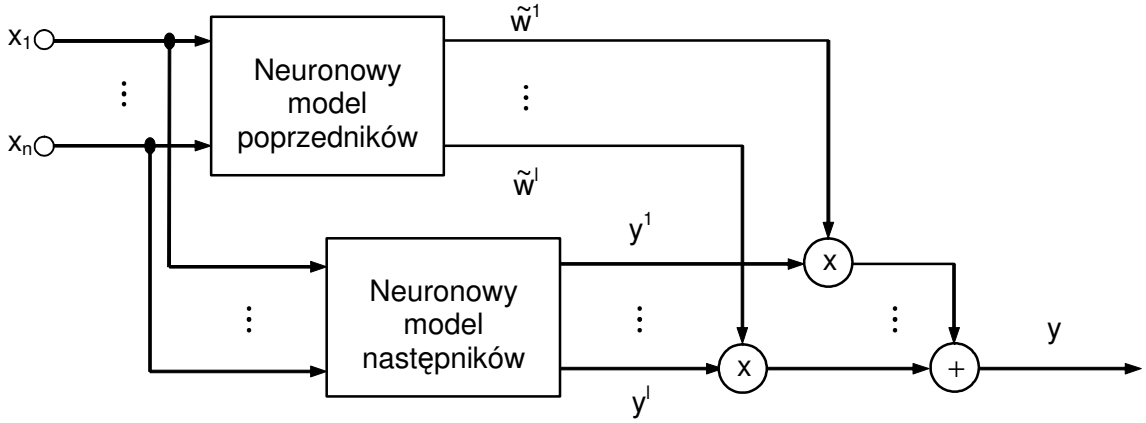
Reguła *i*: jeśli x_1 jest X_1^i i … i x_n jest X_n^i , to

$$y^i = \sum_{j=1}^{n} a_j^i \cdot x_j + a_0^i . \tag{4.42}$$

Wyjście modelu rozmytego typu Takagi–Sugeno jest z kolei dane wzorem:

$$y = \sum_{i=1}^{l} \tilde{w}_i \cdot y^i , \tag{4.43}$$

gdzie $\tilde{w}_i$ są znormalizowanymi wagami. W takim razie, ogólną strukturę rozmytej sieci neuronowej odzwierciedlającej model rozmyty z wyjściem opisanym równaniem (4.43) można przedstawić, jak na rys. 4.16.



Rys. 4.16. Ogólna struktura rozmytej sieci neuronowej

Neuronowy model poprzedników

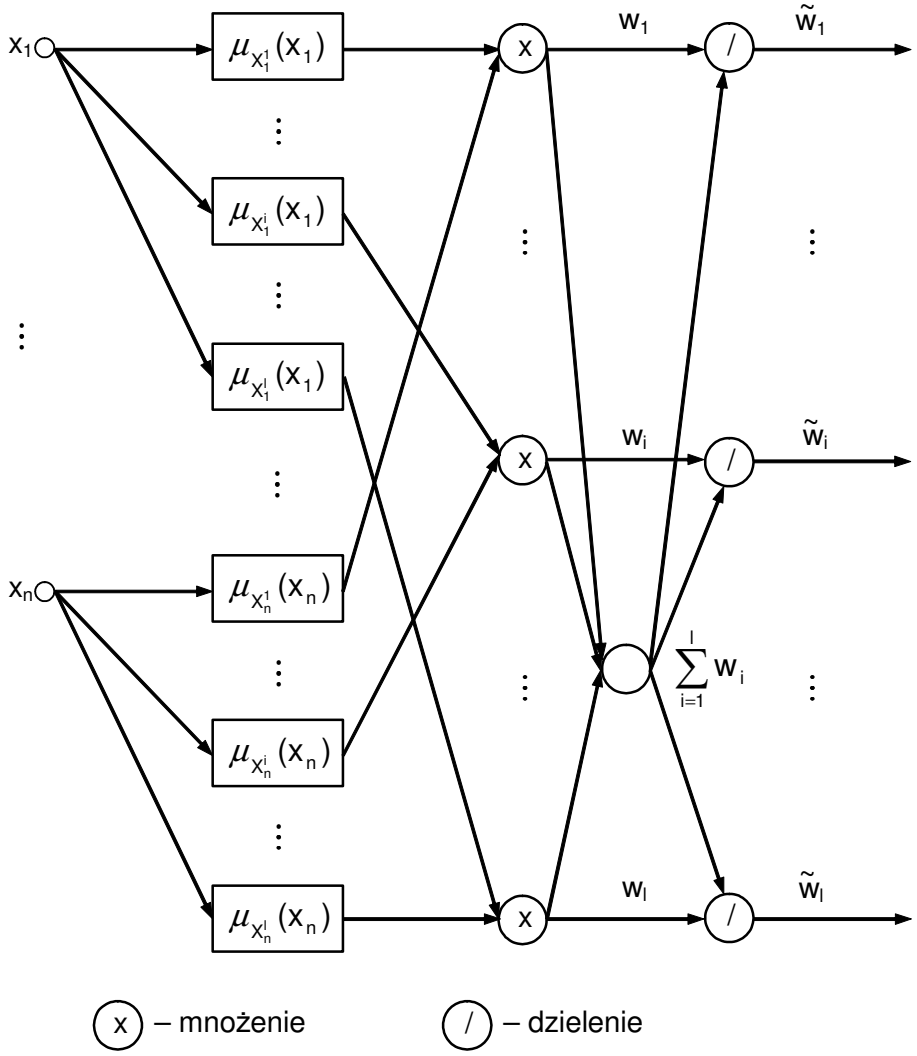
Przypomnijmy, że znormalizowane wagi są opisane wzorem:

$$\tilde{w}_i = \frac{w_i}{\sum_{i=1}^{l} w_i}, \tag{4.44}$$

gdzie poszczególne wagi w_i, przy założeniu, że jako operatora koniunkcji użyto mnożenia, są iloczynem wartości funkcji przynależności:

$$w_i = \prod_{j=1}^{n} \mu_{X_j^i}(x_j) . \tag{4.45}$$

W takim razie struktura neuronowego modelu poprzedników będzie miała postać taką, jak na rys. 4.17.



Rys. 4.17. Neuronowy model poprzedników rozmytego modelu Takagi–Sugeno

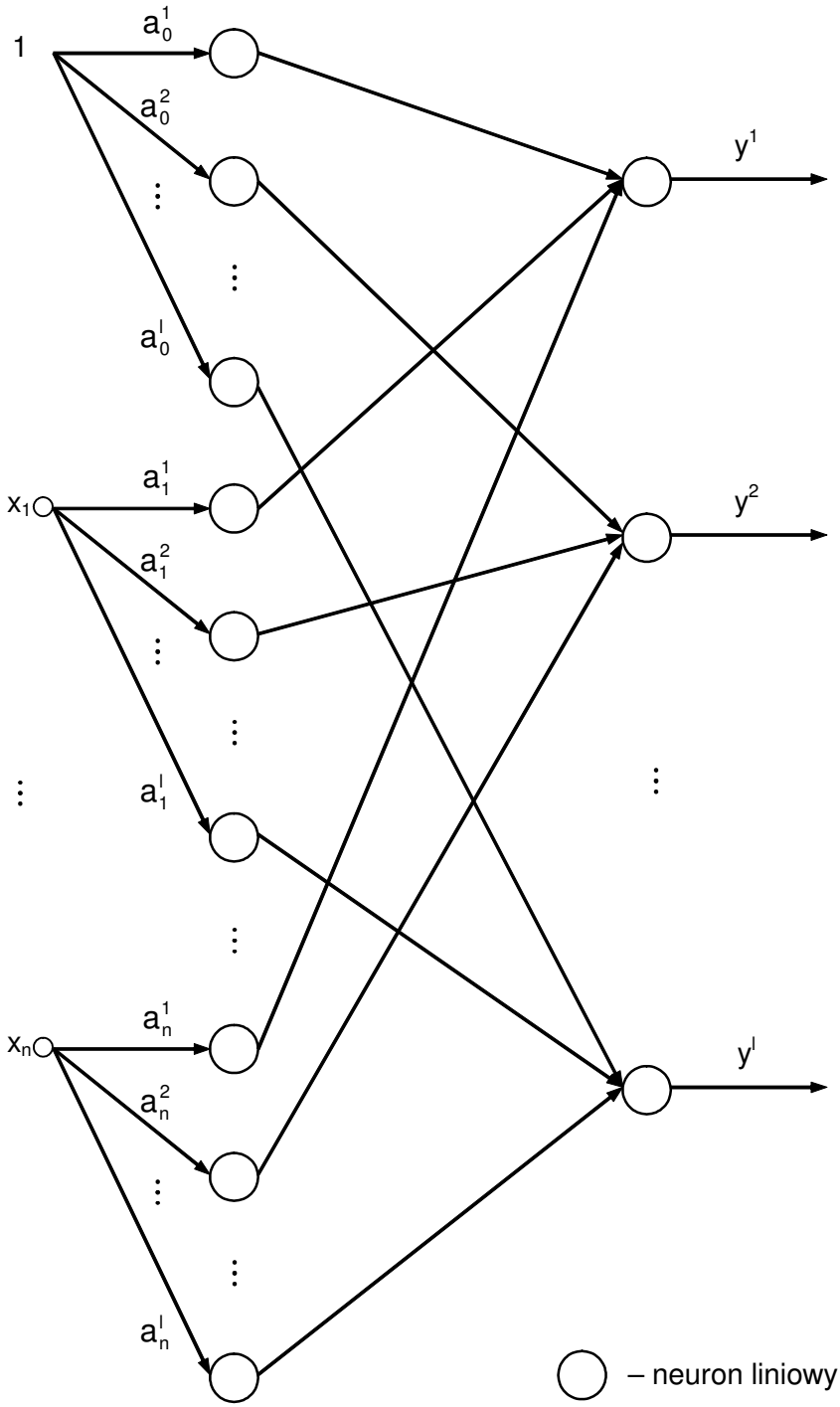
Zauważmy, że w powyższym modelu neuronowym poprzedników, uczeniu podlegają parametry funkcji przynależności $\mu_{X_j^i}(x_j)$. Funkcje te zostały oznaczone elementami prostokątnymi, ponieważ są to elementy bardziej złożone, które można byłoby przedstawić za pomocą prostszych neuronów. Nie jest to jednak konieczne do przeprowadzenia dalszych rozważań.

Neuronowy model następników

Przypomnijmy postać następników poszczególnych reguł w rozważanym modelu rozmytym:

$$y^i = \sum_{j=1}^{n} a_j^i \cdot x_j + a_0^i;\ i = 1,\ldots, l.$$

Zauważmy że jest to zależność liniowa. W takim razie struktura następników reguł rozważanego modelu Takagi–Sugeno może być przedstawiona jako sztuczna sieć neuronowa z rys. 4.18. W tym modelu neuronowym, uczeniu podlegają parametry a_j^i funkcji liniowych występujących w następnikach reguł, z których jest złożony model rozmyty.



Rys. 4.18. Neuronowy model następników rozmytego modelu Takagi–Sugeno

Przykład 4.8 (rozmyta sieć neuronowa Wanga–Mendela)

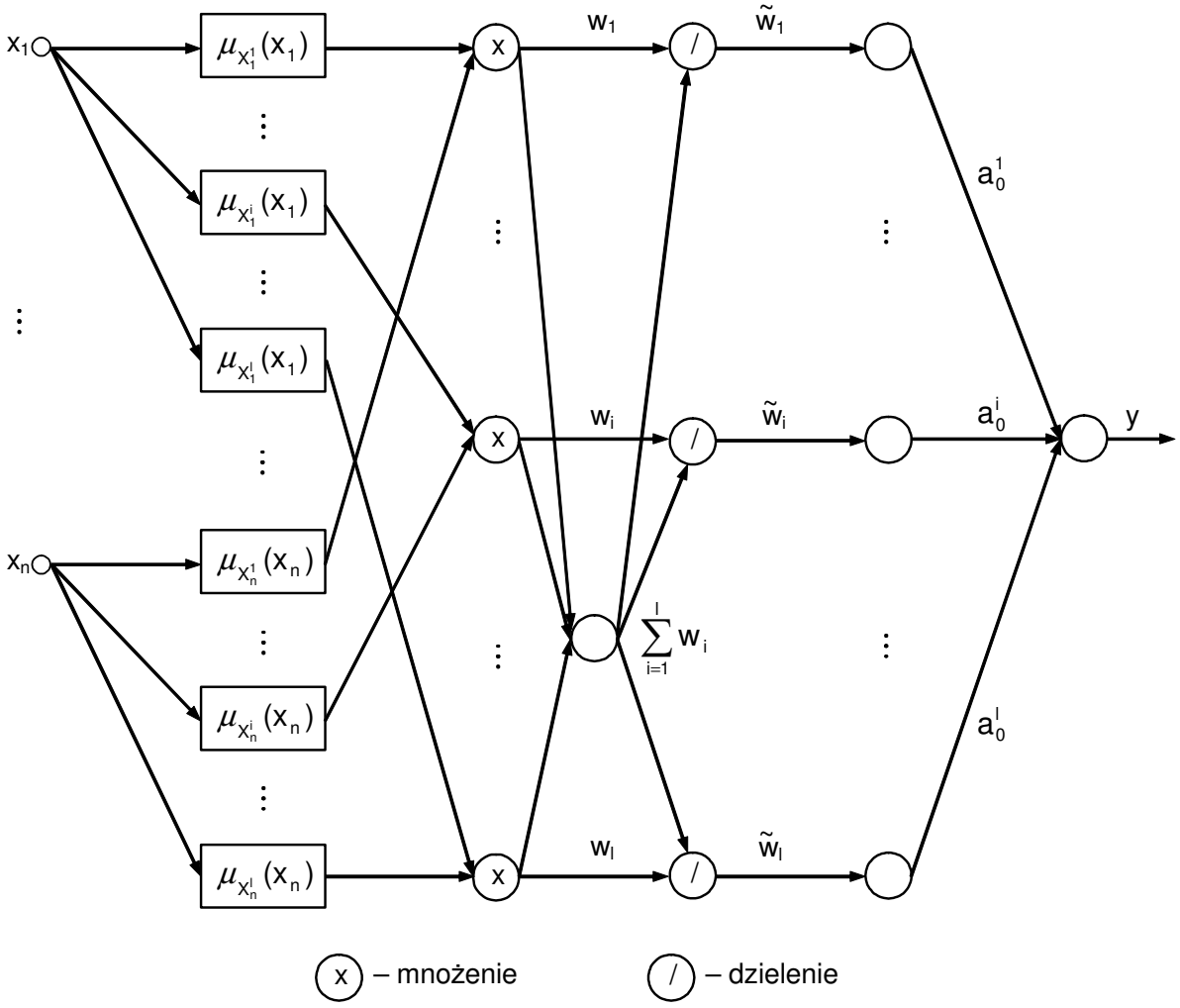
W przypadku, gdy następniki w modelu Takagi–Sugeno są stałe, tzn.

$$y^i = a_0^i, \tag{4.46}$$

model ulega znacznemu uproszczeniu. W takim przypadku, wyjście modelu jest bowiem dane wzorem:

$$y = \sum_{i=1}^{l} \tilde{w}_i \cdot a_0^i \,. \tag{4.47}$$

Także struktura sztucznej sieci neuronowej opisującej taki model znaczenie się upraszcza. Tego typu model neuronowy jest nazywany modelem Wanga–Mendela a jego postać została pokazana na rys. 4.19. Parametry modelu, które mogą zostać dobrane dzięki zastosowaniu mechanizmu uczenia sieci to parametry funkcji przynależności $\mu_{X_j^i}(x_j)$ oraz stałe a_0^i występujące w następnikach reguł modelu rozmytego.



Rys. 4.19. Neuronowy model Wanga–Mendela

Przykład 4.9

Rozpatrzmy rozmyty model statyki zaworu z przykładu 4.5. Przypomnijmy, że model ten jest złożony z następujących dwóch reguł:

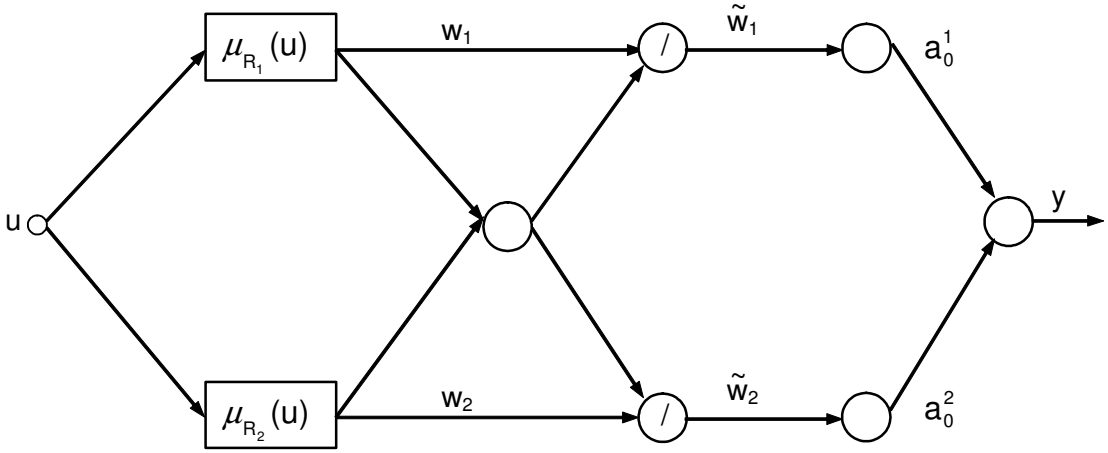
Reguła 1: jeśli *u* jest R_1, to

$$y^1 = a_0^1 = -0{,}3289\,,$$

Reguła 2: jeśli *u* jest R_2, to

$$y^2 = a_0^2 = 0{,}3289\,.$$

W takim razie sztuczna sieć neuronowa opisująca ten model będzie miała postać Wanga–Mendela. Zauważmy, że założono sigmoidalne funkcje przynależności $\mu_{R_1}(u)$ oraz $\mu_{R_2}(u)$. Ponieważ w modelu użyte zostały poprzedniki proste, struktura sieci również jest szczególnie prosta (nie jest potrzebne użycie mnożenia).



Rys. 4.20. Neuronowy model statyki zaworu z przykładu 4.5

4.3.2. Hybrydowy algorytm uczenia rozmytych sieci neuronowych

Zauważmy, że szczegółowa zależność opisująca wyjście modelu Takagi–Sugeno (a tym samym wyjście rozmytej sieci neuronowej, za pomocą której można taki model przedstawić) jest dana następującym wzorem:

$$y(x_1,\ldots,x_n)=\frac{\sum_{i=1}^{l}\prod_{j=1}^{n}\mu_{X_j^i}(x_j)\cdot\left(\sum_{j=1}^{n}a_j^i\cdot x_j+a_0^i\right)}{\sum_{i=1}^{l}\prod_{j=1}^{n}\mu_{X_j^i}(x_j)}. \tag{4.48}$$

Parametry modelu rozmytego można więc dostroić metodą uczenia rozmytej sieci neuronowej. W tym celu można wykorzystać dobrze znany mechanizm wstecznej propagacji błędu. Ze względu na strukturę rozmytej sieci neuronowej (wynikającą ze struktury reguł modelu rozmytego) możliwe jest zastosowanie dwóch zasadniczych podejść. Pierwsze podejście polega na dostrajaniu zarówno parametrów poprzedników jak i następników reguł (wszystkich parametrów podlegających procesowi identyfikacji) analogicznie, jak w przypadku perceptronu wielowarstwowego. Drugie podejście to algorytm hybrydowy, w którym parametry poprzedników są dostrajane z wykorzystaniem mechanizmu uczenia zaś parametry następników – z wykorzystaniem metody najmniejszych kwadratów. Druga z wymienionych metod daje lepsze rezultaty [29, 38] i na tej właśnie metodzie się teraz skoncentrujemy.

Podczas uczenia sieci będziemy dążyć do minimalizowania następującej funkcji błędu:

$$E=\frac{1}{2}\sum_{m=1}^{p}\left(y_m-y_m^d\right)^2, \tag{4.49}$$

gdzie p jest liczbą próbek uczących, $y_m=y_m(x_1^m,\ldots,x_n^m)$ jest wartością wyjścia uczonej sieci, y_m^d jest wartością wyjścia (zmierzoną) obiektu otrzymaną dla zestawu sygnałów wejściowych

$(x_1^m,\ldots,x_n^m)$. Jak już wspomniano na wstępie, w podejściu hybrydowym, dostrajanie parametrów poprzedników i następników odbywa się osobno. Dlatego też w algorytmie można wyróżnić dwa etapy, powtarzane na zmianę podczas procesu uczenia [29].

Etap I

Dla ustalonych (bieżących) wartości parametrów funkcji przynależności, dostrajane są wartości parametrów następników reguł rozmytych. Zauważmy, że w przypadku przyjęcia stałych parametrów funkcji przynależności, wyjście modelu rozmytego jest dane wzorem:

$$y = y(x_1,\ldots,x_n) = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(\sum_{j=1}^{n} a_j^i \cdot x_j + a_0^i \right). \tag{4.50}$$

Powyższy wzór można zapisać w postaci wektorowej:

$$y = \begin{bmatrix} \tilde{w}_1 \cdot x_n & \ldots & \tilde{w}_1 \cdot x_1 & \tilde{w}_1 & \ldots & \tilde{w}_l \cdot x_n & \ldots & \tilde{w}_l \cdot x_1 & \tilde{w}_l \end{bmatrix} \cdot \begin{bmatrix} a_n^1 \\ \vdots \\ a_1^1 \\ a_o^1 \\ \vdots \\ a_n^l \\ \vdots \\ a_1^l \\ a_0^l \end{bmatrix}. \tag{4.51}$$

Naszym zadaniem jest określenie na podstawie p próbek uczących, którymi dysponujemy, wartości parametrów a_j^i ($i = 1,\ldots, l$; $j = 1,\ldots, n$). W idealnym przypadku chcemy otrzymać następującą równość:

$$\boldsymbol{W} \cdot \boldsymbol{a} = \boldsymbol{y}^{\boldsymbol{d}}, \tag{4.52}$$

gdzie

$$\boldsymbol{W} = \begin{bmatrix} \tilde{w}_1^1 \cdot x_n^1 & \ldots & \tilde{w}_1^1 \cdot x_1^1 & \tilde{w}_1^1 & \ldots & \tilde{w}_l^1 \cdot x_n^1 & \ldots & \tilde{w}_l^1 \cdot x_1^1 & \tilde{w}_l^1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \tilde{w}_1^m \cdot x_n^m & \ldots & \tilde{w}_1^m \cdot x_1^m & \tilde{w}_1^m & \ldots & \tilde{w}_l^m \cdot x_n^m & \ldots & \tilde{w}_l^m \cdot x_1^m & \tilde{w}_l^m \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \vdots \\ \tilde{w}_1^p \cdot x_n^p & \ldots & \tilde{w}_1^p \cdot x_1^p & \tilde{w}_1^p & \ldots & \tilde{w}_l^p \cdot x_n^p & \ldots & \tilde{w}_l^p \cdot x_1^p & \tilde{w}_l^p \end{bmatrix}, \quad \boldsymbol{a} = \begin{bmatrix} a_n^1 \\ \vdots \\ a_1^1 \\ a_o^1 \\ \vdots \\ a_n^l \\ \vdots \\ a_1^l \\ a_0^l \end{bmatrix},$$

$$\boldsymbol{y^d} = \begin{bmatrix} y_1^d \\ \vdots \\ y_m^d \\ \vdots \\ y_p^d \end{bmatrix}.$$

Zwykle zbiór próbek uczących jest liczny i liczba tych próbek znacznie przekracza liczbę parametrów zgrupowanych w wektorze $\boldsymbol{a}$. W związku z tym wartości parametrów następników reguł rozmytych wyznacza się korzystając z metody najmniejszych kwadratów, co w programie Matlab sprowadza się do użycia operatora tzw. lewego dzielenia macierzowego (ang. left matrix divide).

Etap II

Dla bieżących wartości parametrów następników reguł modelu rozmytego oraz funkcji przynależności, dla poszczególnych próbek uczących wyznacza się wartości wyjściowe uczonej sieci $y_m = y_m(x_1^m,\ldots,x_n^m)$. Na podstawie tych wartości oraz wartości pożądanych y_m^d otrzymuje się wektor błędu:

$$\boldsymbol{\varepsilon} = \begin{bmatrix} y_1 - y_1^d \\ \vdots \\ y_m - y_m^d \\ \vdots \\ y_p - y_p^d \end{bmatrix}.$$

Następnie korzysta się z mechanizmu wstecznej propagacji błędu. Dostrajanie wartości parametrów funkcji przynależności wykonuje się, korzystając z wybranej metody optymalizacji.

Przykład 4.10

Oznaczmy przez c_i parametry danej funkcji przynależności. Załóżmy także, że funkcja ta jest różniczkowalna. Wówczas, w przypadku użycia metody najszybszego spadku, otrzymuje się następujący wzór opisujący iteracyjne dostrajanie tych parametrów:

$$c_i(n+1) = c_i(n) - \eta_i \cdot \frac{\partial E}{\partial c_i}, \tag{4.53}$$

gdzie n jest numerem kolejnej iteracji uczenia, η_i jest współczynnikiem uczenia oraz

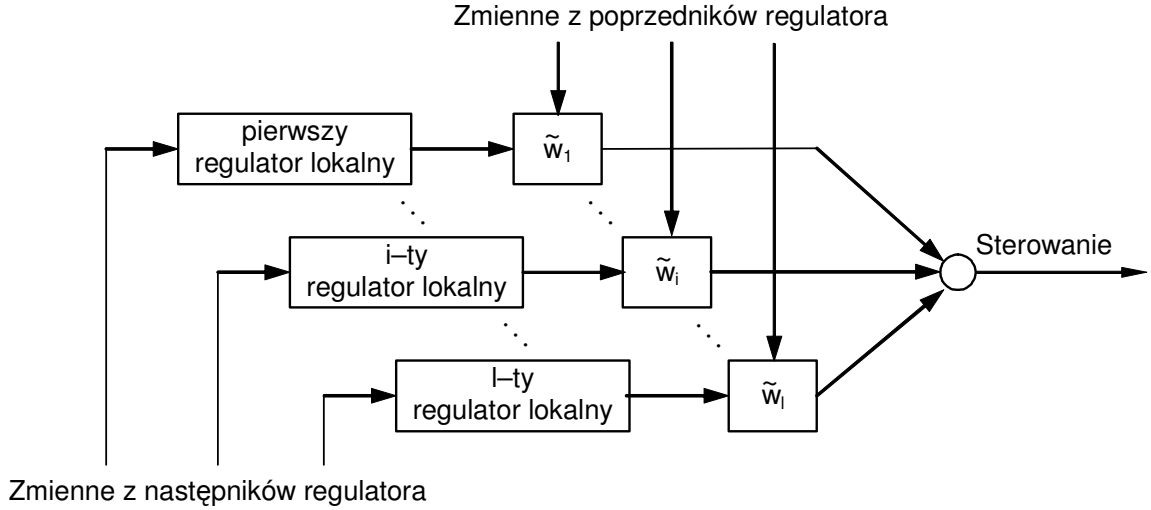
$$\frac{\partial E}{\partial c_i} = \left(y - y^d\right) \cdot \sum_{i=1}^{l} \left(\sum_{j=1}^{n} a_j^i \cdot x_j + a_0^i \right) \cdot \frac{\partial \tilde{w}_i}{\partial c_i}, \tag{4.54}$$

gdzie $\frac{\partial \tilde{w}_i}{\partial c_i}$ zależy od postaci przyjętej funkcji przynależności.

Uwaga: Gradientowe metody optymalizacji można zastosować w przypadku różniczkowalnych funkcji przynależności (np. sigmoidalna, dzwonowa, Gaussa). W przypadku funkcji nieróżniczkowalnych można jednak użyć metod bezgradientowych.

4.4. Regulatory bazujące na modelach Takagi–Sugeno

Często spotykanym sposobem wykorzystania modelu Takagi–Sugeno obiektu do syntezy regulatora jest skorzystanie z mechanizmu rozproszonej równoległej kompensacji (ang. Parallel Distributed Compensation – PDC).



Rys. 4.21. Regulator typu Takagi–Sugeno otrzymany za pomocą podejścia PDC

Podejście PDC polega na dobraniu dla każdego następnika modelu Takagi–Sugeno obiektu regulacji, liniowego regulatora. W wyniku tego zabiegu otrzymuje się tyle regulatorów lokalnych, w projektowanym regulatorze rozmytym typu Takagi–Sugeno, z ilu modeli lokalnych złożony jest model obiektu. Poprzedniki zakłada się początkowo takie same, jak w modelu obiektu i w miarę potrzeby są one później dostrajane. W wyniku zastosowania podejścia PDC otrzymuje się więc regulator, którego ogólna struktura została pokazana na rys. 4.21. Regulatorami lokalnymi mogą być regulatory analityczne przypomniane w rozdz. 2, chociaż nie wyczerpują one oczywiście wszystkich możliwości. Zastosowane mogą bowiem zostać dowolne regulatory, dla których jest możliwe otrzymanie prawa regulacji.

4.4.1. Rozmyty regulator ze sprzężeniem od stanu

Załóżmy, że obiekt regulacji jest opisany modelem rozmytym Takagi–Sugeno złożonym z następujących reguł:

Reguła *i*: jeśli $x_1(k)$ jest X_1^i i … i $x_n(k)$ jest X_n^i, to

$$\boldsymbol{x}_{k+1}^i = \boldsymbol{A}_i \cdot \boldsymbol{x}_k + \boldsymbol{B}_i \cdot u_k , \tag{4.55}$$

gdzie $x_1(k),\ldots,x_n(k)$ są zmiennymi stanu, $X_1^i,\ldots,X_n^i$ są zbiorami rozmytymi, $\boldsymbol{A}_i$ są macierzami stanu modeli lokalnych, $\boldsymbol{B}_i$ są wektorami sterowań modeli lokalnych ($i = 1,\ldots,l$), u_k jest sterowaniem obiektu, $\boldsymbol{x}_k$ jest wektorem stanu:

$$\boldsymbol{x}_k = \left[x_1(k),\ldots,x_n(k)\right]^T .$$

Wyjście tego modelu Takagi–Sugeno jest więc opisane wzorem:

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(\boldsymbol{A}_i \cdot \boldsymbol{x}_k + \boldsymbol{B}_i \cdot u_k\right). \tag{4.56}$$

Dla każdego następnika modelu rozmytego (4.55) można zaprojektować regulator lokalny ze sprzężeniem od stanu korzystając na przykład z metody lokowania biegunów układu zamkniętego przypomnianej w rozdz. 2.3. Taki regulator rozmyty będzie złożony z następujących reguł:

Reguła *j*: jeśli $x_1(k)$ jest X_1^j i … i $x_n(k)$ jest X_n^j, to

$$u_k^j = \boldsymbol{K}_j \cdot \boldsymbol{x}_k , \tag{4.57}$$

gdzie $\boldsymbol{K}_j$ ($j = 1,\ldots,l$) są wektorami wzmocnień. Wartości elementów tych wektorów należy dobrać podczas projektowania regulatora.

Uwaga: Chociaż w ogólnym przypadku poprzedniki reguł w regulatorze można przyjąć inne niż w modelu obiektu, to naturalnym podejściem jest założenie takich samych poprzedników i użycie mechanizmu PDC.

Wyjście regulatora rozmytego Takagi–Sugeno ze sprzężeniem od stanu jest opisane wzorem:

$$u_k = \sum_{j=1}^{l} \tilde{w}_j \cdot \boldsymbol{K}_j \cdot \boldsymbol{x}_k . \tag{4.58}$$

W takim razie, po podstawieniu zależności (4.58) do (4.56) otrzymamy opis układu regulacji obiektu opisanego modelem (4.55), z regulatorem rozmytym typu Takagi–Sugeno ze sprzężeniem od stanu:

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{l}\sum_{j=1}^{l} \tilde{w}_i \cdot \tilde{w}_j \cdot \left(\boldsymbol{A}_i + \boldsymbol{B}_i \cdot \boldsymbol{K}_j\right) \cdot \boldsymbol{x}_k . \tag{4.59}$$

Przykład 4.11

Rozważmy układ dynamiczny opisany modelem rozmytym Takagi–Sugeno złożonym z następujących reguł:

Reguła 1: jeśli $x_1(k)$ jest X_1^1, to

$$x^1(k+1) = \boldsymbol{A}_1 \cdot x(k) + \boldsymbol{B} \cdot u(k) , \tag{4.60}$$

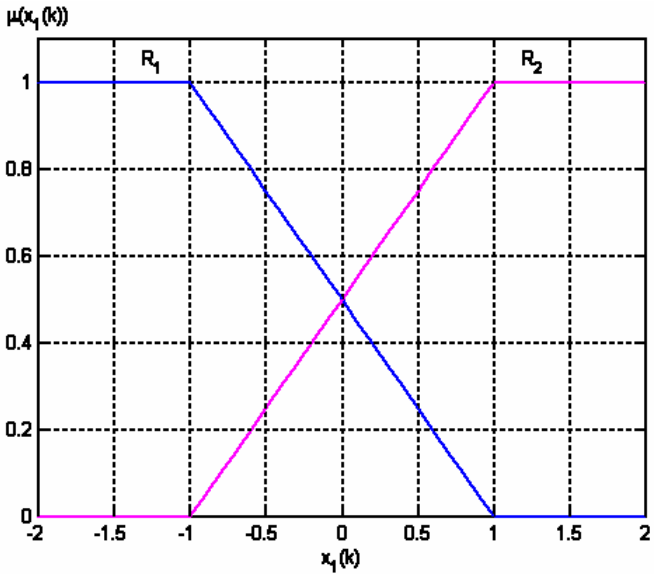
Reguła 2: jeśli $x_1(k)$ jest X_1^2, to

$$x^2(k+1) = \boldsymbol{A}_2 \cdot x(k) + \boldsymbol{B} \cdot u(k) , \tag{4.61}$$

gdzie

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & -0{,}6 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 1 & -0{,}4 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

Załóżmy, że funkcje przynależności są takie, jak na rys. 4.22.



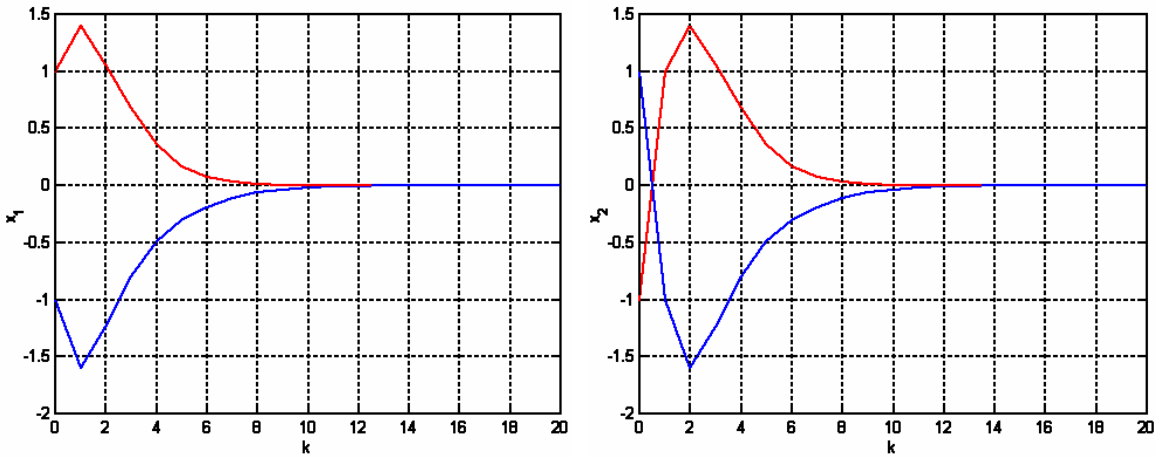
Rys. 4.22. Funkcje przynależności przykładowego obiektu

Załóżmy także, że poprzedniki regulatora są takie same jak w modelu obiektu oraz że bieguny układów lokalnych ze sprzężeniem od stanu (wartości własne macierzy $\boldsymbol{A}_1+\boldsymbol{B}\boldsymbol{K}_1$ oraz $\boldsymbol{A}_2+\boldsymbol{B}\boldsymbol{K}_2$) są położone w punktach 0,5 i 0,4. W takim razie wektory wzmocnień w regulatorach lokalnych są następujące:

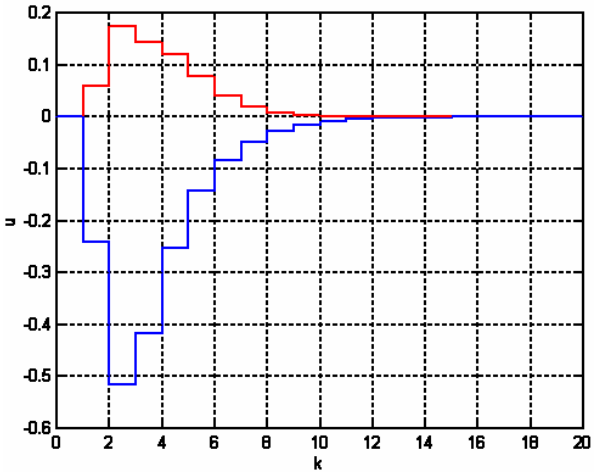
$$\boldsymbol{K}_1 = \begin{bmatrix} -0{,}1 & 0{,}4 \end{bmatrix}, \quad \boldsymbol{K}_2 = \begin{bmatrix} -0{,}1 & 0{,}2 \end{bmatrix}.$$

Przebiegi otrzymane w zaprojektowanym układzie regulacji dla różnych warunków początkowych zostały zaprezentowane na rys. 4.23. W przykładzie założono, że pomiar obydwu zmiennych stanu jest dostępny. Zauważmy jednak, że w przypadku, gdyby tak nie było, można zaprojektować obserwator rozmyty w sposób analogiczny do projektowania rozmytego regulatora ze sprzężeniem od stanu.

a) b)



Rys. 4.23a. Odpowiedzi układu regulacji z rozmytym regulatorem ze sprzężeniem od stanu na niezerowe warunki początkowe $\boldsymbol{x}_0=[1\ -1]^T$ oraz $\boldsymbol{x}_0=[-1\ 1]^T$; przebiegi: a) zmiennej stanu x_1, b) zmiennej stanu x_2



Rys. 4.23b. Odpowiedzi układu regulacji z rozmytym regulatorem ze sprzężeniem od stanu na niezerowe warunki początkowe $\boldsymbol{x}_0=[1\ -1]^T$ oraz $\boldsymbol{x}_0=[-1\ \ 1]^T$; przebieg zmiennej sterującej u

4.4.2. Rozmyty regulator PID

Popularnym regulatorem jest regulator typu PID. Jego odmiana rozmyta typu Takagi–Sugeno jest złożona z następujących reguł:

Reguła i: jeśli x_1 jest X_1^i i … i x_n jest X_n^i , to

$$u^i(k) = u(k-1) + r_1^i \cdot e(k) + r_2^i \cdot e(k-1) + r_3^i \cdot e(k-2)\,, \tag{4.62}$$

gdzie r_1^i, r_2^i, r_3^i $(i = 1,\ldots l)$ są parametrami regulatorów lokalnych, e_k jest uchybem regulacji w k–tej chwili próbkowania, x_j są zmiennymi (najczęściej będą to wartości wyjścia i sterowania z przeszłych chwil próbkowania). Tak jak w przypadku regulatora ze sprzężeniem od stanu, najwygodniej jest skorzystać z podejścia PDC i założyć takie same poprzedniki, jak w rozmytym modelu obiektu. Następnie parametry regulatorów lokalnych można dobrać stosując dowolną metodę. Wyjście rozmytego regulatora PID typu Takagi–Sugeno można obliczyć korzystając z następującego wzoru:

$$u(k) = u(k-1) + \tilde{r}_1 \cdot e(k) + \tilde{r}_2 \cdot e(k-1) + \tilde{r}_3 \cdot e(k-2)\,, \tag{4.63}$$

gdzie

$$\tilde{r}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot r_j^i\,.$$

Przykład 4.12

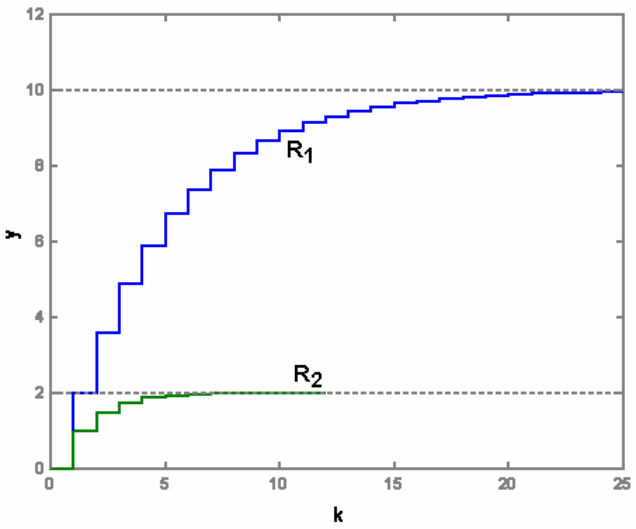
Rozważmy układ dynamiczny opisany modelem rozmytym Takagi–Sugeno złożonym z następujących reguł:

Reguła 1: jeśli $y(k)$ jest Y^1, to

$$y^1(k+1) = 0{,}8 \cdot y(k) + 2 \cdot u(k)\,, \tag{4.64}$$

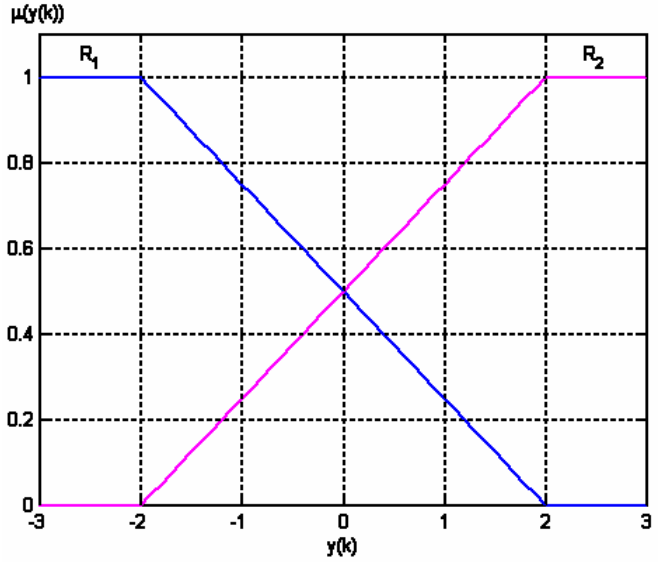
Reguła 2: jeśli $y(k)$ jest Y^2, to

$$y^2(k+1) = 0{,}5 \cdot y(k) + u(k)\,, \tag{4.65}$$



Rys. 4.24. Odpowiedzi skokowe modeli lokalnych

Zauważmy, że występuje duża różnica wzmocnień modeli lokalnych (wzmocnienie pierwszego wynosi k_1=10 a drugiego k_2=2). Ponadto, porównując odpowiedzi skokowe modeli lokalnych (rys. 4.24) można zauważyć, że pierwszy z nich jest znacznie wolniejszy od drugiego. Założone funkcje przynależności zostały przedstawione na rys. 4.25.



Rys. 4.25. Funkcje przynależności przykładowego obiektu

Do rozważanego obiektu opracowano dyskretny, rozmyty regulator PI typu Takagi–Sugeno. Postać przesłanek założono taką samą, jak w modelu obiektu. W związku z tym regulator składa się z następujących dwóch reguł:

Reguła 1: jeśli $y(k)$ jest Y^1, to

$$u^1(k) = u(k-1) + r_1^1 \cdot e(k) + r_2^1 \cdot e(k-1), \tag{4.66}$$

Reguła 2: jeśli $y(k)$ jest Y^2, to

$$u^2(k) = u(k-1) + r_1^2 \cdot e(k) + r_2^2 \cdot e(k-1). \tag{4.67}$$

Zadaniem projektanta jest więc znalezienie wartości parametrów r_1^1, r_2^1, r_1^2, r_2^2 regulatorów lokalnych. W niniejszym przykładzie założono, że parametry te należy dobrać w taki sposób, aby bieguny układów regulacji złożonych z odpowiadających sobie modelu i regulatora lokalnych były położone w wybranych przez projektanta miejscach. W takim razie wielomian charakterystyczny układu zamkniętego ma być następujący:

$$(z - z_1)\cdot(z - z_2) = z^2 - (z_1 + z_2)\cdot z + z_1 \cdot z_2, \tag{4.68}$$

gdzie z_1 i z_2 to założone bieguny układu zamkniętego. Po przekształceniach otrzyma się następujące wzory służące do obliczenia parametrów regulatorów lokalnych:

$$r_1^i = \frac{-(z_1 + z_2) - (a^i - 1)}{b^i}, \tag{4.69}$$

$$r_2^i = \frac{z_1 \cdot z_2 + a^i}{b^i}, \tag{4.70}$$

gdzie a^i oraz b^i to parametry modeli lokalnych przy założeniu następującej ich postaci:

$$y^i(k+1) = -a^i \cdot y(k) + b^i \cdot u(k). \tag{4.71}$$

W takim razie, w rozważanym przykładzie $a^1 = -0{,}8$, $b^1 = 2$, $a^2 = -0{,}5$, $b^2 = 1$.

Ostatecznie założono, że bieguny powinny być takie same i położone w punkcie:

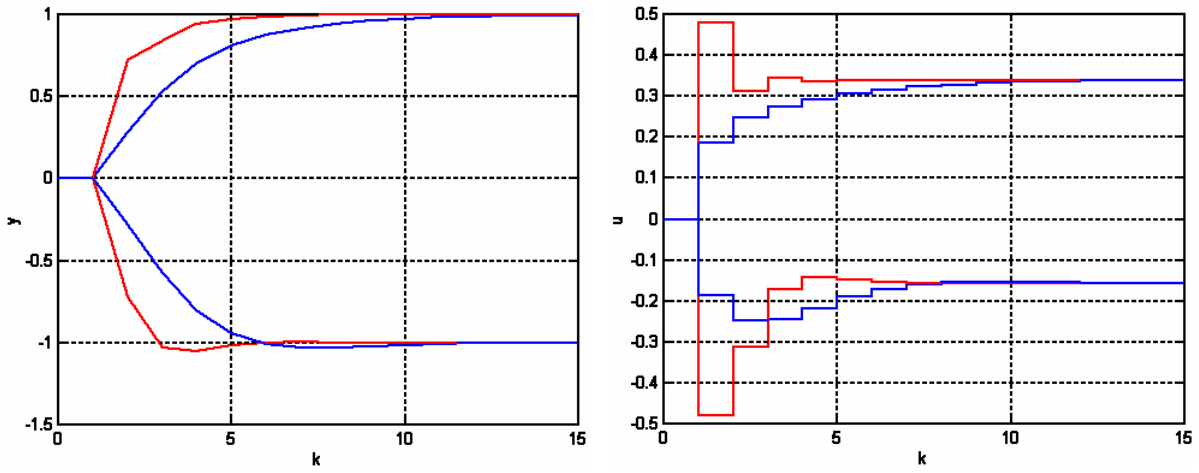
a) $z_1 = z_2 = 0{,}5$; w tym przypadku otrzymano następujące wartości parametrów regulatorów lokalnych:

$$r_1^1 = 0{,}4;\ r_2^1 = -0{,}275;\ r_1^2 = 0{,}5;\ r_2^2 = -0{,}25;$$

b) $z_1 = z_2 = 0{,}2$; tym razem otrzymano:

$$r_1^1 = 0{,}7;\ r_2^1 = -0{,}38;\ r_1^2 = 1{,}1;\ r_2^2 = -0{,}46.$$

a) b)



Rys. 4.26. Odpowiedzi układu regulacji z rozmytym regulatorem PI na skoki wartości zadanej z $y_{zad}^0 = 0$ do $y_{zad}^1 = 1$ i $y_{zad}^2 = -1$; bieguny lokalnych układów regulacji położone w $z_1 = z_2 = 0{,}5$ oraz $z_1 = z_2 = 0{,}2$: a) przebiegi wyjścia y, b) przebiegi sterowania u

Odpowiedzi na zmiany wartości zadanej otrzymane w układach regulacji z rozmytymi regulatorami PI oraz przykładowym obiektem zostały przedstawione na rys. 4.26. Zgodnie z przewidywaniami, im bliżej 0 są położone bieguny tym szybciej działa układ regulacji ale dzieje się to przy gwałtowniejszych zmianach sygnału sterującego. Warto także zwrócić uwagę na różnicę w charakterze odpowiedzi na skoki wartości zadanej do wartości 1 i –1 spowodowane nieliniowością układu.

4.4.3. Rozmyte regulatory predykcyjne w wersji analitycznej

Regulatory predykcyjne bazujące na modelach rozmytych tak, jak regulatory bazujące na modelach liniowych, występują w dwóch głównych wersjach: analitycznej i numerycznej. W przypadku tej pierwszej, podobnie jak w przypadku regulatorów PID lub ze sprzężeniem od stanu można zastosować podejście PDC.

Regulator rozmyty DMC

Na początku naszych rozważań zajmijmy się analitycznym regulatorem rozmytym DMC typu Takagi–Sugeno. Załóżmy, że rozmyty model obiektu ma przesłanki takie, jak w przykładzie 4.6 a modele lokalne w postaci odpowiedzi skokowych (por. (2.46)) pozyskanych w okolicach kilku punktów pracy. W takim razie model ten ma postać:

Reguła *i*: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = \sum_{j=1}^{D-1} s_j^i \cdot \Delta u_{k-j+1} + s_D^i \cdot u_{k-D+1}, \tag{4.72}$$

gdzie s_j^i ($i = 1,\ldots,l$, $j = 1,\ldots,D$) są współczynnikami modeli lokalnych. Zauważmy, że tego typu model jest stosunkowo łatwy do otrzymania i umożliwia dokonanie szybkiej syntezy regulatora. Poprzedniki oraz kształt funkcji przynależności mogą być dobrane z wykorzystaniem wiedzy eksperta, wspomaganej na przykład analizą charakterystyk oraz odpowiedzi skokowych obiektu regulacji.

Po otrzymaniu zestawu modeli lokalnych (lokalnych odpowiedzi skokowych), dla założonych wartości parametrów regulatora DMC, tzn. horyzontu predykcji N, horyzontu sterowania N_u oraz parametru dostrajalnego λ, dla każdego modelu lokalnego otrzymuje się regulator DMC w wersji analitycznej (2.55). Otrzymane regulatory są regulatorami lokalnymi analitycznego regulatora rozmytego DMC typu Takagi–Sugeno, który jest opisany zestawem następujących reguł:

Reguła *i*: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$\Delta u(k) = \left(k^e\right)^i \cdot e(k) - \sum_{j=1}^{D-1} \left(k_j^u\right)^i \cdot \Delta u(k-j) \tag{4.73}$$

gdzie $\left(k^e\right)^i, \left(k_1^u\right)^i, \ldots, \left(k_D^u\right)^i$ ($i = 1,\ldots l$) są parametrami lokalnych regulatorów DMC. W takim razie wyjście analitycznego regulatora rozmytego DMC typu Takagi–Sugeno można obliczyć korzystając ze wzoru:

$$\Delta u(k) = \tilde{k}^e \cdot e(k) - \sum_{j=1}^{D-1} \tilde{k}_j^u \cdot \Delta u(k-j), \tag{4.74}$$

gdzie

$$\tilde{k}^{e} = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(k^{e}\right)^{i}, \; \tilde{k}_j^{u} = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(\tilde{k}_j^{u}\right)^{i}.$$

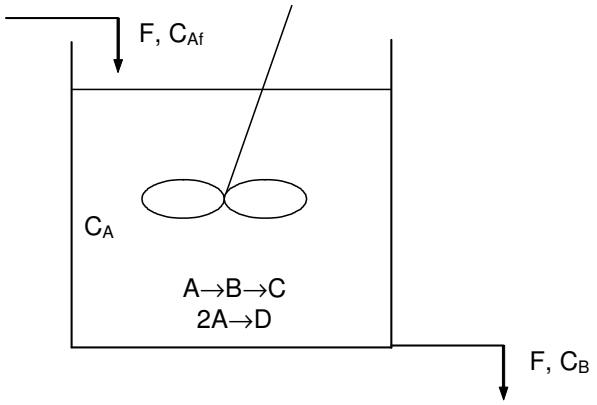
Wyjściem analitycznego regulatora rozmytego DMC typu Takagi–Sugeno jest przyrost sterowania. W celu wyznaczenia wartości sterowania, należy skorzystać z następującej zależności:

$$u(k) = u(k-1) + \tilde{k}^{e} \cdot e(k) - \sum_{j=1}^{D-1} \tilde{k}_j^{u} \cdot \Delta u(k-j). \tag{4.75}$$

Uwaga: Zauważmy, że poprzedniki regulatora rozmytego mogą być zmieniane już po opracowaniu lokalnych regulatorów DMC i przetestowaniu regulatora ze wstępną wersją poprzedników. Zmiany te będą miały wpływ na działanie układu regulacji. W takim razie, można spróbować poprawić działanie układu regulacji dzięki manipulowaniu poprzednikami reguł w regulatorze rozmytym (manipulowaniu sposobem obliczania wag dla poszczególnych regulatorów lokalnych).

Przykład 4.13

Aby poznać możliwości oferowane przez podejście rozmyte, prześledźmy proces opracowania analitycznego, rozmytego regulatora DMC na przykładzie z pracy [21]. Obiektem regulacji jest nieliniowy reaktor chemiczny, w którym zachodzi reakcja van de Vusse'a [5]. Schemat reaktora został pokazany na rys. 4.27. Sterowaniem jest natężenie dopływu surowca F, wyjściem regulowanym jest stężenie C_B substancji B w produkcie, wejściem zakłócającym jest stężenie C_{Af} substancji A w strumieniu zasilającym reaktor.



Rys. 4.27. Schemat reaktora z reakcją van de Vusse'a

Równania opisujące zmiany stężeń substancji A i B, zachodzące w reaktorze są następujące:

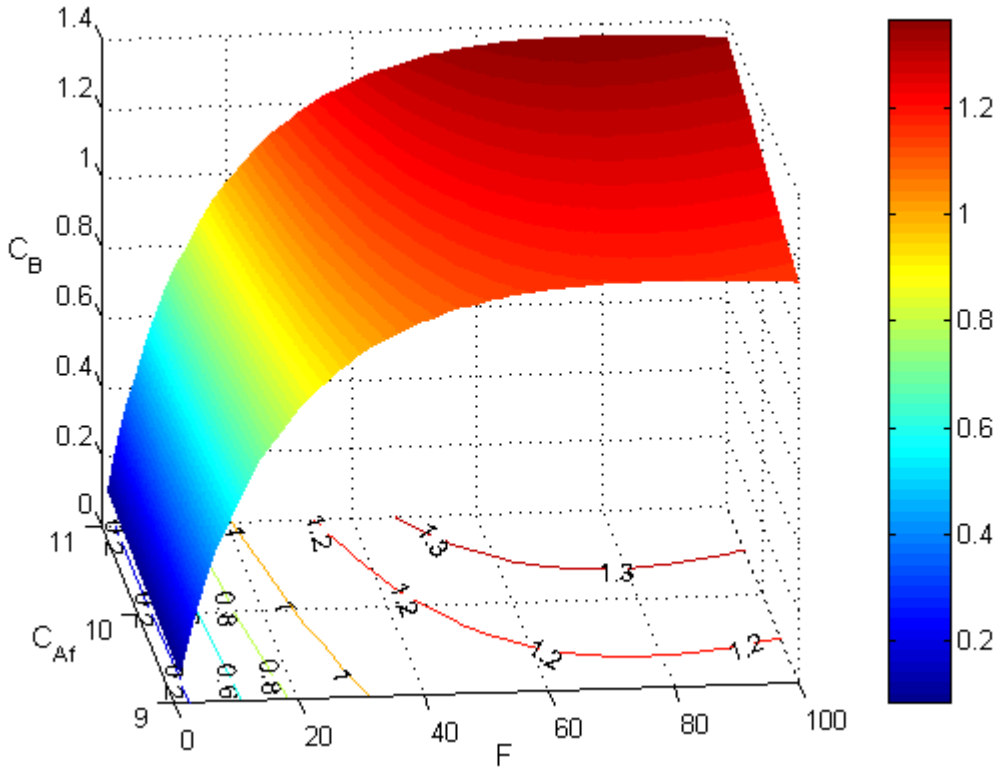
$$\begin{aligned} \frac{dC_A}{dt} &= -k_1 C_A - k_3 C_A^2 + \frac{F}{V}(C_{Af} - C_A), \\ \frac{dC_B}{dt} &= k_1 C_A - k_2 C_B - \frac{F}{V} C_B, \end{aligned} \tag{4.76}$$

gdzie C_A, C_B są stężeniami substancji A i B, założono, że dopływ surowca F jest równy odpływowi z reaktora, V jest objętością, w której zachodzi reakcja (założono, że jest ona stała), C_{Af} jest stężeniem substancji A w strumieniu zasilającym reaktor; jeśli nie zaznaczono,

że jest inaczej, przyjmowana jest wartość stała równa C_{Af0}, k_1, k_2 i k_3 są parametrami reakcji. Wartości stałych występujących w modelu (4.76) zestawiono w tabl. 4.2. Rozważany obiekt jest nieliniowy o czym świadczy jego charakterystyka statyczna, pokazana na rys. 4.28.

Tablica 4.2. Parametry i stałe w modelu (4.76) reaktora z reakcją van de Vusse'a

$k_1 = 50\,1/h$	$V = 1\,l$
$k_2 = 100\,1/h$	$C_{Af\,0} = 10\ mol/l$
$k_3 = 10\,l/h \cdot mol$	



Rys. 4.28. Charakterystyka statyczna reaktora z reakcją van de Vusse'a

Dla przykładowego obiektu zaprojektowano analityczny regulator rozmyty DMC. W tym celu opracowano model rozmyty typu Takagi–Sugeno (4.72), z modelami lokalnymi w postaci odpowiedzi skokowych otrzymanych z okolic trzech punktów pracy:

R1) $C_{B0} = 0{,}91\ mol/l$, $C_{A0} = 2{,}18\ mol/l$, $F = 20\,l/h$;

R2) $C_{B0} = 1{,}12\ mol/l$, $C_{A0} = 3\ mol/l$, $F = 34{,}3\,l/h$;

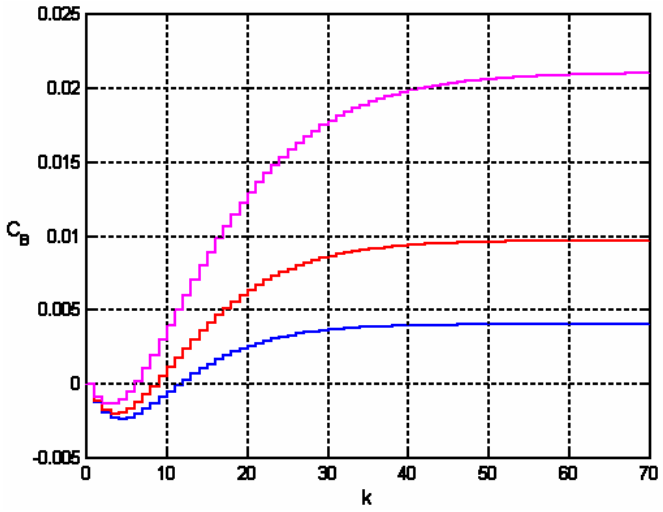
R3) $C_{B0} = 1{,}22\ mol/l$, $C_{A0} = 3{,}66\ mol/l$, $F = 50\,l/h$.

Otrzymane odpowiedzi zostały pokazane na rys. 4.29. Warto przy tym zauważyć, że rozważany obiekt ma odpowiedź odwrotną. Jest więc trudny do regulacji i naturalne przy wyborze regulatora jest zastosowanie regulatora predykcyjnego. Wstępnie załóżmy, że zmienną wykorzystaną w przesłankach będzie ostatnio zmierzona wartość stężenia substancji B w produkcie $C_B(k)$. W takim razie model obiektu regulacji użyty do opracowania rozmytego regulatora DMC ma następującą postać:

Reguła *i*: jeśli $C_B(k)$ jest Ri, to

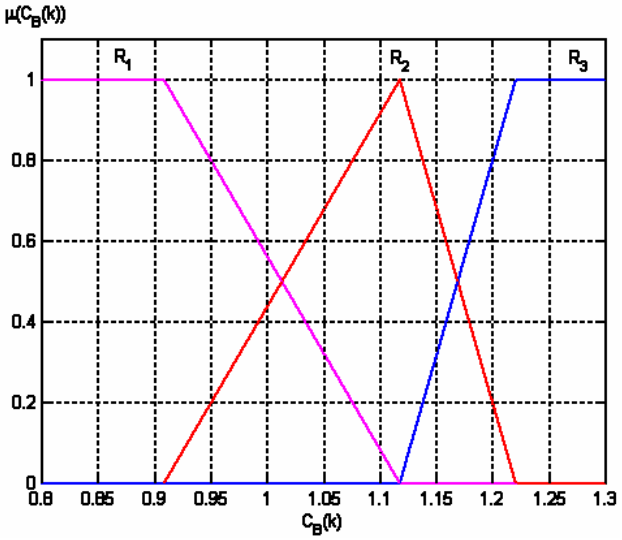
$$y_{k+1}^i = \sum_{j=1}^{D-1} s_j^i \cdot \Delta u_{k-j+1} + s_D^i \cdot u_{k-D+1},$$

gdzie parametry lokalnych odpowiedzi skokowych s_j^i (i = 1, 2, 3, j = 1,…, D, D=70) zostały otrzymane ze znormalizowanych odpowiedzi skokowych przedstawionych na rys. 4.29. Zauważmy, że obserwując zestaw odpowiedzi skokowych, można łatwo dostrzec nieliniowy charakter obiektu.



Rys. 4.29. Znormalizowane odpowiedzi skokowe reaktora z reakcją van de Vusse'a otrzymane w okolicach punktów pracy: R1, R2, R3

Początkowo przyjęte funkcje przynależności, zależne od bieżącej wartości stężenia substancji B w produkcie, zostały pokazane na rys. 4.30.



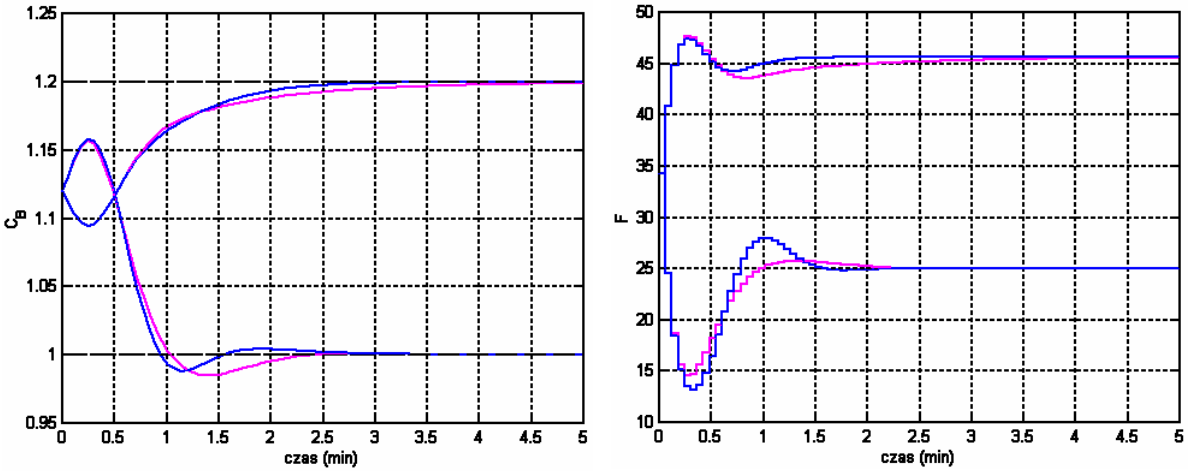
Rys. 4.30. Funkcje przynależności w regulatorze FDMC1

Założono następujące parametry rozmytego regulatora DMC (FDMC – ang. Fuzzy DMC): horyzont predykcji N=70, horyzont sterowania N_u=35, oraz parametr dostrajalny λ=0,001. Dla porównania, korzystając z odpowiedzi otrzymanej w okolicach drugiego punktu pracy (R2),

opracowano konwencjonalny (bazujący na modelu liniowym) algorytm DMC.

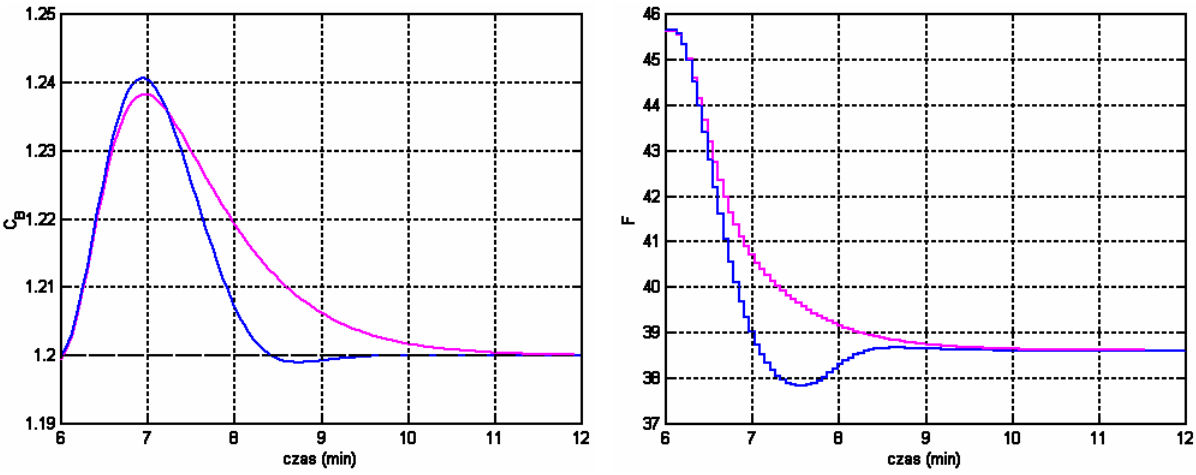
Przebiegi otrzymane w pierwszym z zaprojektowanych układów regulacji, w odpowiedzi na skoki wartości zadanej z $C_B^{zad} = 1{,}12$ do $C_B^{zad} = 1$ i $C_B^{zad} = 1{,}2$ (zmiany w stronę wyższej i niższej wartości) zostały zaprezentowane na rys. 4.31. Odpowiedzi otrzymane w układzie regulacji z regulatorem rozmytym FDMC1 (oznaczone kolorem niebieskim) są szybsze od odpowiedzi uzyskanych z konwencjonalnym regulatorem DMC (oznaczone kolorem różowym). Ponadto w przypadku zmniejszenia wartości zadanej, przeregulowanie jest mniejsze w układzie regulacji z regulatorem FDMC1. Interesujące są także odpowiedzi układów regulacji na skok zakłócenia C_{Af} (rys. 4.32). W tym przypadku regulator FDMC1 również okazał się lepszy od regulatora konwencjonalnego (szybsza kompensacja).

a) b)



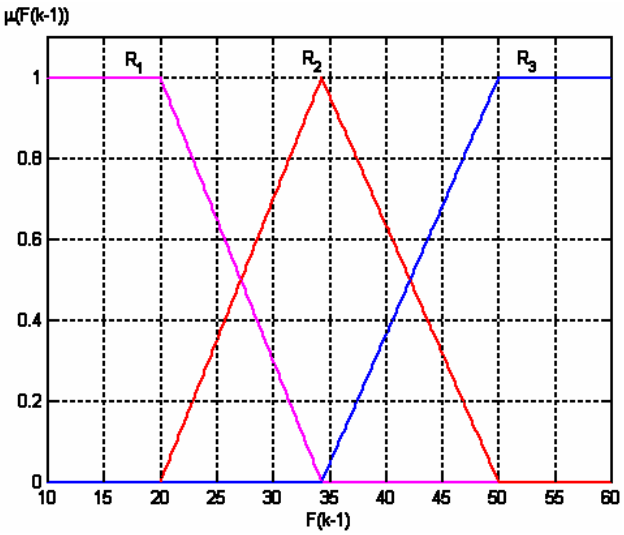
Rys. 4.31. Odpowiedzi układów regulacji z regulatorami DMC i FDMC1 na skoki wartości zadanej z $C_B^{zad} = 1{,}12$ do $C_B^{zad} = 1$ i $C_B^{zad} = 1{,}2$;
a) przebiegi wyjścia C_B, b) przebiegi sterowania F

a) b)



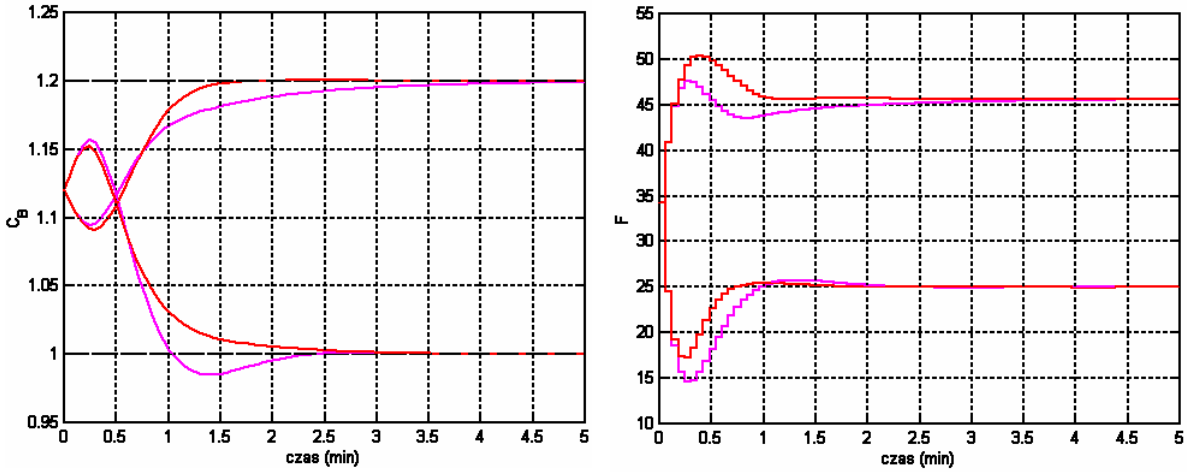
Rys. 4.32. Odpowiedzi układów regulacji z regulatorami DMC i FDMC1 na skok zakłócenia C_{Af} z $C_{Af0} = 10$ *mol/l* do $C_{Af1} = 10{,}5$ *mol/l* (o 5 %);
a) przebiegi wyjścia C_B, b) przebiegi sterowania F

Sprawdźmy, co się stanie, jeśli w poprzednikach zamiast wartości wyjściowej C_B, zostanie użyta wartość sterowania F. Załóżmy przy tym, że funkcje przynależności mają kształt pokazany na rys. 4.33. Przebiegi otrzymane w wyniku przeprowadzenia takiego samego eksperymentu, jak poprzednio zostały zaprezentowane na rys. 4.34.



Rys. 4.33. Funkcje przynależności w regulatorze FDMC2

a) b)



Rys. 4.34. Odpowiedzi układów regulacji z regulatorami DMC i FDMC2 na skoki wartości zadanej z $C_B^{zad} = 1{,}12$ do $C_B^{zad} = 1$ i $C_B^{zad} = 1{,}2$;
a) przebiegi wyjścia C_B, b) przebiegi sterowania F

Zauważmy, że odpowiedź na skok wartości zadanej do $C_B^{zad} = 1{,}2$ otrzymana w układzie regulacji z regulatorem rozmytym FDMC2 (oznaczona kolorem czerwonym na rys. 4.34) jest szybsza zarówno od odpowiedzi uzyskanej z konwencjonalnym regulatorem DMC, jak i od odpowiedzi otrzymanej wcześniej w układzie z regulatorem FDMC1. Z drugiej strony odpowiedź układu regulacji z regulatorem FDMC2 na skok wartości zadanej do $C_B^{zad} = 1$ jest wolna, chociaż warto zauważyć, że w tym przypadku wartość wyjściowa dochodzi do wartości zadanej bez przeregulowania.

Regulator FDMC1 sprawdza się więc bardzo dobrze dla małych wartości zadanych, a regulator FDMC2 działa bardzo dobrze dla dużych wartości zadanych. W takim razie skorzystajmy z zalet podejścia rozmytego i opracujmy kombinację obydwu regulatorów rozmytych zakładając, że wagi dla poszczególnych reguł w nowym regulatorze FDMC3 są obliczane w następujący sposób:

$$w_1^{FDMC3} = w_1^{FDMC1},$$

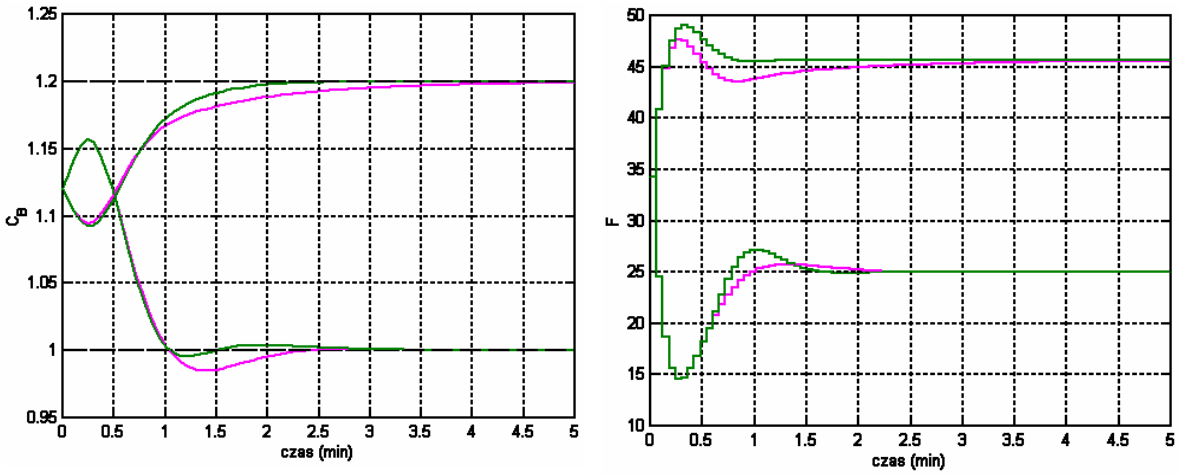
$$w_2^{FDMC3} = \frac{w_2^{FDMC1} + w_2^{FDMC2}}{2},$$

$$w_3^{FDMC3} = w_3^{FDMC2},$$

gdzie w_i^{FDMCx} ($i = 1, 2, 3$, $x = 1, 2, 3$) jest wagą dla i–tego regulatora lokalnego w regulatorze FDMCx. Następnie otrzymane w powyższy sposób wagi są normalizowane.

Odpowiedzi układu regulacji z regulatorem FDMC3 otrzymane w wyniku przeprowadzenia takiego samego eksperymentu, jak w poprzednich przypadkach, zostały zaprezentowane na rys. 4.35 (oznaczone kolorem zielonym). Zauważmy, że regulator FDMC3 działa najlepiej spośród wszystkich testowanych regulatorów. Odpowiedzi zarówno na zwiększenie wartości zadanej do $C_B^{zad} = 1{,}2$ jak i jej zmniejszenie do $C_B^{zad} = 1$ są bardzo dobre. W przypadku skoku wartości zadanej do $C_B^{zad} = 1$, czas regulacji jest znacznie mniejszy niż w przypadku otrzymanym w układzie regulacji z regulatorem FDMC2 a przeregulowanie jest wyraźnie mniejsze niż to otrzymane z regulatorem FDMC1.

a) b)



Rys. 4.35. Odpowiedzi układów regulacji z regulatorami DMC i FDMC3 na skoki wartości zadanej z $C_B^{zad} = 1{,}12$ do $C_B^{zad} = 1$ i $C_B^{zad} = 1{,}2$;
a) przebiegi wyjścia C_B, b) przebiegi sterowania F

Regulator rozmyty GPC

Załóżmy, że dysponujemy rozmytym modelem obiektu takim, jak w przykładzie 4.6, czyli z modelami lokalnymi w postaci równań różnicowych. Przypomnijmy, że model taki ma postać:

Reguła i: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = b_1^i \cdot y_k + \ldots + b_n^i \cdot y_{k-n+1} + c_1^i \cdot u_k + \ldots + c_m^i \cdot u_{k-m+1}.$$

Skorzystajmy z podejścia PDC. Dla każdego z modeli lokalnych w rozmytym modelu obiektu należy więc dobrać analityczny regulator GPC typu (2.79), dla założonych wartości horyzontu predykcji N, horyzontu sterowania N_u oraz parametru dostrajalnego λ. Załóżmy (jak to się czyni w podejściu PDC), że poprzedniki reguł w regulatorze są takie same, jak w modelu obiektu. W takim razie analityczny regulator rozmyty GPC typu Takagi–Sugeno jest opisany następującym wzorem:

Reguła *i*: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$\Delta u(k) = \left(k^e\right)^i \cdot e(k) + \sum_{j=1}^{m-1} \left(k_j^u\right)^i \cdot \Delta u(k-j) + \sum_{j=1}^{n+1} \left(k_j^y\right)^i \cdot y(k-j+1) \tag{4.77}$$

gdzie $\left(k^e\right)^i, \left(k_1^u\right)^i, \ldots, \left(k_{m-1}^u\right)^i, \left(k_1^y\right)^i, \ldots, \left(k_{n+1}^y\right)^i$ $(i = 1,\ldots l)$ są parametrami lokalnych regulatorów GPC. Wyjście analitycznego regulatora rozmytego GPC jest więc dane wzorem:

$$\Delta u(k) = \tilde{k}^e \cdot e(k) + \sum_{j=1}^{m-1} \tilde{k}_j^u \cdot \Delta u(k-j) + \sum_{j=1}^{n+1} \tilde{k}_j^y \cdot y(k-j+1), \tag{4.78}$$

gdzie

$$\tilde{k}^e = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(k^e\right)^i, \; \tilde{k}_j^u = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(\tilde{k}_j^u\right)^i, \; \tilde{k}_j^y = \sum_{i=1}^{l} \tilde{w}_i \cdot \left(\tilde{k}_j^y\right)^i.$$

Wartość sterowania można obliczyć ze wzoru:

$$u(k) = u(k-1) + \tilde{k}^e \cdot e(k) + \sum_{j=1}^{m-1} \tilde{k}_j^u \cdot \Delta u(k-j) + \sum_{j=1}^{n+1} \tilde{k}_j^y \cdot y(k-j+1). \tag{4.79}$$

Uwaga: Zauważmy, że w analitycznych, rozmytych regulatorach predykcyjnych, w każdym z regulatorów lokalnych wyjściem może być nie przyrost sterowania a bezpośrednio wartość sterowania. Wówczas stosownej zmianie ulegnie postać regulatorów lokalnych. Jest to jednak tylko zmiana zapisu a regulator będzie działał w obu przypadkach tak samo.

4.4.4. Badanie stabilności układów regulacji typu Takagi–Sugeno

Warunkiem koniecznym stabilności projektowanego układu regulacji jest zapewnienie stabilności układów zamkniętych złożonych z par model lokalny – regulator lokalny, w ogólnym przypadku, dla każdej z możliwych par tego typu. Nie jest to jednak niestety warunek wystarczający. Podstawowy, wystarczający warunek stabilności układów regulacji z modelami i regulatorami Takagi–Sugeno to kryterium Tanaki–Sugeno wynikające z metody Lapunowa.

Rozważmy najpierw autonomiczny, dynamiczny model Takagi–Sugeno. Ma on postać następujących reguł:

Reguła *i*: jeśli $x_1(k)$ jest X_1^i i … i $x_n(k)$ jest X_n^i, to

$$\boldsymbol{x}_{k+1}^i = \boldsymbol{A}_i \cdot \boldsymbol{x}_k. \tag{4.80}$$

W takim razie wyjście modelu Takagi–Sugeno (4.80) jest dane wzorem:

$$\boldsymbol{x}_{k+1} = \tilde{w}_i \cdot \boldsymbol{A}_i \cdot \boldsymbol{x}_k. \tag{4.81}$$

Zauważmy, że układ regulacji z obiektem opisanym modelem Takagi–Sugeno oraz regulatorem typu Takagi–Sugeno łatwo można przekształcić do postaci (4.80). W przypadku użycia modeli typu wejście–wyjście oraz sprzężenia od wyjścia, należy wektor stanu układu zastąpić wektorem quasi–stanu złożonym z wartości wyjść i sterowań w przeszłości, więcej informacji na ten temat czytelnik znajdzie np. w pracy [20] oraz w dalszej części rozdziału.

Kryterium Tanaki–Sugeno

Układ rozmyty (4.80) jest asymptotycznie stabilny jeśli istnieje dodatnio określona macierz $\boldsymbol{P}$ taka, że dla wszystkich macierzy $\boldsymbol{A}_i$ spełnione są następujące nierówności:

$$\left(\boldsymbol{A}_i\right)^T \cdot \boldsymbol{P} \cdot \boldsymbol{A}_i - \boldsymbol{P} < 0, \quad (i = 1,\ldots,l). \tag{4.82}$$

W praktyce, macierz $\boldsymbol{P}$ otrzymuje się w wyniku rozwiązania układu nierówności macierzowych (4.82). W programie Matlab służy do tego LMI toolbox (LMI – Linear Matrix Inequalities).

Uwaga: W przypadku nieosobliwych macierzy $\boldsymbol{A}_i$ warunkiem koniecznym istnienia dodatnio określonej macierzy $\boldsymbol{P}$ z kryterium Tanaki–Sugeno jest to, aby każda z macierzy $\boldsymbol{A}_i \cdot \boldsymbol{A}_j$ ($i = 1,\ldots, l, j = 1,\ldots, l$) miała wartości własne o modułach mniejszych od jedności.

Przykład 4.14

Rozważmy model rozmyty Takagi–Sugeno złożony z następujących reguł:

Reguła 1: jeśli $x_1(k)$ jest X_1^1, to

$$x^1(k+1) = \boldsymbol{A}_1 \cdot x(k), \tag{4.83}$$

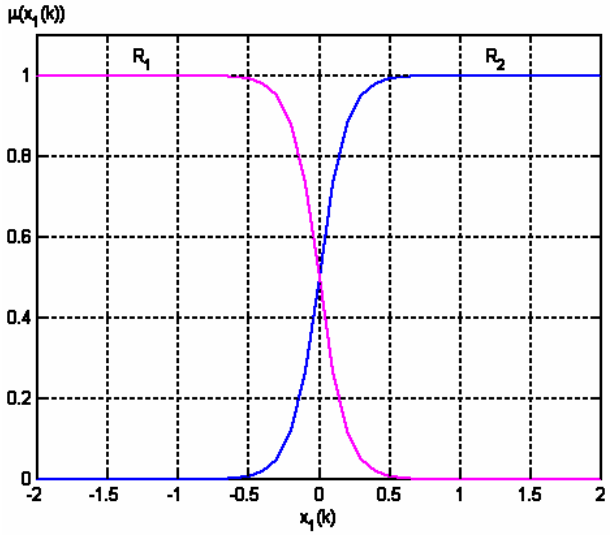
Reguła 2: jeśli $x_1(k)$ jest X_1^2, to

$$x^2(k+1) = \boldsymbol{A}_2 \cdot x(k), \tag{4.84}$$

gdzie

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & -0{,}3 \\ 1 & 0 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} -1 & -0{,}3 \\ 1 & 0 \end{bmatrix}.$$

Załóżmy, że funkcje przynależności są takie, jak na rys. 4.36.



Rys. 4.36. Funkcje przynależności przykładowego modelu

Zauważmy, że obydwie macierze $\boldsymbol{A}_1$ i $\boldsymbol{A}_2$ mają wartości własne o modułach mniejszych od jedności (w przypadku macierzy $\boldsymbol{A}_1$ wartości własne to $a_{1,2} = 0{,}5000 \pm 0{,}2236i$ a w przypadku macierzy $\boldsymbol{A}_2$ – $a_{1,2} = -0{,}5000 \pm 0{,}2236i$). Sprawdźmy teraz wartości własne macierzy $\boldsymbol{A}_1 \cdot \boldsymbol{A}_2$.

Macierz ta ma następującą postać:

$$\boldsymbol{A}_1 \cdot \boldsymbol{A}_2 = \begin{bmatrix} -1{,}3 & -0{,}3 \\ -1 & -0{,}3 \end{bmatrix}.$$

Macierz ta ma wartości własne: $a_1 = -1{,}5416$ oraz $a_2 = -0{,}0584$. W takim razie dodatnio określona macierz $\boldsymbol{P}$ z kryterium Tanaki–Sugeno nie istnieje.

Przykład 4.15

Rozważmy tym razem model rozmyty Takagi–Sugeno złożony z następujących reguł:

Reguła 1: jeśli $x_1(k)$ jest X_1^1, to

$$x^1(k+1) = \boldsymbol{A}_1 \cdot x(k), \tag{4.85}$$

Reguła 2: jeśli $x_1(k)$ jest X_1^2, to

$$x^2(k+1) = \boldsymbol{A}_2 \cdot x(k), \tag{4.86}$$

gdzie

$$\boldsymbol{A}_1 = \begin{bmatrix} 0{,}5 & 1 \\ -0{,}4 & 0 \end{bmatrix}, \boldsymbol{A}_2 = \begin{bmatrix} -0{,}5 & 1 \\ -0{,}4 & 0 \end{bmatrix}.$$

Załóżmy, że funkcje przynależności są takie same, jak w przykładzie 4.14.

Wartości własne macierzy $\boldsymbol{A}_1$ to: $a_{1,2} = 0{,}2500 \pm 0{,}5809i$ a macierzy $\boldsymbol{A}_2$ to: $a_{1,2} = -0{,}2500 \pm 0{,}5809i$. Macierz $\boldsymbol{A}_1 \cdot \boldsymbol{A}_2$ ma tym razem postać:

$$\boldsymbol{A}_1 \cdot \boldsymbol{A}_2 = \begin{bmatrix} -0{,}65 & 0{,}5 \\ 0{,}2 & -0{,}4 \end{bmatrix}$$

i wartości własne: $a_1 = -0{,}8650$ oraz $a_2 = -0{,}1850$. W takim razie spróbujmy znaleźć dodatnio określoną macierz $\boldsymbol{P}$ z kryterium Tanaki–Sugeno. Przykład takiej macierzy to:

$$\boldsymbol{P} = \begin{bmatrix} 1{,}0050 & 0 \\ 0 & 2{,}1328 \end{bmatrix}.$$

Badany model jest więc asymptotycznie stabilny.

Rozpatrzmy teraz układ regulacji ze sprzężeniem od stanu. Układ regulacji złożony z obiektu opisanego modelem rozmytym Takagi–Sugeno i regulatora ze sprzężeniem od stanu jest opisany następującym wzorem (po podstawieniu zależności (4.58) do (4.56)):

$$\boldsymbol{x}_{k+1} = \sum_{i=1}^{lo} \sum_{j=1}^{l} \tilde{w}_i \cdot \tilde{w}_j \cdot \boldsymbol{D}_{ij} \cdot \boldsymbol{x}_k, \tag{4.87}$$

gdzie $\boldsymbol{D}_{ij} = \boldsymbol{A}_i + \boldsymbol{B}_i \cdot \boldsymbol{K}_j$; tym razem założono, że model oraz regulator mogą mieć różną postać poprzedników, *lo* jest więc liczbą modeli lokalnych w rozmytym modelu obiektu. W celu zbadania stabilności otrzymanego układu regulacji, należy użyć kryterium Tanaki–Sugeno badając tym razem właściwości macierzy $\boldsymbol{D}_{ij}$, tzn. należy znaleźć dodatnio określoną macierz $\boldsymbol{P}$ spełniającą warunki:

$$\left(\boldsymbol{D}_{ij}\right)^T \cdot \boldsymbol{P} \cdot \boldsymbol{D}_{ij} - \boldsymbol{P} < 0, \quad (i = 1,\ldots,lo,\ j = 1,\ldots,l). \tag{4.88}$$

Uwaga: Jeśli korzysta się z podejścia PDC i poprzedniki w regulatorze rozmytym są takie same, jak w modelu obiektu wówczas *lo* = *l* oraz

$$\tilde{w}_i \cdot \tilde{w}_j = \tilde{w}_j \cdot \tilde{w}_i, \quad (i = 1,\ldots,l,\ j = 1,\ldots,l) \tag{4.89}$$

a w takim razie zamiast analizowania macierzy $\boldsymbol{D}_{ij}$ oraz $\boldsymbol{D}_{ji}$ (dla i ≠ j) można analizować macierze

$$\boldsymbol{D}_{ij}^{*} = \frac{1}{2}\left((\boldsymbol{A}_i + \boldsymbol{B}_i \cdot \boldsymbol{K}_j) + (\boldsymbol{A}_j + \boldsymbol{B}_j \cdot \boldsymbol{K}_i)\right), \quad i < j,\ (i, j = 1,\ldots,l). \tag{4.90}$$

Zauważmy, że zmniejsza się zatem stopień skomplikowania układu nierówności macierzowych, który musi być rozwiązany w celu znalezienia dodatnio określonej macierzy $\boldsymbol{P}$, co ma znaczenie zwłaszcza w przypadku gdy mamy do czynienia z dużą liczbą modeli i regulatorów lokalnych.

Przykład 4.16

Wróćmy do przykładu 4.11. Zbadamy stabilność otrzymanego tam układu regulacji ze sprzężeniem od stanu. Przypomnijmy, że macierze opisujące obiekt są następujące:

$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & -0{,}6 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 1 & -0{,}4 \\ 1 & 0 \end{bmatrix}, \quad \boldsymbol{B} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

zaś wektory sprzężeń stanu:

$$\boldsymbol{K}_1 = \begin{bmatrix} -0{,}1 & 0{,}4 \end{bmatrix}, \quad \boldsymbol{K}_2 = \begin{bmatrix} -0{,}1 & 0{,}2 \end{bmatrix}.$$

W takim razie macierze opisujące układ zamknięty mają postać:

$$\boldsymbol{D}_{11} = \boldsymbol{A}_1 + \boldsymbol{B} \cdot \boldsymbol{K}_1 = \begin{bmatrix} 0{,}9 & -0{,}2 \\ 1 & 0 \end{bmatrix},$$

$$\boldsymbol{D}_{22} = \boldsymbol{A}_2 + \boldsymbol{B} \cdot \boldsymbol{K}_2 = \begin{bmatrix} 0{,}9 & -0{,}2 \\ 1 & 0 \end{bmatrix}$$

oraz ponieważ założono w regulatorze takie same poprzedniki, jak w modelu obiektu, to wystarczy rozpatrzeć jeszcze jedną macierz:

$$\boldsymbol{D}_{12}^{*} = \frac{1}{2}\left(\boldsymbol{A}_1 + \boldsymbol{B} \cdot \boldsymbol{K}_2 + \boldsymbol{A}_2 + \boldsymbol{B} \cdot \boldsymbol{K}_1\right) = \begin{bmatrix} 0{,}9 & -0{,}2 \\ 1 & 0 \end{bmatrix}.$$

Zauważmy, że wszystkie trzy powyższe macierze mają wartości własne o modułach mniejszych od jedności. Następnym krokiem było znalezienie dodatnio określonej macierzy $\boldsymbol{P}$ dla badanego układu. Przykładowa macierz ma w rozpatrywanym przypadku postać:

$$\boldsymbol{P} = 10^8 \cdot \begin{bmatrix} 7{,}9497 & -1{,}1924 \\ -1{,}1924 & 1{,}9874 \end{bmatrix}.$$

Badany układ regulacji jest więc stabilny.

Rozpatrzymy teraz sposób badania stabilności układów z regulatorami ze sprzężeniem od wyjścia obiektu. W takim przypadku, jak już wspomniano przy omawianiu kryterium Tanaki–Sugeno, należy wektor stanu układu zastąpić wektorem quasi–stanu złożonym z wartości wyjść i sterowań z przeszłych chwil. Zastosowanie tej metody prześledzimy na przykładzie.

Przykład 4.17

Powróćmy tym razem do przykładu 4.12. Zbadamy stabilność układów regulacji z regulatorami PI, które zostały tam zaprojektowane. Przypomnijmy, że modele lokalne w rozmytym modelu obiektu mają postać:

$$y^i(k+1) = -a^i \cdot y(k) + b^i \cdot u(k),\ i = 1, 2; \tag{4.91}$$

wartości parametrów są następujące: $a^1 = -0{,}8$, $b^1 = 2$, $a^2 = -0{,}5$, $b^2 = 1$. Z kolei regulatory lokalne są opisane wzorem:

$$u^j(k) = u(k-1) + r_1^j \cdot e(k) + r_2^j \cdot e(k-1),\ j = 1, 2; \tag{4.92}$$

a wartości parametrów w przypadku a) (z biegunami układów zamkniętych ulokowanych w punkcie 0,5):

$$r_1^1 = 0{,}4;\ r_2^1 = -0{,}275;\ r_1^2 = 0{,}5;\ r_2^2 = -0{,}25;$$

oraz w przypadku b) (z biegunami położonymi w punkcie 0,2):

$$r_1^1 = 0{,}7;\ r_2^1 = -0{,}38;\ r_1^2 = 1{,}1;\ r_2^2 = -0{,}46.$$

Zauważmy, że równania regulatorów lokalnych można zapisać w postaci:

$$u^j(k) = u(k-1) + r_1^j \cdot y^{zad}(k) + r_2^j \cdot y^{zad}(k-1) - r_1^j \cdot y(k) - r_2^j \cdot y(k-1),\ j = 1, 2; \tag{4.93}$$

W celu zbadania stabilności układu regulacji wprowadźmy następujący wektor quasi–stanu:

$$x^*(k) = \begin{bmatrix} y(k) & y(k-1) & u(k-1) \end{bmatrix}. \tag{4.94}$$

W takim razie należy analizować macierze opisujące zamknięty układ regulacji o następującej postaci (wynikającej z połączenia poszczególnych modeli lokalnych z regulatorami lokalnymi, polegającego na wstawieniu równania (4.93) do (4.91)):

$$\boldsymbol{D}_{ij} = \begin{bmatrix} -a^i - b^i \cdot r_1^j & -b^i \cdot r_2^j & b^i \\ 1 & 0 & 0 \\ -r_1^j & -r_2^j & 1 \end{bmatrix}. \tag{4.95}$$

Zauważmy, że pierwszy wiersz macierzy $\boldsymbol{D}_{ij}$ ze wzoru (4.95) opisuje układ zamknięty otrzymany w wyniku połączenia *i*–tego modelu lokalnego z *j*–tym regulatorem lokalnym a ostatni wiersz – *j*–ty regulator lokalny.

Zbadajmy stabilność układu regulacji z regulatorem PI z biegunami układu zamkniętego położonymi w 0,5 (przypadek a)). Wówczas poszczególne macierze mają postać:

$$\boldsymbol{D}_{11} = \begin{bmatrix} 0 & 0{,}55 & 2 \\ 1 & 0 & 0 \\ -0{,}4 & 0{,}275 & 1 \end{bmatrix},$$

$$\boldsymbol{D}_{22} = \begin{bmatrix} 0 & 0{,}25 & 1 \\ 1 & 0 & 0 \\ -0{,}5 & 0{,}25 & 1 \end{bmatrix},$$

$$\boldsymbol{D}_{12} = \begin{bmatrix} -0{,}2 & 0{,}5 & 2 \\ 1 & 0 & 0 \\ -0{,}5 & 0{,}25 & 1 \end{bmatrix}, \quad \boldsymbol{D}_{21} = \begin{bmatrix} 0{,}1 & 0{,}275 & 1 \\ 1 & 0 & 0 \\ -0{,}4 & 0{,}275 & 1 \end{bmatrix}.$$

Zauważmy jednak, że w regulatorze przyjęto takie same poprzedniki, jak w modelu obiektu. Dlatego zamiast rozpatrywać macierze $\boldsymbol{D}_{12}$ i $\boldsymbol{D}_{21}$ z osobna, można wziąć pod uwagę jedną macierz:

$$\boldsymbol{D}_{12}^{*} = \begin{bmatrix} -0{,}05 & 0{,}3875 & 1{,}5 \\ 1 & 0 & 0 \\ -0{,}45 & 0{,}2625 & 1 \end{bmatrix}.$$

Przykład macierzy $\boldsymbol{P}$ spełniającej warunki kryterium Tanaki–Sugeno jest następujący:

$$\boldsymbol{P} = \begin{bmatrix} 1{,}3478 & -0{,}6244 & -1{,}4590 \\ -0{,}6244 & 0{,}9164 & 1{,}6468 \\ -1{,}4590 & 1{,}6468 & 6{,}8192 \end{bmatrix}.$$

Badany układ regulacji jest więc stabilny.

W przypadku układu regulacji z regulatorem PI z biegunami układu zamkniętego położonymi w punkcie 0,2 (przypadek b)), poszczególne macierze mają postać:

$$\boldsymbol{D}_{11} = \begin{bmatrix} -0{,}6 & 0{,}76 & 2 \\ 1 & 0 & 0 \\ -0{,}7 & 0{,}38 & 1 \end{bmatrix},$$

$$\boldsymbol{D}_{22} = \begin{bmatrix} -0{,}6 & 0{,}46 & 1 \\ 1 & 0 & 0 \\ -1{,}1 & 0{,}46 & 1 \end{bmatrix},$$

$$\boldsymbol{D}_{12}^{*} = \begin{bmatrix} -0{,}8 & 0{,}65 & 1{,}5 \\ 1 & 0 & 0 \\ -0{,}9 & 0{,}42 & 1 \end{bmatrix}.$$

Przykładowa dodatnio określona macierz $\boldsymbol{P}$ to:

$$\boldsymbol{P} = \begin{bmatrix} 34{,}9231 & -17{,}4116 & -36{,}7059 \\ -17{,}4116 & 19{,}8015 & 33{,}0615 \\ -36{,}7059 & 33{,}0615 & 82{,}6820 \end{bmatrix}.$$

Badany układ regulacji jest więc również stabilny.

4.4.5. Regulatory predykcyjne w wersji numerycznej

W rozdz. 2.4.3 przedstawiono ogólną ideę wykorzystania modeli nieliniowych w efektywnych obliczeniowo numerycznych algorytmach predykcyjnych. Przedstawiono przy tym ogólną zasadę działania algorytmów z sukcesywną linearyzacją (MPC–SL) oraz algorytmów z nieliniową predykcją i sukcesywną linearyzacją (MPC–NPL). W niniejszym rozdziale omówiono szczegóły dotyczące użycia w tego typu algorytmach rozmytych modeli Takagi–Sugeno. Warto już na wstępie zauważyć, że algorytmy regulacji predykcyjnej oparte na linearyzacji i wykorzystujące rozważane modele rozmyte są naturalnym rozszerzeniem konwencjonalnych algorytmów regulacji predykcyjnej. Ponadto ich sformułowanie jest stosunkowo proste i intuicyjne.

Rozmyte algorytmy regulacji predykcyjnej z sukcesywną linearyzacją (FMPC–SL)

Algorytm FDMC–SL

Tak, jak w przypadku rozmytego analitycznego regulatora DMC załóżmy, że rozmyty model Takagi–Sugeno obiektu regulacji ma modele lokalne w postaci odpowiedzi skokowych, czyli jest złożony z następujących reguł:

Reguła *i*: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = \sum_{j=1}^{D-1} s_j^i \cdot \Delta u_{k-j+1} + s_D^i \cdot u_{k-D+1}, \tag{4.96}$$

gdzie s_j^i ($i = 1,\ldots,l, j = 1,\ldots,D$) są współczynnikami modeli lokalnych.

W pierwszej iteracji algorytmu FDMC–SL należy otrzymać liniową aproksymację modelu nieliniowego. W przypadku modelu rozmytego jest to szczególnie łatwe, ponieważ wystarczy skorzystać z mechanizmu wnioskowania rozmytego. Zauważmy, że wyjście modelu rozmytego (4.96) jest opisane następującym wzorem:

$$y_{k+1}^M = \sum_{i=1}^{D-1} \tilde{s}_i \cdot \Delta u_{k-i+1} + \tilde{s}_D \cdot u_{k-D+1}, \tag{4.97}$$

gdzie $\tilde{s}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot s_j^i$ ($j = 1,\ldots, D$) można interpretować jako rzędne odpowiedzi skokowej obiektu modelującej zachowanie tego obiektu w okolicy bieżącego punktu pracy. Dysponując tymi rzędnymi, można otrzymać odpowiedź swobodną oraz macierz dynamiczną w taki sam sposób, jak w klasycznym algorytmie DMC. Ostatecznie otrzyma się więc predykcję wartości wyjścia obiektu w postaci:

$$\boldsymbol{y} = \tilde{\boldsymbol{y}}^0 + \boldsymbol{M}_k \cdot \Delta \boldsymbol{u}, \tag{4.98}$$

gdzie odpowiedź swobodna jest dana następującym wzorem:

$$\tilde{\boldsymbol{y}}^0 = \boldsymbol{y}_k + \boldsymbol{M}_k^P \cdot \Delta \boldsymbol{u}^P, \tag{4.99}$$

gdzie $\boldsymbol{y}_k = [y_k,\ldots, y_k]^{\mathrm{T}}$ jest wektorem N–elementowym, $\Delta \boldsymbol{u}^P = [\Delta u_{k-1},\ldots, \Delta u_{k-D+1}]^{\mathrm{T}}$ oraz

$$\boldsymbol{M}_k^P = \begin{bmatrix} \tilde{s}_2 - \tilde{s}_1 & \tilde{s}_3 - \tilde{s}_2 & \cdots & \tilde{s}_{D-1} - \tilde{s}_{D-2} & \tilde{s}_D - \tilde{s}_{D-1} \\ \tilde{s}_3 - \tilde{s}_1 & \tilde{s}_4 - \tilde{s}_2 & \cdots & \tilde{s}_D - \tilde{s}_{D-2} & \tilde{s}_D - \tilde{s}_{D-1} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{s}_{N+1} - \tilde{s}_1 & \tilde{s}_{N+2} - \tilde{s}_2 & \cdots & \tilde{s}_D - \tilde{s}_{D-2} & \tilde{s}_D - \tilde{s}_{D-1} \end{bmatrix}. \tag{4.100}$$

Zwróćmy uwagę na to, że tym razem macierz $\boldsymbol{M}_k^P$ będzie się zmieniała w każdej iteracji algorytmu, co zostało zaznaczone przez dodanie dolnego indeksu czasu dyskretnego k w oznaczeniu tej macierzy. Jest tak, ponieważ w każdej iteracji otrzyma się nowe wartości zmiennych $\tilde{s}_i$ ($i = 1,\ldots, D$). Podobna sytuacja dotyczy macierzy dynamicznej $\boldsymbol{M}_k$, która tym razem jest opisana wzorem:

$$\boldsymbol{M}_k = \begin{bmatrix} \tilde{s}_1 & 0 & \cdots & 0 & 0 \\ \tilde{s}_2 & \tilde{s}_1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \tilde{s}_N & \tilde{s}_{N-1} & \cdots & \tilde{s}_{N-N_u+2} & \tilde{s}_{N-N_u+1} \end{bmatrix}. \tag{4.101}$$

Następnie, jeśli rozwiązuje się zadanie optymalizacji bez ograniczeń, można skorzystać ze wzoru używanego w algorytmach regulacji predykcyjnej w wersji analitycznej:

$$\Delta\boldsymbol{u} = \left(\boldsymbol{M}_k^T \cdot \boldsymbol{M}_k + \lambda \cdot \boldsymbol{I}\right)^{-1} \cdot \boldsymbol{M}_k^T \cdot \left(\boldsymbol{y}^{zad} - \tilde{\boldsymbol{y}}^0\right). \tag{4.102}$$

Należy jednak podkreślić, że tym razem z powyższego wzoru należy korzystać w każdej iteracji algorytmu, ze względu na zmiany macierzy $\boldsymbol{M}_k^P$ oraz $\boldsymbol{M}_k$.

W przypadku problemu z ograniczeniami, predykcję (4.98) można wykorzystać do sformułowania problemu optymalizacji liniowo–kwadratowej, który będzie miał w takim razie następującą postać:

$$\min_{\Delta\boldsymbol{u}} \left\{ \left(\boldsymbol{y}^{zad} - \tilde{\boldsymbol{y}}^0 - \boldsymbol{M}_k \cdot \Delta\boldsymbol{u}\right)^T \cdot \left(\boldsymbol{y}^{zad} - \tilde{\boldsymbol{y}}^0 - \boldsymbol{M}_k \cdot \Delta\boldsymbol{u}\right) + \lambda \cdot \Delta\boldsymbol{u}^T \cdot \Delta\boldsymbol{u} \right\}, \tag{4.103}$$

przy ograniczeniach:

$$\Delta\boldsymbol{u}_{\min} \le \Delta\boldsymbol{u} \le \Delta\boldsymbol{u}_{\max}, \tag{4.104}$$

$$\boldsymbol{u}_{\min} \le \boldsymbol{u}_{k-1} + \boldsymbol{J} \cdot \Delta\boldsymbol{u} \le \boldsymbol{u}_{\max}, \tag{4.105}$$

$$\boldsymbol{y}_{\min} \le \tilde{\boldsymbol{y}}^0 + \boldsymbol{M}_k \cdot \Delta\boldsymbol{u} \le \boldsymbol{y}_{\max}. \tag{4.106}$$

Algorytm FGPC–SL

Załóżmy, że obiekt regulacji jest opisany rozmytym modelem Takagi–Sugeno takim, jak w przykładzie 4.6, czyli z modelami lokalnymi w postaci równań różnicowych. Model ten jest więc złożony z następujących reguł:

Reguła i: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = b_1^i \cdot y_k + \ldots + b_n^i \cdot y_{k-n+1} + c_1^i \cdot u_k + \ldots + c_m^i \cdot u_{k-m+1}.$$

Tym razem wyjście modelu rozmytego jest więc opisane następującym wzorem:

$$y_{k+1}^{M} = \tilde{b}_1 \cdot y_k + \ldots + \tilde{b}_n \cdot y_{k-n+1} + \tilde{c}_1 \cdot u_k + \ldots + \tilde{c}_m \cdot u_{k-m+1}, \tag{4.107}$$

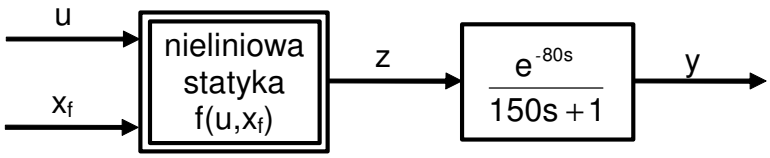
gdzie $\tilde{b}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot b_j^i$ (j=1,…,n), $\tilde{c}_j = \sum_{i=1}^{l} \tilde{w}_i \cdot c_j^i$ (j=1,…,m) można interpretować jako parametry liniowego modelu obiektu otrzymanego w okolicach bieżącego punktu pracy. Dysponując takim modelem, w bieżącej iteracji należy wyznaczyć odpowiedź swobodną (korzystając ze stosownych wzorów lub z metody iteracyjnej) oraz macierz dynamiczną tak, jak w konwencjonalnym algorytmie GPC. Następnie w celu wyznaczenia sterowania, można skorzystać z analitycznego wzoru (4.102) w przypadku problemu bez ograniczeń lub sformułować i rozwiązać problem optymalizacji z ograniczeniami (4.103)–(4.106). Sposób postępowania jest więc analogiczny do użytego w przypadku algorytmu FDMC–SL.

Przykład 4.18

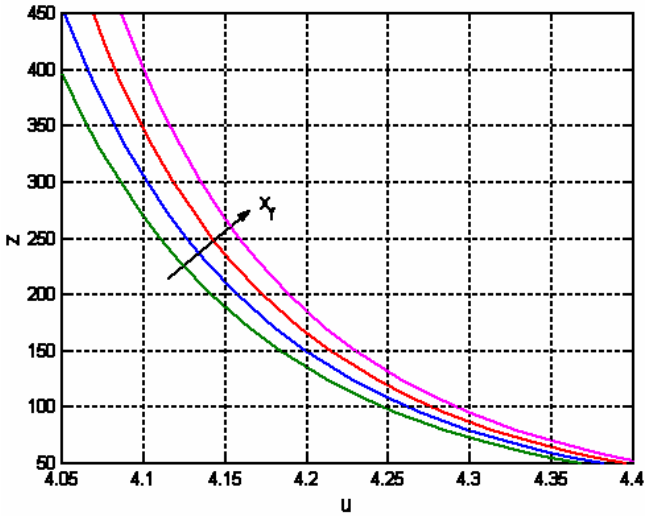
Działanie algorytmu FDMC–SL prześledźmy na przykładzie układu regulacji kolumny etylenowej rozważanego w pracy [20]. Model Hammersteina obiektu regulacji został pokazany na rys. 4.37. Zmienną wyjściową y jest zanieczyszczenie produktu liczone w ppm (liczbie cząstek na milion), zmienną sterującą u jest stosunek refluksu do produktu. Im większy jest refluks, tym mniej zanieczyszczony produkt jest otrzymywany; podczas eksperymentów założono, że sterowanie jest ograniczone:

$$4{,}05 \le u \le 4{,}4;$$

skład surowca x_f jest zakłóceniem. Stałe czasowe na rys. 4.37 podano w minutach. Zauważmy więc, że rozważany obiekt ma znaczne opóźnienie. Jest on ponadto silnie nieliniowy, co dobrze ilustrują charakterystyki statyczne, przedstawione na rys. 4.38.



Rys. 4.37. Model Hammersteina kolumny etylenowej



Rys. 4.38. Charakterystyki statyczne kolumny etylenowej dla różnych wartości zakłócenia x_f = 0.8011, x_f = 0.8064, x_f = 0.8118, x_f = 0.8171

Załóżmy okres próbkowania $T_p = 40$ min. Model rozmyty obiektu regulacji, przekształcony do standardowej postaci modelu Takagi–Sugeno, jest złożony z następujących trzech reguł:

Reguła 1: jeśli u_{k-2} jest U_1, to

$$y_{k+1}^1 = 0{,}7659 \cdot y_k - 520{,}2638 \cdot u_{k-2} + 2220{,}9067 ;$$

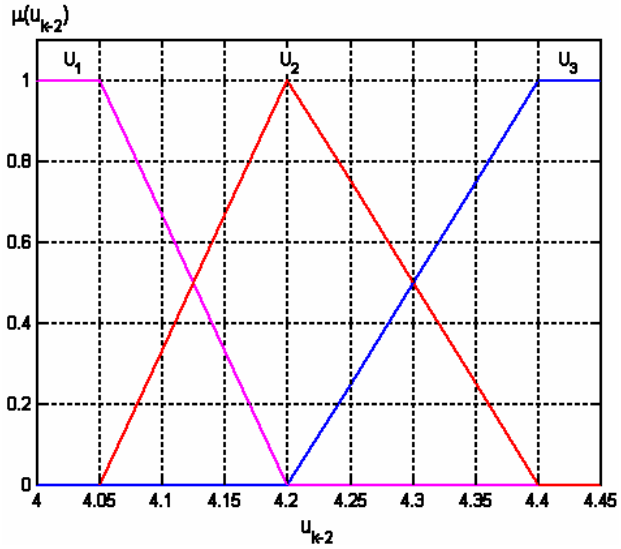
Reguła 2: jeśli u_{k-2} jest U_2, to

$$y_{k+1}^2 = 0{,}7659 \cdot y_k - 253{,}5771 \cdot u_{k-2} + 1102{,}4471 ;$$

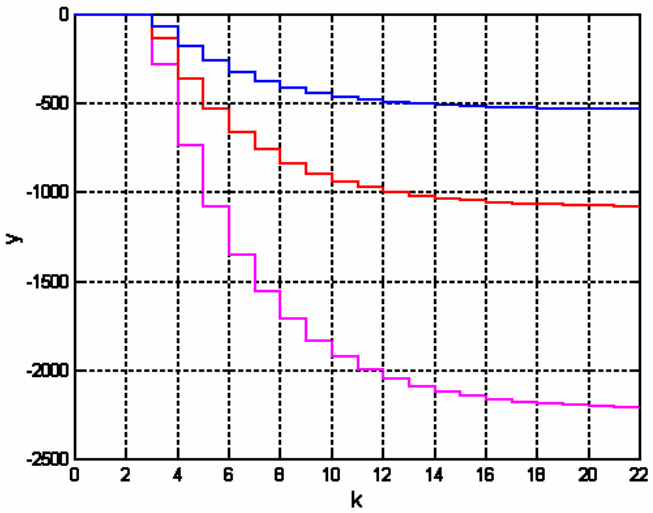
Reguła 3: jeśli u_{k-2} jest U_3, to

$$y_{k+1}^3 = 0{,}7659 \cdot y_k - 125{,}1030 \cdot u_{k-2} + 563{,}8767 .$$

Kształt przyjętych funkcji przynależności został pokazany na rys. 4.39 a znormalizowane odpowiedzi skokowe poszczególnych modeli lokalnych zostały przedstawione na rys. 4.40.

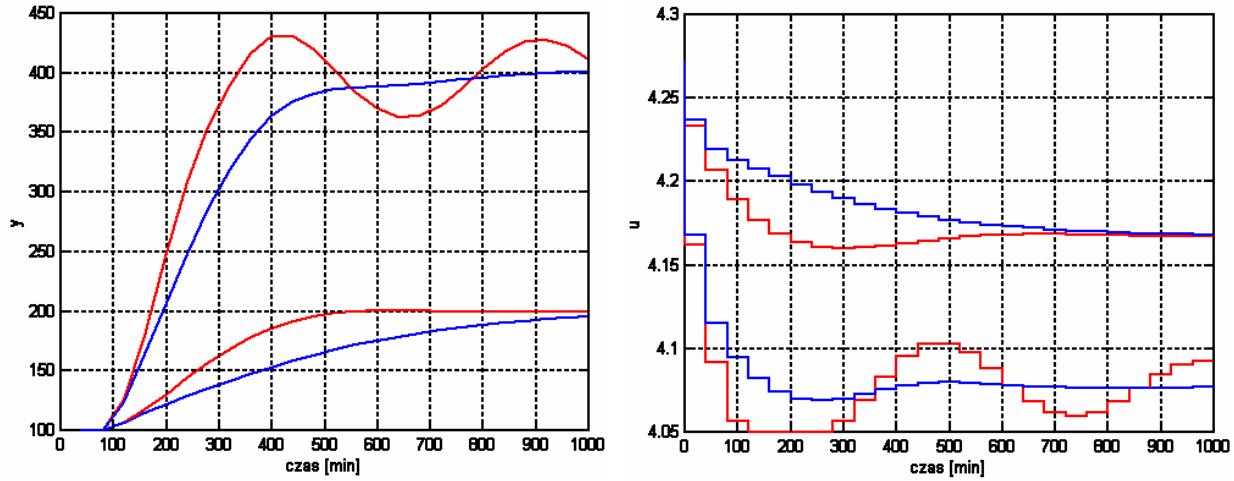


Rys. 4.39. Funkcje przynależności w rozmytym modelu kolumny etylenowej



Rys. 4.40. Znormalizowane odpowiedzi skokowe poszczególnych modeli lokalnych w rozmytym modelu Takagi–Sugeno kolumny etylenowej

a) b)

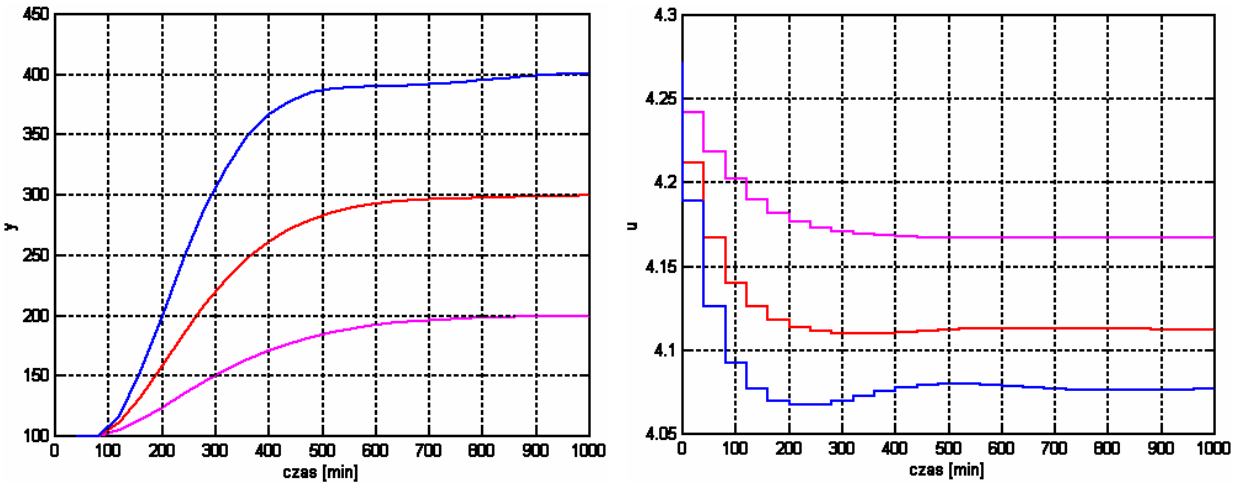


Rys. 4.41. Odpowiedzi układów regulacji z regulatorami DMC na skoki wartości zadanej z y_0=100 ppm; regulatory zaprojektowano dla odpowiedzi z okolicy małych i dużych zanieczyszczeń; a) przebiegi wyjścia y, b) przebiegi sterowania u

Najpierw przeprowadzono eksperymenty polegające na dobraniu do przykładowego obiektu konwencjonalnego regulatora DMC. Przyjęto przy tym horyzont predykcji $N = 22$ a horyzont sterowania $N_u = 10$. Przykładowe wyniki przeprowadzonych eksperymentów zostały zaprezentowane na rys. 4.41. Zauważmy, że gdy dobrano regulator działający dobrze w zakresie małych wartości zadanych, rzędu 200 ppm, działał on w sposób dalece niezadowalający dla dużych wartości zadanych, rzędu 400 ppm, ze względu na utrzymujące się oscylacje sygnału wyjściowego (czerwone przebiegi na rys. 4.41). Z kolei, gdy dobrano regulator działający w sposób akceptowalny dla dużych wartości zadanych, regulator ten pracował bardzo wolno dla małych wartości zadanych (niebieskie przebiegi na rys. 4.41).

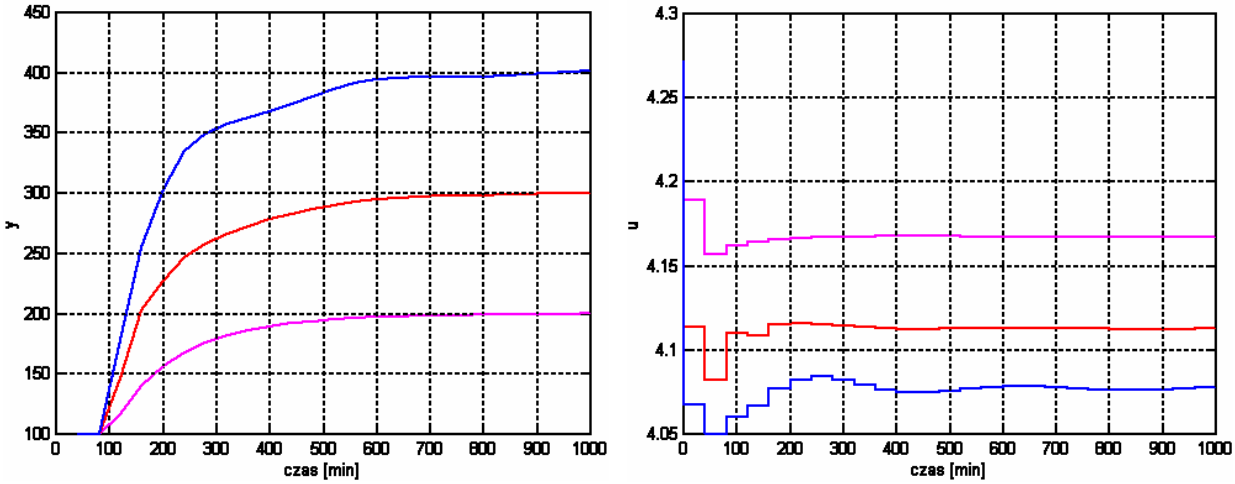
W celu poprawienia jakości regulacji, na podstawie otrzymanego zestawu trzech odpowiedzi skokowych, opracowano regulator FDMC–SL. Zastosowanie takiego regulatora do badanego obiektu przyniosło poprawę jakości regulacji (rys. 4.42). Charakter otrzymanych odpowiedzi jest podobny niezależnie od wielkości skoku wartości zadanej.

a) b)



Rys. 4.42. Odpowiedzi układu regulacji z regulatorem FDMC–SL na skoki wartości zadanej z y_0=100 ppm do y_{zad}= 200, 300, 400 ppm; $\lambda=9\times10^6$; a) przebiegi wyjścia y, b) przebiegi sterowania u

a) b)



Rys. 4.43. Odpowiedzi układu regulacji z regulatorem FDMC–SL na skoki wartości zadanej z y_0=100 ppm do y_{zad}= 200, 300, 400 ppm; $\lambda=9\times10^5$; a) przebiegi wyjścia y, b) przebiegi sterowania u

Spróbujmy przyspieszyć działanie układu regulacji. W tym celu zmniejszmy wartość parametru dostrajalnego λ dziesięciokrotnie, tj. do $\lambda=9\times10^5$. Odpowiedzi układu regulacji otrzymane po tym zabiegu zostały pokazane na rys. 4.43. Odpowiedź przy skoku wartości zadanej do 200 ppm jest co prawda szybsza, jednak w przypadku pozostałych skoków nie otrzymaliśmy znacznego skrócenia czasu regulacji a sygnał sterujący zmienia się na początku odpowiedzi gwałtownie.

Dalszą poprawę działania układu regulacji może przynieść zastosowanie algorytmów z wyznaczaniem odpowiedzi swobodnej obiektu przy użyciu nieliniowego modelu obiektu, co zostanie zademonstrowane za chwilę. Przedtem jednak krótko omówiony zostanie sposób generacji trajektorii swobodnej w tego typu algorytmach bazujących na modelach rozmytych typu Takagi–Sugeno.

Rozmyte algorytmy regulacji predykcyjnej z nieliniową predykcją i sukcesywną linearyzacją (FMPC–NPL)

Przypomnijmy, że algorytmy z nieliniową predykcją odpowiedzi swobodnej i sukcesywną linearyzacją są naturalnym rozwinięciem algorytmów z sukcesywną linearyzacją. Sposób wyznaczenia macierzy dynamicznej, a tym samym sposób obliczania odpowiedzi wymuszonej jest w tych algorytmach taki sam, jak w algorytmach z sukcesywną linearyzacją. Zmiana polega jedynie na innym sposobie obliczania odpowiedzi swobodnej obiektu regulacji. Jest to czynione iteracyjnie z wykorzystaniem nieliniowego, w rozważanym przypadku rozmytego, modelu obiektu i zależy od postaci tego modelu.

Algorytm FDMC–NPL

Rozpatrzmy tak, jak w algorytmie FDMC–SL rozmyty model Takagi–Sugeno z modelami lokalnymi w postaci odpowiedzi skokowych, zakładając, że zamierzamy wyznaczyć wyjście modelu dla bieżącej chwili k, czyli:

Reguła i: jeśli y_{k-1} jest B_1^i i … i y_{k-n} jest B_n^i i u_{k-1} jest C_1^i i … i u_{k-m} jest C_m^i to

$$y_k^i = \sum_{j=1}^{D-1} s_j^i \cdot \Delta u_{k-j} + s_D^i \cdot u_{k-D} . \tag{4.108}$$

W celu wyznaczenia trajektorii swobodnej zastosujemy teraz podejście iteracyjne, analogiczne to tego, które zostało zaproponowane dla algorytmu GPC w rozdz. 2. W tym celu najpierw obliczmy wartość $d_k = y_k - y_k^M$. Zauważmy, że wyjście modelu rozmytego w chwili k jest opisane następującym wzorem:

$$y_k^M = \sum_{i=1}^{D-1} \tilde{s}_i \cdot \Delta u_{k-i} + \tilde{s}_D \cdot u_{k-D}. \tag{4.109}$$

W takim razie otrzymamy:

$$d_k = y_k - \sum_{i=1}^{D-1} \tilde{s}_i \cdot \Delta u_{k-i} - \tilde{s}_D \cdot u_{k-D}. \tag{4.110}$$

Zakładając, że wpływ zakłóceń niemierzalnych oraz błędów modelowania na całym horyzoncie predykcji nie zmieni się (rozsądne założenie jeśli nie dysponujemy innym modelem zakłóceń), wyznaczamy poszczególne elementy trajektorii swobodnej korzystając z zależności:

$$y_{k+i|k}^0 = y_{k+i}^M + d_k; (i = 1,\ldots, N), \tag{4.111}$$

przy czym składniki y_{k+i}^M otrzymujemy iteracyjnie korzystając z rozmytego modelu procesu i zakładając, że przyszłe przyrosty sygnału sterującego są równe $\Delta u_{k+i|k}=0$ ($i = 1,\ldots, N_u$). W takim razie w celu wyznaczenia elementu trajektorii swobodnej dla chwili k+1, korzystamy z następującego modelu:

Reguła i: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = \sum_{j=2}^{D-1} s_j^i \cdot \Delta u_{k-j+1} + s_D^i \cdot u_{k-D+1}. \tag{4.112}$$

W takim razie

$$y_{k+1}^M = \sum_{i=2}^{D-1} \tilde{s}_i^1 \cdot \Delta u_{k-i+1} + \tilde{s}_D^1 \cdot u_{k-D+1}, \tag{4.113}$$

gdzie w oznaczeniu $\tilde{s}_j^1$ górny indeks oznacza numer chwili z horyzontu predykcji, dla której wyznaczono te wartości (w tym przypadku 1). Wartość pierwszego elementu trajektorii swobodnej można więc obliczyć korzystając ze wzoru:

$$y_{k+1|k}^0 = \sum_{i=2}^{D-1} \tilde{s}_i^1 \cdot \Delta u_{k-i+1} + \tilde{s}_D^1 \cdot u_{k-D+1} + d_k. \tag{4.114}$$

Uwaga: Zauważmy, że w powyższym wzorze celowo pominięto składnik zależny od bieżącego przyrostu sterowania. Uczyniliśmy tak, ponieważ wyznaczamy odpowiedź swobodną.

Sytuacja ulegnie komplikacji w kolejnej chwili z horyzontu predykcji (k+2), ponieważ w pierwszym poprzedniku prostym powinna pojawić się wartość wyjścia z chwili k+1. Wartością tą jednak nie dysponujemy. Dlatego posłużymy się już wyznaczoną wartością pierwszego elementu trajektorii swobodnej ze wzoru (4.114). W takim razie otrzymamy:

Reguła i: jeśli $y_{k+1|k}^0$ jest B_1^i i y_k jest B_2^i i … i y_{k-n+2} jest B_n^i i

u_k jest C_1^i i u_k jest C_2^i i u_{k-1} jest C_3^i i … i u_{k-m+2} jest C_m^i to

$$y_{k+2}^{i} = \sum_{j=3}^{D-1} s_j^i \cdot \Delta u_{k-j+2} + s_D^i \cdot u_{k-D+2} \tag{4.115}$$

oraz wartość wyjścia modelu

$$y_{k+2}^{M} = \sum_{i=3}^{D-1} \tilde{s}_i^{\,2} \cdot \Delta u_{k-i+2} + \tilde{s}_D^{\,2} \cdot u_{k-D+2} \,. \tag{4.116}$$

W takim razie drugi element odpowiedzi swobodnej będzie opisany zależnością:

$$y_{k+2|k}^{0} = \sum_{i=3}^{D-1} \tilde{s}_i^{\,2} \cdot \Delta u_{k-i+2} + \tilde{s}_D^{\,2} \cdot u_{k-D+2} + d_k \,. \tag{4.117}$$

Tę iteracyjną procedurę należy przeprowadzić dla wszystkich N elementów odpowiedzi swobodnej. Następnie, otrzymaną odpowiedź swobodną oraz macierz dynamiczną otrzymaną w taki sam sposób, jak w algorytmie FDMC–SL należy użyć do sformułowania zadania optymalizacji. Zauważmy, że będzie to łatwe do rozwiązania zadanie optymalizacji kwadratowej.

Uwaga: Jest możliwe uproszczenie algorytmów FDMC–SL i FDMC–NPL polegające na formułowaniu w każdej iteracji algorytmu zadania optymalizacji bez ograniczeń, jak w algorytmie DMC w wersji analitycznej. Zauważmy jednak, że niestety i tak w każdej iteracji algorytmu należy powtórzyć wyznaczenie macierzy dynamicznej i odpowiedzi swobodnej gdyż ulegną one zmianie ze względu na nieliniowość obiektu regulacji.

Algorytm FGPC–NPL

Prześledźmy teraz proces otrzymywania elementów odpowiedzi swobodnej w rozmytym algorytmie typu GPC. Rozpatrzmy więc rozmyty model Takagi–Sugeno taki, jak w algorytmie FGPC–SL, zapiszmy go jednak w postaci umożliwiającej otrzymanie bieżącej wartości wyjścia (dla chwili k):

Reguła i: jeśli y_{k-1} jest B_1^i i … i y_{k-n} jest B_n^i i u_{k-1} jest C_1^i i … i u_{k-m} jest C_m^i to

$$y_k^i = b_1^i \cdot y_{k-1} + \ldots + b_n^i \cdot y_{k-n} + c_1^i \cdot u_{k-1} + \ldots + c_m^i \cdot u_{k-m} \,.$$

Wyjście rozpatrywanego modelu rozmytego w chwili k jest więc opisane następującym wzorem:

$$y_k^M = \tilde{b}_1 \cdot y_{k-1} + \ldots + \tilde{b}_n \cdot y_{k-n} + \tilde{c}_1 \cdot u_{k-1} + \ldots + \tilde{c}_m \cdot u_{k-m} \,. \tag{4.118}$$

W takim razie model zakłóceń typu DMC będzie tym razem dany wzorem:

$$d_k = y_k - \tilde{b}_1 \cdot y_{k-1} - \ldots - \tilde{b}_n \cdot y_{k-n} - \tilde{c}_1 \cdot u_{k-1} - \ldots - \tilde{c}_m \cdot u_{k-m} \,. \tag{4.119}$$

Dysponując wartością d_k i korzystając ze wzoru (4.111) można wyznaczyć poszczególne elementy trajektorii swobodnej. Czyni się to analogicznie, jak w algorytmie FDMC–NPL, tzn. iteracyjnie korzystając z rozmytego modelu procesu i zakładając, że przyszłe przyrosty sygnału sterującego są równe $\Delta u_{k+i|k}=0$ ($i = 1,\ldots, N_u$). Zauważmy, że tym razem jednak wartości wyjścia procesu występują nie tylko w poprzednikach ale i w następnikach rozmytego modelu obiektu. Proces wyznaczania elementów trajektorii swobodnej będzie więc nieco bardziej złożony.

W celu wyznaczenia elementu trajektorii swobodnej dla chwili k+1, korzystamy z modelu o następującej postaci:

Reguła i: jeśli y_k jest B_1^i i … i y_{k-n+1} jest B_n^i i u_k jest C_1^i i … i u_{k-m+1} jest C_m^i to

$$y_{k+1}^i = b_1^i \cdot y_k + \ldots + b_n^i \cdot y_{k-n+1} + c_1^i \cdot u_k + \ldots + c_m^i \cdot u_{k-m+1}. \tag{4.120}$$

W takim razie

$$y_{k+1}^M = \tilde{b}_1^1 \cdot y_k + \ldots + \tilde{b}_n^1 \cdot y_{k-n+1} + \tilde{c}_1^1 \cdot u_k + \ldots + \tilde{c}_m^1 \cdot u_{k-m+1} \tag{4.121}$$

a wartość pierwszego elementu trajektorii swobodnej można obliczyć ze wzoru:

$$y_{k+1|k}^0 = \tilde{b}_1^1 \cdot y_k + \ldots + \tilde{b}_n^1 \cdot y_{k-n+1} + \tilde{c}_1^1 \cdot u_k + \ldots + \tilde{c}_m^1 \cdot u_{k-m+1} + d_k. \tag{4.122}$$

Aby wyznaczyć wartość elementu trajektorii swobodnej dla kolejnej chwili z horyzontu predykcji (k+2), należy tak, jak w algorytmie FDMC–NPL skorzystać z już wyznaczonej wartości pierwszego elementu trajektorii swobodnej. Zauważmy, że tym razem wystąpi on jednak nie tylko w poprzednikach reguł modelu rozmytego, ale także w następnikach. Otrzymamy więc:

Reguła i: jeśli $y_{k+1|k}^0$ jest B_1^i i y_k jest B_2^i i … i y_{k-n+2} jest B_n^i i
u_k jest C_1^i i u_k jest C_2^i i u_{k-1} jest C_3^i i … i u_{k-m+2} jest C_m^i to

$$\begin{aligned} y_{k+2}^i = {} & b_1^i \cdot y_{k+1|k}^0 + b_2^i \cdot y_k + \ldots + b_n^i \cdot y_{k-n+2} + \\ & + c_1^i \cdot u_k + c_2^i \cdot u_k + c_3^i \cdot u_{k-1} + \ldots + c_m^i \cdot u_{k-m+2}. \end{aligned} \tag{4.123}$$

W takim razie wartość wyjścia modelu będzie opisana wzorem:

$$\begin{aligned} y_{k+2}^M = {} & \tilde{b}_1^2 \cdot y_{k+1|k}^0 + \tilde{b}_2^2 \cdot y_k + \ldots + \tilde{b}_n^2 \cdot y_{k-n+2} + \\ & + \tilde{c}_1^2 \cdot u_k + \tilde{c}_2^2 \cdot u_k + \tilde{c}_3^2 \cdot u_{k-1} + \ldots + \tilde{c}_m^2 \cdot u_{k-m+2}. \end{aligned} \tag{4.124}$$

Drugi element odpowiedzi swobodnej będzie więc opisany zależnością:

$$\begin{aligned} y_{k+2|k}^0 = {} & \tilde{b}_1^2 \cdot y_{k+1|k}^0 + \tilde{b}_2^2 \cdot y_k + \ldots + \tilde{b}_n^2 \cdot y_{k-n+2} + \\ & + \tilde{c}_1^2 \cdot u_k + \tilde{c}_2^2 \cdot u_k + \tilde{c}_3^2 \cdot u_{k-1} + \ldots + \tilde{c}_m^2 \cdot u_{k-m+2} + d_k. \end{aligned} \tag{4.125}$$

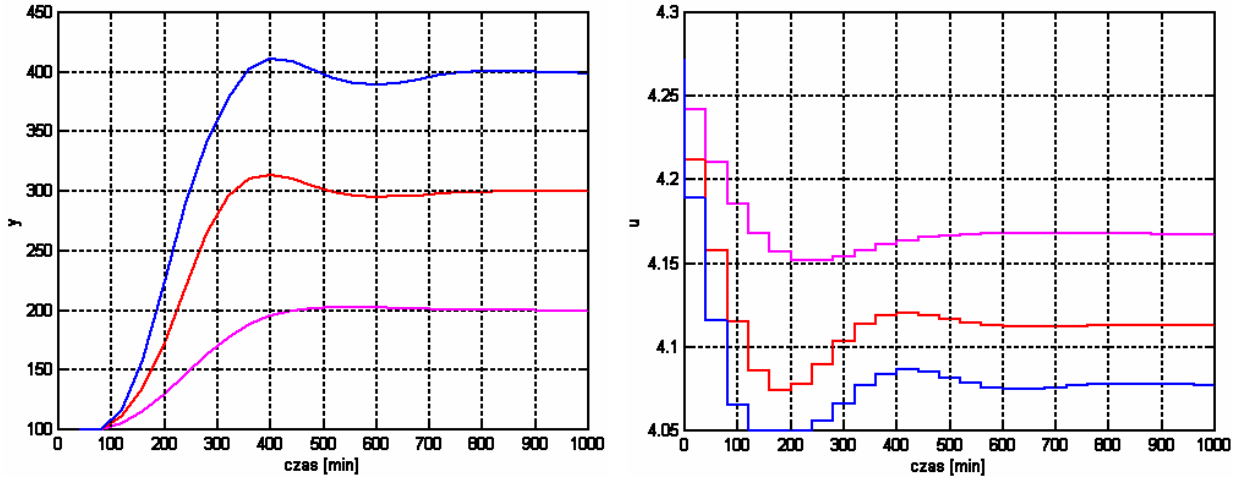
Kolejne elementy odpowiedzi swobodnej należy wyznaczać iteracyjnie w sposób analogiczny do opisanego.

Przykład 4.19

Sprawdźmy, co przyniesie zastosowanie algorytmu FDMC–NPL w układzie regulacji kolumny etylenowej z przykładu 4.18. Przykładowe odpowiedzi zostały pokazane na rys. 4.44. Po zastosowaniu algorytmu typu NPL, otrzymano znacznie szybsze odpowiedzi układu regulacji w porównaniu z odpowiedziami otrzymanymi w układzie regulacji z regulatorem FDMC–SL. Pojawiło się także niewielkie przeregulowanie.

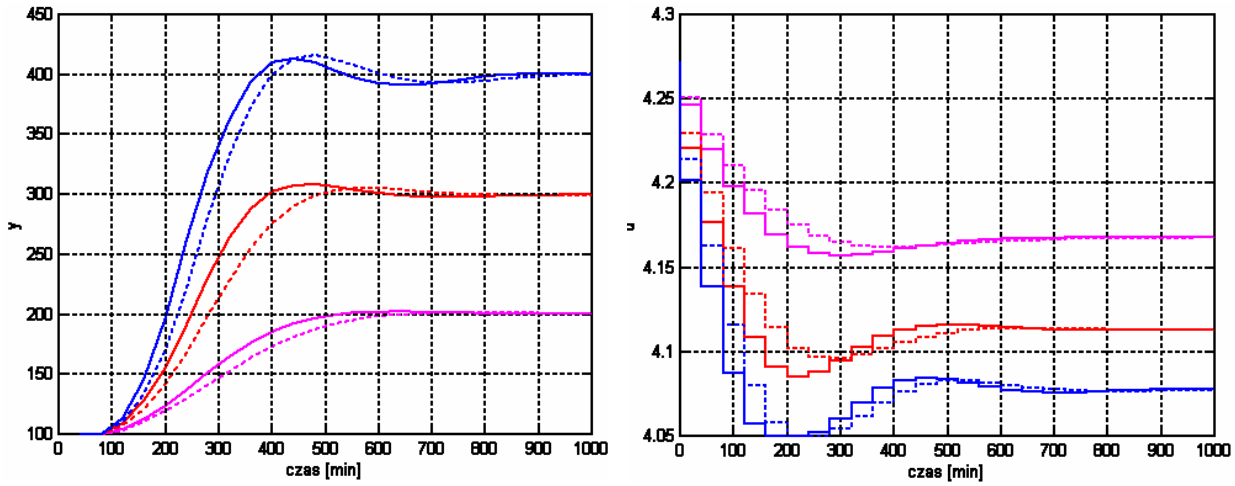
Sprawdźmy także, co się stanie, jeśli zwiększy się wartość parametru dostrajalnego λ. Otrzymane przebiegi zostały pokazane na rys. 4.45. Wraz ze wzrostem wartości parametru λ, zmniejsza się zmienność sygnału sterującego oraz otrzymane odpowiedzi stają się coraz wolniejsze. Odwrotny skutek przynosi zmniejszanie wartości parametru λ.

a) b)



Rys. 4.44. Odpowiedzi układu regulacji z regulatorem FDMC–NPL na skoki wartości zadanej z y_0=100 ppm do y_{zad}= 200, 300, 400 ppm; $\lambda=9\times10^6$; a) przebiegi wyjścia y, b) przebiegi sterowania u

a) b)



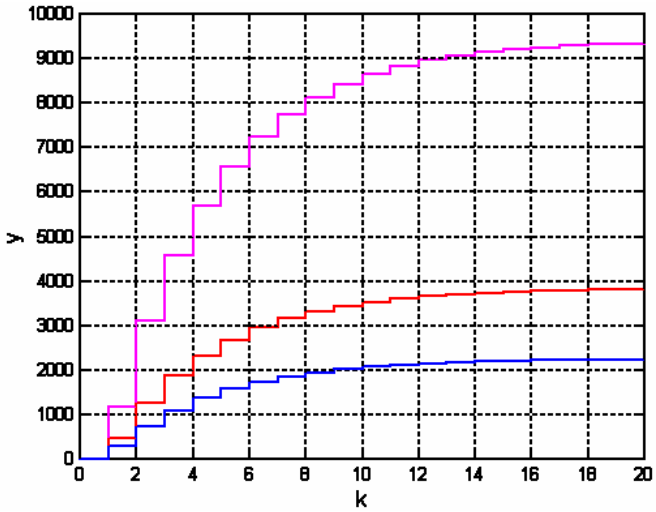
Rys. 4.45. Odpowiedzi układu regulacji z regulatorem FDMC–NPL na skoki wartości zadanej z y_0=100 ppm do y_{zad}= 200, 300, 400 ppm; $\lambda=13\times10^6$ – linie ciągłe, $\lambda=20\times10^6$ – linie przerywane; a) przebiegi wyjścia y, b) przebiegi sterowania u

Uwzględnianie zakłócenia mierzalnego

Algorytmy FMPC–SL i FMPC–NPL można rozszerzyć o mechanizm uwzględniania wpływu zakłócenia mierzalnego na obiekt regulacji. Czyni się to analogicznie, jak w konwencjonalnych algorytmach regulacji predykcyjnej. Najpierw należy model obiektu regulacji rozszerzyć o elementy opisujące wpływ zakłócenia na obiekt. W przypadku modeli rozmytych typu Takagi–Sugeno można to przeprowadzić w stosunkowo prosty sposób dzięki rozbudowaniu następników reguł w modelu rozmytym. Następnie, w przypadku algorytmów z sukcesywną linearyzacją (FMPC–SL), po otrzymaniu w bieżącej iteracji algorytmu liniowej aproksymacji modelu obiektu, postępuje się tak, jak podczas formułowania algorytmów konwencjonalnych, opisanego w rozdz. 2. W algorytmach FMPC–NPL natomiast, model rozszerzony o opis wpływu zakłócenia na proces, jest używany w iteracyjnej procedurze generacji odpowiedzi swobodnej.

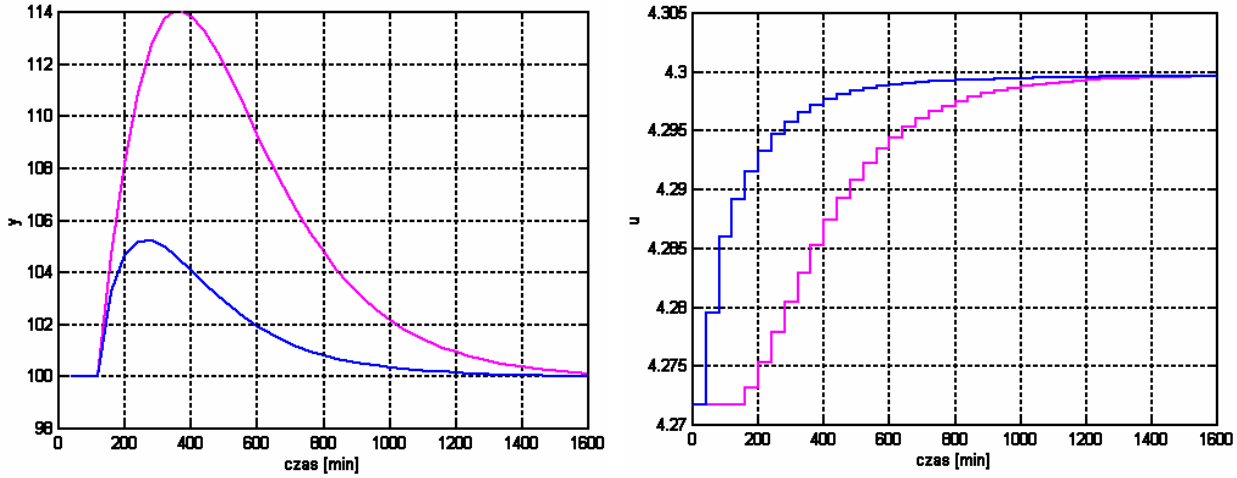
Przykład 4.20

Powróćmy do układu regulacji kolumny etylenowej z algorytmem FDMC–SL z przykładu 4.18. W celu rozszerzenia algorytmu o mechanizm uwzględniania pomiaru zakłócenia, pozyskano odpowiedzi obiektu na skok zakłócenia w okolicach trzech punktów pracy, dla każdej reguły z modelu rozmytego obiektu (rys. 4.46).



Rys. 4.46. Znormalizowane odpowiedzi na skok zakłócenia dla poszczególnych modeli lokalnych w rozmytym modelu Takagi–Sugeno kolumny etylenowej

a) b)



Rys. 4.47. Odpowiedzi układu regulacji z regulatorem FDMC–SL na skok zakłócenia z $x_f = 0{,}81$ do $x_f = 0{,}82$; $\lambda = 9 \times 10^6$; y_{zad}=100 ppm; algorytm z i bez mechanizmu uwzględniania pomiaru zakłócenia; a) przebiegi wyjścia y, b) przebiegi sterowania u

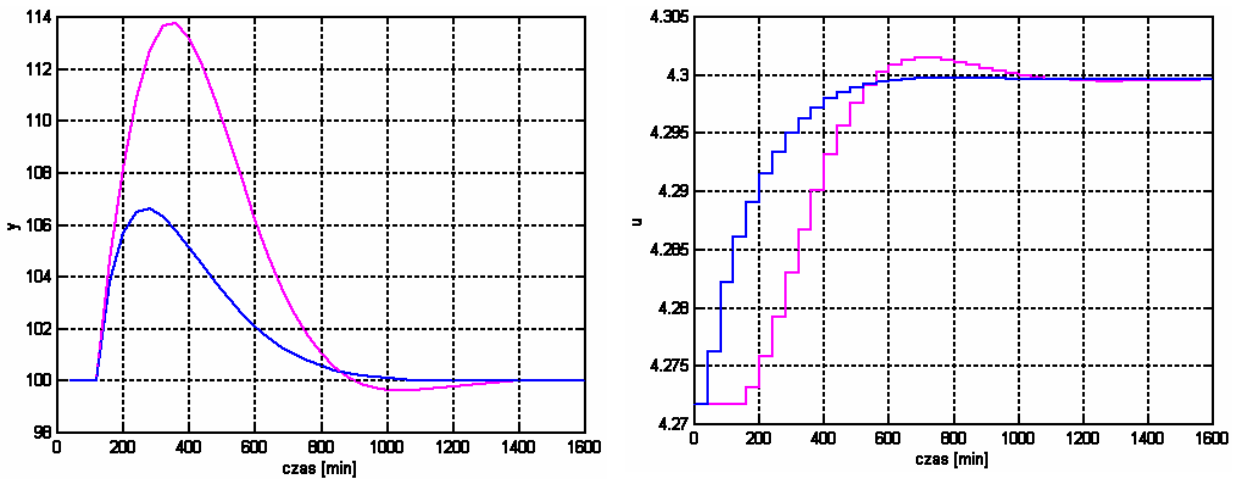
Podczas eksperymentów założono, że wartość zakłócenia zmienia się na samym początku z $x_f = 0{,}81$ do $x_f = 0{,}82$. Na rys. 4.47 pokazano odpowiedzi otrzymane w układzie regulacji z algorytmem FDMC–SL. W przypadku, gdy pomiar zakłócenia nie był uwzględniany (różowe przebiegi na rys. 4.47), ze względu na opóźnienie istniejące w obiekcie, reakcja na zmianę zakłócenia występuje dopiero w 160 minucie eksperymentu. Maksymalny uchyb regulacji jest stosunkowo duży (wynosi ok. 14 ppm). Po zastosowaniu mechanizmu uwzględniania pomiaru

zakłócenia otrzymano znaczną poprawę działania układu regulacji (niebieskie przebiegi na rys. 4.47). Maksymalny uchyb regulacji uległ znacznemu zmniejszeniu. Regulator znacznie szybciej zareagował na zmianę zakłócenia.

Przykład 4.21

Taki sam eksperyment jak w poprzednim przykładzie, przeprowadzono w układzie regulacji z algorytmem FDMC–NPL (rys. 4.48). W wyniku zastosowania mechanizmu uwzględniania pomiaru zakłócenia również otrzymano znaczną poprawę jakości regulacji. Warto jednak także zauważyć różnice wynikające z zastosowania różnych algorytmów. W przypadku algorytmu FDMC–SL bez mechanizmu uwzględniania pomiaru zakłócenia, maksymalny uchyb regulacji jest nieco większy niż przypadku układu z algorytmem FDMC–NPL. Z kolei po zastosowaniu mechanizmu uwzględniania pomiaru zakłócenia w algorytmie FDMC–NPL, maksymalny uchyb jest nieco większy niż w układzie z algorytmem FDMC–SL. Zauważmy jednak, że w każdym przypadku (z i bez uwzględniania pomiaru zakłócenia), algorytm FDMC–NPL zdecydowanie szybciej niż algorytm FMDC–SL kompensuje wpływ zakłócenia.

a) b)



Rys. 4.48. Odpowiedzi układu regulacji z regulatorem FDMC–NPL na skok zakłócenia z $x_f = 0{,}81$ do $x_f = 0{,}82$; $\lambda=9\times10^6$; y_{zad}=100 ppm; algorytm z i bez mechanizmu uwzględniania pomiaru zakłócenia; a) przebiegi wyjścia y, b) przebiegi sterowania u