Politechnika Śląska
Wydział Mechaniczny Technologiczny
Katedra Podstaw Konstrukcji Maszyn

Piotr PRZYSTAŁKA

Metodyka modelowania neuronowego w diagnostyce procesów z uwzględnieniem elementów teorii chaosu

Gliwice 2009

Recenzenci
Prof. dr hab. inż. Józef Korbicz, Uniwersytet Zielonogórski
Prof. dr hab. inż. Wojciech Cholewa, Politechnika Śląska

Redaktor zeszytów
Wojciech Cholewa

Redaktor techniczny
Marek Wyleżoł

Projekt okładki
Wojciech Cholewa, Marek Wyleżoł

BibTeX

```
@BOOK{, title = {Metodyka modelowania neuronowego w diagnostyce
procesów z uwzględnieniem elementów teorii chaosu}, publisher =
{Politechnika Śląska, Katedra Podstaw Konstrukcji Maszyn}, year
= {2009}, author = {Przystałka, P.}, volume = {141}, series =
{Zeszyty}, address = {Gliwice}}
```

Znaczące fragmenty pracy zrealizowano w ramach projektu badawczego promotorskiego N N514 3412 33 finansowanego przez MNiSW.

ISBN 978-83-60759-13-4

Wydawca
Katedra Podstaw Konstrukcji Maszyn
Wydział Mechaniczny Technologiczny
Politechnika Śląska
ul. Konarskiego 18a, 44-100 Gliwice
tel. (32) 237-14-67, fax (32) 237-13-60
http://kpkm.polsl.pl

Druk i oprawa (z dostarczonych materiałów)
D&D Sp. z o.o. ul. Moniuszki 6, 44-100 Gliwice

Od autora

Zeszyt został opracowany na podstawie rozprawy doktorskiej, którą wykonałem pod kierunkiem prof. dra hab. Wojciecha Moczulskiego. Publiczna obrona rozprawy odbyła się 22. września 2009 roku przed Komisją powołaną przez Radę Wydziału Mechanicznego Technologicznego. Znaczące fragmenty pracy zrealizowano w ramach projektu badawczego promotorskiego N N514 3412 33 finansowanego przez MNiSW. Kierownikiem projektu jest promotor rozprawy.

Składam serdeczne podziękowania recenzentom rozprawy doktorskiej, prof. dr. hab. inż. Józefowi Korbiczowi oraz prof. dr. hab. inż. Wojciechowi Cholewie za cenne uwagi, które starałem się uwzględnić w niniejszym zeszycie.

Pragnę wyrazić swoją wdzięczność mojemu promotorowi prof. Wojciechowi Moczulskiemu nie tylko za cenne wskazówki, inspirację i wsparcie podczas przygotowywania rozprawy doktorskiej, ale również za intensywną i wszechstronną opiekę naukową podczas studiów magisterskich i doktoranckich. Składam serdeczne podziękowania Koleżankom i Kolegom z Katedry Podstaw Konstrukcji Maszyn Politechniki Śląskiej za okazaną mi pomoc i życzliwość w czasie wykonywania pracy.

Niniejszą książkę dedykuję mojej najbliższej rodzinie a w szczególności mojej żonie Monice, dziękując za ich wsparcie, wyrozumiałość i cierpliwość.

Gliwice, wrzesień 2009 *Piotr Przystałka*

Spis treści

Od autora **3**

Wykaz ważniejszych oznaczeń **9**

Lista skrótów **11**

Rozdział 1. Wstęp **13**
1.1. Cel rozprawy . . . 15
1.2. Teza rozprawy . . . 15
1.3. Zakres rozprawy . . . 16

Rozdział 2. Wybrane koncepcje diagnostyki procesów technicznych **19**
2.1. Pojęcia elementarne . . . 19
2.1.1. Obiekt diagnozowania . . . 20
2.1.2. Diagnozowanie bezpośrednie . . . 21
2.1.3. Diagnozowanie z zastosowaniem modelu procesu . . . 21
2.1.4. Metody detekcji i lokalizacji uszkodzeń . . . 23
2.1.5. Metody identyfikacji uszkodzeń . . . 24
2.2. Modelowanie w diagnostyce procesów . . . 24
2.3. Modele procesów technicznych . . . 25
2.3.1. Modele teoretyczne . . . 26
2.3.2. Klasyczne modele parametryczne . . . 27
2.3.3. Modele neuronowe . . . 27
2.3.4. Modele rozmyte . . . 28
2.3.5. Modele hybrydowe . . . 29
2.3.6. Modele heurystyczne . . . 30
2.4. Modele do lokalizacji uszkodzeń i rozpoznawania stanów obiektu . . . 31
2.5. Modele neuronowe w detekcji uszkodzeń . . . 32
2.6. Odporna detekcja uszkodzeń . . . 34
2.7. Teoria chaosu w diagnostyce procesów . . . 35
2.8. Podsumowanie . . . 38

Rozdział 3. Rekurencyjne sieci neuronowe **41**
3.1. Struktury w pełni i częściowo rekurencyjne . . . 42
3.1.1. Sieci Jordana i Elmana . . . 42
3.1.2. Sieci typu NNARX . . . 44
3.1.3. Sieci rekurencyjne z czasem ciągłym . . . 45
3.2. Sieci lokalnie rekurencyjne . . . 46
3.2.1. Modele neuronu wg A. Yazdizadeha i K. Khorasaniego . . . 46
3.2.2. Neuron z czasem ciągłym . . . 48
3.2.3. Neuron z filtrem IIR . . . 49
3.2.4. Neuron G-FGS . . . 50
3.2.5. Neuron chaotyczny . . . 50
3.3. Przykłady sieci neuronów dynamicznych . . . 51
3.3.1. Sieć z liniami opóźniającymi . . . 52
3.3.2. Sieć neuronów z filtrem IIR . . . 52
3.4. Projektowanie sieci rekurencyjnych . . . 53
3.4.1. Uczenie sieci rekurencyjnych . . . 53
3.4.2. Dobór struktury w sieciach rekurencyjnych . . . 54
3.4.3. Stabilność i stabilizacja sieci rekurencyjnych . . . 55
3.5. Podsumowanie . . . 55

Rozdział 4. Metodyka modelowania neuronowego w diagnostyce procesów **57**
4.1. Koncepcja metodyki . . . 57
4.2. Zastosowana sieć neuronów dynamicznych . . . 58
4.2.1. Uogólniony model neuronu dynamicznego . . . 58
4.2.2. Właściwości neuronu . . . 60
4.2.3. Sieć lokalnie rekurencyjna . . . 63
4.3. Hybrydowa metoda uczenia . . . 65
4.4. Selekcja wejść modelu neuronowego . . . 68
4.4.1. Współrzędne z opóźnieniem . . . 68
4.4.2. Test Gamma . . . 68
4.4.3. Metoda wskaźników pojemności informacyjnej . . . 69
4.5. Ocena działania modelu neuronowego procesu . . . 72
4.6. Analiza stabilności modelu neuronowego . . . 74
4.7. Poszukiwanie modelu suboptymalnego . . . 77
4.8. Metoda odpornej detekcji uszkodzeń . . . 78
4.9. Ocena sprawności systemu diagnostycznego . . . 80
4.10. Podsumowanie . . . 81

Rozdział 5. Badania weryfikacyjne **83**
5.1. Plan weryfikacji . . . 83
5.2. Oprogramowanie . . . 83
5.3. Weryfikacja wstępna . . . 84
5.3.1. Wybór modelowanych procesów i struktur wzorcowych . . . 84

5.3.2. Porównanie hybrydowych algorytmów uczących 86
5.3.3. Identyfikacja wielowymiarowego systemu dynamicznego 92
5.3.4. Identyfikacja układu chaotycznego 94
5.3.5. Modelowanie procesu rzeczywistego 98
5.3.6. Modelowanie złożonego procesu przemysłowego 101
5.4. Odporna detekcja uszkodzeń wybranego elementu wykonawczego 104
5.4.1. Obiekt diagnozowania 105
5.4.2. Weryfikacja - Etap I 108
5.4.3. Weryfikacja - Etap II 116
5.5. Podsumowanie badań weryfikacyjnych 120

Rozdział 6. Podsumowanie i wnioski 121
6.1. Wnioski ogólne 122
6.2. Wnioski szczegółowe 122
6.3. Kierunki dalszych badań 123

Rozdział A. Globalne i lokalne algorytmy uczenia 125
A.1. Algorytm ewolucyjny 125
A.1.1. Populacja oraz kodowanie chromosomów 125
A.1.2. Funkcja przystosowania 125
A.1.3. Selekcja i sukcesja 126
A.1.4. Operatory ewolucyjne 126
A.2. Algorytm symulowanego wyżarzania 128
A.2.1. Generowanie nowych rozwiązań 128
A.2.2. Redukcja temperatury 128
A.2.3. Akceptacja nowego rozwiązania 129
A.3. Algorytm przeszukiwania bezpośredniego 129
A.4. Algorytm największego spadku 130
A.5. Algorytm zmiennej metryki 130
A.6. Algorytm Levenberga-Marquardta 131
A.7. Algorytmy z losowaniem kierunku 131
A.8. Gradient i jakobian funkcji celu 132
A.8.1. Aproksymacja numeryczna 133
A.8.2. Aproksymacja stochastyczna 133

Rozdział B. Wybrane zagadnienia modelowania neuronowego 135
B.1. Reprezentacja danych 135
B.2. Wybrane metody wstępnego przetwarzanie danych 136
B.3. Podział zgromadzonych danych 138
B.4. Wstępny dobór struktury sieci 139
B.5. Wartości początkowe swobodnych parametrów sieci 141

Bibliografia **157**

Streszczenie **159**

Summary **160**

Wykaz ważniejszych oznaczeń

$\mathbf{CW}$	macierz wag warstwy kontekstowej
Δt	odstęp próbkowania
δ_s	liczba osobników w sukcesji elitarnej
det	miara determinizmu diagramu rekurencyjnego
$E(\cdot)$, E_c	funkcja celu, liczba wyliczeń funkcji celu $E(\cdot)$
ent	miara entropii rozkładu linii diagonalnych diagramu rekurencyjnego
$\mathbf{f}^i$, f^i	wektor funkcji wyjściowych neuronów i-tej warstwy, funkcja wyjścia i-tego neuronu
γ_1, γ_2	parametry mutacji nierównomiernej
$\mathbf{H}$	hesjan
$\mathcal{I}$	maksymalna liczba iteracji algorytmu optymalizacji lokalnej
$\mathbf{IW}$	macierz wag warstwy wejściowej
$\mathbf{J}$	jakobian
k	czas dyskretny
$\mathcal{L}_T$	zbiór przykładów trenujących
$\mathcal{L}_V$	zbiór przykładów weryfikujących
$\mathcal{L}_G$	zbiór przykładów testowych
$\mathbf{LW}$	macierz wag warstwy ukrytej
λ	parametr funkcji celu określający moc czynnika regularyzacyjnego
λ_h	parametr krzyżowania heurystycznego
lam	wskaźnik laminarności diagramu rekurencyjnego
$\nabla E(\cdot)$	gradient funkcji celu $E(\cdot)$
$\boldsymbol{\omega}$	wektor parametrów swobodnych sieci neuronowej
ω_0	parametr składnika regularyzacyjnego funkcji celu
$\boldsymbol{\omega}^*$	wektor optymalny parametrów sieci
p, p_w	liczba swobodnych parametrów modelu, modelu wzorcowego
p_k	prawdopodobieństwo krzyżowania
$\mathbf{r}(k)$, $r(k)$	wektor residuów, residuum
r_m	prawdopodobieństwo mutacji równomiernej
rr	wskaźnik rekurencji diagramu rekurencyjnego
R	parametr czynnika błędu Minkowskiego funkcji celu $E(\cdot)$
tt	czas pułapkowania

$\mathbf{u}(k)$	wektor wejść obiektu/modelu w chwili k
$\mathbf{y}(k)$	wektor wyjść obiektu/modelu w chwili k
$\mathbf{x}(k)$	wektor stanu obiektu/modelu w chwili k

Lista skrótów

AIC	kryterium informacyjne Akaike
ARX	system/model autoregresyjny z zewnętrznym wejściem
ARMAX	system/model autoregresyjny ze średnią ruchomą i zewnętrznym wejściem
BFGS	algorytm zmiennej metryki z formułą BFGS (Broydena-Fletchera-Goldfarba-Shanno)
BIC	kryterium informacyjne Schwarza
DFP	algorytm zmiennej metryki z formułą DFP (Davidona-Fletchera-Powella)
DS	algorytm przeszukiwania bezpośredniego
EA	algorytm ewolucyjny
FPE	kryterium końcowego błędu predykcji
GB	algorytm największego spadku
HQC	kryterium informacyjne Hannana-Quinna
IIR	filtr o nieskończonej odpowiedzi impulsowej
INIT	metoda wyznaczania wstępnych wartości parametrów modelu neuronowego
JEW	kryterium informacyjne Jenkinsa-Wattsa
MAPE	średni bezwzględny błąd procentowy
mMAPE	zmodyfikowany średni bezwzględny błąd procentowy
MIMO	system/model o wielu wejściach i wielu wyjściach
MISO	system/model o wielu wejściach i jednym wyjściu
MSE	błąd średniokwadratowy
NNARX	neuronowy nieliniowy model autoregresyjny z zewnętrznym wejściem
nRMSE	znormalizowany błąd RMSE
OBD	metoda przycinania sieci (ang. *Optional Brain Damage*)
RMSE	pierwiastek z błędu średniokwadratowego
RPE	względny błąd predykcji
RS	algorytm z losowym wyborem kierunku
RQA	analiza diagramów rekurencyjnych
SA	algorytm symulowanego wyżarzania
SISO	system/model o jednym wejściu i jednym wyjściu
SSE	suma kwadratów reszt
TDL	linie opóźniające (ang. *tapped delay line*)

Rozdział 1

Wstęp

Narastająca złożoność współczesnych środków technicznych, od których wymaga się optymalnej wydajności przy jednoczesnym zachowaniu maksymalnego bezpieczeństwa pracy, czyni problematykę diagnostyki procesów technicznych jednym z najistotniejszych kierunków rozwoju badań nowoczesnej automatyki i robotyki (Caccavale i Villani, 2003; Korbicz *i in.*, 2004; Patton, Frank i Clark, 2000). Ważnym aspektem eksploatacji układów automatycznej regulacji i sterowania ze względu na czynniki zagrożenia ludzkiego życia oraz skażenia środowiska jest zagadnienie bezpieczeństwa. Można podać wiele obszarów współzależności człowieka i środka technicznego, gdzie bezpieczeństwo odgrywa kluczową rolę, takich jak przemysł lotniczy, kosmiczny, motoryzacyjny, energetyczny czy też chemiczny. Innym bardzo ważnym czynnikiem jest wzgląd ekonomiczny powodujący, że przy obecnym sposobie podejścia do problemu możliwe jest prowadzenie złożonego obiektu nawet wówczas, gdy występują uszkodzenia jego podzespołów składowych. Uszkodzenia, o których jest mowa, mogą dotyczyć zarówno komponentów instalacji technologicznej, jak i skomplikowanych urządzeń wykonawczych i pomiarowych (Blanke *i in.*, 2006).

Biorąc pod uwagę różne schematy wnioskowania w diagnostyce technicznej (Cholewa *i in.*, 2008; Korbicz *i in.*, 2004; Moczulski, 2002), można wyróżnić dwie podstawowe klasy modeli, które realizują takie zadania jak odwzorowanie zachowania obiektu oraz wnioskowanie diagnostyczne. Pierwszą grupę stanowią modele obiektów (statyczne lub dynamiczne), drugą - modele odwzorowujące oraz modele diagnostyczne. Diagnozowanie z zastosowaniem modelu obiektu jest podejściem charakterystycznym w diagnostyce procesów przemysłowych, której przedmiotem może być proces technologiczny lub/i proces eksploatacji maszyn i urządzeń. Diagnozowanie w tym podejściu polega na odpowiednim zastosowaniu modelu diagnozowanego obiektu do detekcji, lokalizacji oraz identyfikacji uszkodzeń (Isermann i Ballé, 1997; Isermann, 2006; Patton, Frank i Clark, 2000). Jest więc uzasadnione, że dla potrzeb diagnostyki obiektów technicznych celowe jest dysponowanie odpowiednimi ich modelami (Isermann, 2005).

W przypadku typowych obiektów badań diagnostycznych, jak np. maszyny krytyczne (Kiciński J. (Red.), 2005) lub procesy przemysłowe (Korbicz *i in.*, 2004), realizowane w energetyce, przemyśle chemicznym, przemyśle wytwórczym, hutnictwie i wielu innych, modele analityczne dynamiki, jak również same obiekty są tak skomplikowane

i różnorodne, że często wymagają podejścia heurystycznego (Morrison, 1996). Klasyczne techniki modelowania w niektórych przypadkach nie prowadzą do zadowalających rezultatów. Jest to spowodowane częściowym brakiem informacji o istotnych wejściach, wyjściach oraz stanie obiektu w danej chwili, złożonością obiektów, ogromną ilością danych procesowych itp. Powoduje to, iż coraz częściej sięga się do metod, które L. Zadeh (1994) nazwał miękkimi obliczeniami (ang. *soft computing*). Można zauważyć, że metody sztucznej inteligencji stosowane w układach sterowania i diagnostyki służą raczej jako uzupełnienie i nie stanowią konkurencji dla dobrze znanych metod klasycznych (Isermann, 2005; Korbicz, 2006; Ovaska *i in.*, 2006; Patton, Uppal i Lopez-Toribio, 2000), co szczególnie pokazują zastosowania przemysłowe (Berenyi *i in.*, 2001; Isermann i Ballé, 1997; Kamiya *i in.*, 2005; Schlang *i in.*, 2001).

Jedną z gałęzi diagnostyki procesów wspartej modelem diagnozowanego obiektu, obecnie intensywnie rozwijaną, stanowią metody tworzenia modeli obiektów bazujące na sztucznych sieciach neuronowych. Spowodowane jest to faktem, że sieci neuronowe uważane są za uniwersalne aproksymatory różnego rodzaju zależności funkcyjnych (Duch *i in.*, 2000; Korbicz *i in.*, 1994; Tadeusiewicz, 1993), a w szczególności nieliniowych zależności dynamicznych (Gupta *i in.*, 2003; Patan, 2008a). Modele neuronowe procesów wykorzystuje się zazwyczaj w fazie detekcji uszkodzeń do generowania residuów, na podstawie których wyznaczane są sygnały diagnostyczne (symptomy uszkodzeń). Modele tego typu stosowane są zazwyczaj w sytuacjach, gdy utworzenie modelu obiektu klasycznymi metodami modelowania jest zbyt trudne lub wręcz niemożliwe do wykonania.

Systemy diagnostyczne czasu rzeczywistego z reguły projektuje się w ten sposób, aby rozdzielić trzy podstawowe zadania przytoczone powyżej. Detekcja uszkodzenia (czyli wskazanie występowania uszkodzenia w obiekcie) jest pierwszym stadium procesu diagnozowania. Skuteczne i szybkie wykrycie uszkodzenia ma bezpośredni wpływ na końcową diagnozę. Wszystkie znane metody detekcji uszkodzeń rozwijane na gruncie diagnostyki procesów wspartej modelem obiektu wrażliwe są na niemierzalne zakłócenia, szumy pomiarowe oraz niepewność modelu (niezależnie od jego typu). Jest to głównym powodem rozwoju odpornych metod diagnozowania uwzględniających poza zakłóceniami i szumami również niepewność modelu (Chen i Patton, 1998; Korbicz, 2006). Odporność systemu detekcji rozumie się jako niewrażliwość tego systemu na różnego rodzaju błędy i niepewności (Chen i Patton, 1998).

Źródłem niepewności modelu neuronowego są błędy strukturalne i parametryczne, które spowodowane są głównie przez dwa czynniki. Pierwszy z nich wywołany jest potrzebą doboru odpowiedniej klasy sieci oraz jej struktury do danego zadania. Natomiast drugi czynnik wynika z niedoskonałości metod estymacji nieznanych parametrów swobodnych sieci oraz z jakości danych użytych do estymacji. Autorzy opracowań krajowych i zagranicznych wskazują na potrzebę rozwoju zarówno elementarnych modeli neuronu, jak również struktur neuronowych (głównie rekurencyjnych) umożliwiających wierniejsze odwzorowanie procesów dynamicznych (Ayoubi, 1994; Korbicz *i in.*, 2004; Patan *i in.*, 2008; Wieczorek, 2008). Prowadzi to do równoległego rozwoju metod uczenia sieci neuronowych, które bazują na algorytmach optymalizacji globalnej i lokalnej funkcji

wielu zmiennych, metod optymalizacji struktury sztucznej sieci neuronowej oraz metod analizy wrażliwości danych uczących. Takie działania mogą przyczynić się do stopniowego ograniczenia błędów strukturalnych i parametrycznych. Ze względu na fakt, że całkowite wyeliminowanie niepewności modelu neuronowego nie jest możliwe, wielu autorów podejmuje próby tworzenia pasywnych metod detekcji uszkodzeń odpornych na tego typu niepewności. W podejściu tym sposób postępowania zazwyczaj polega na uwzględnianiu niepewności modelu na etapie wyznaczania adaptacyjnych progów decyzyjnych (Korbicz, 2006; Mrugalski *i in.*, 2008; Patan *i in.*, 2008).

W rozprawie zaproponowano taką metodykę tworzenia modeli neuronowych procesów, której zastosowanie podczas projektowania systemu detekcji uszkodzeń wybranego obiektu rozważań może przyczynić się do poprawy sprawności działania tego systemu. Metodyka wykorzystuje elementy teorii chaosu deterministycznego, które w głównej mierze stosowane są podczas tworzenia modelu neuronowego procesu oraz podczas opracowywania systemu detekcji uszkodzeń. Ważnym elementem metodyki jest zaproponowana przez autora metoda odpornej detekcji uszkodzeń, oparta na opracowanych modelach neuronowych oraz analizie ilościowej diagramów rekurencyjnych. Opracowaną metodykę poddano weryfikacji w dwóch stadiach. Weryfikacja wstępna prowadzona była na danych pozyskanych z symulacji różnej klasy procesów dynamicznych oraz danych zgromadzonych na obiektach rzeczywistych. Weryfikacja docelowa prowadzona była dla danych udostępnionych w ramach europejskiego projektu DAMADICS (ang. *Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems*). Umożliwiają one względne porównanie różnego rodzaju metod diagnozowania (Bartyś *i in.*, 2006).

Badania autora związane z zastosowaniem rekurencyjnych sieci neuronowych w diagnostyce procesów zapoczątkowane zostały pod kierunkiem W. Moczulskiego podczas realizacji wybranych zadań w ramach projektu badawczego KBN 4 T07B 018 27 pt. *"Metodyka heurystycznego modelowania obiektów i procesów dynamicznych w diagnostyce i sterowaniu"*.

1.1. Cel rozprawy

Celem niniejszej rozprawy doktorskiej było opracowanie skutecznej metodyki modelowania neuronowego w diagnostyce procesów z uwzględnieniem elementów teorii chaosu deterministycznego. Opracowana metodyka umożliwia budowę odpornych systemów detekcji uszkodzeń dla możliwie szerokiej klasy procesów, będących obiektem rozważań diagnostyki technicznej.

1.2. Teza rozprawy

Mając na uwadze przedstawiony we wstępie problem badawczy oraz zdefiniowany cel rozprawy, sformułowano następującą tezę rozprawy:

Opracowana metodyka modelowania neuronowego procesów uwzględniająca wybrane elementy teorii chaosu deterministycznego umożliwia tworzenie skutecznie działających systemów detekcji uszkodzeń, odpornych na zakłócenia i błędy modelowania.

1.3. Zakres rozprawy

Rozprawa dotyczy opisu formalnego oraz weryfikacji metodyki modelowania neuronowego w diagnostyce procesów technicznych z zastosowaniem inżynierii chaosu.

Rozdział pierwszy obejmuje wstęp objaśniający przyczynę opracowania metodyki tworzenia modeli neuronowych procesów w oparciu o nowy model neuronu oraz wyjaśnia powód zastosowania nowego sposobu oceny residuów uzyskiwanych za pomocą opracowanych modeli neuronowych. Rozdział ten zawiera cel pracy oraz tezę rozprawy.

Rozdział drugi poświęcono prezentacji obszaru badawczego, jakim jest szeroko rozumiana diagnostyka procesów technicznych. Scharakteryzowano wybrane modele do detekcji i lokalizacji uszkodzeń. Szczególną uwagę poświęcono metodom detekcji uszkodzeń opartym na modelach neuronowych procesu, wskazując jednocześnie główne problemy związane z praktycznym zastosowaniem tego podejścia. Dokonano również przeglądu bieżącego stanu literatury dotyczącej zastosowań teorii chaosu w diagnostyce procesów.

Rozdział trzeci dotyczy rekurencyjnych sieci neuronowych. Omówiono w nim zagadnienia, które autor wykorzystał w celu porównania zaproponowanej przez niego struktury neuropodobnej z rozwiązaniami już istniejącymi. Dokonano krytycznego przeglądu oraz systematyki struktur globalnie i lokalnie rekurencyjnych. Wybrano i omówiono najciekawsze zdaniem autora modele neuronów dynamicznych.

Rozdział czwarty poświęcono realizacji głównego celu rozprawy. Na początku rozdziału omówiono sposób reprezentacji danych, które gromadzono do weryfikacji metodyki. Przedstawiono formalny opis zaproponowanego przez autora uogólnionego modelu neuronu dynamicznego oraz struktury sieci lokalnie rekurencyjnej wyposażonej w tego typu jednostki. Zaprezentowano schematy uczące, będące połączeniem znanych algorytmów optymalizacji globalnej i lokalnej funkcji wielu zmiennych. Przedstawiono sposoby wyboru wejść relewantnych zaprojektowanego modelu neuronowego. Następnie zaproponowano połączenie dwóch znanych metod (metody izolinii kryterialnych oraz metody bazującej na mierze wrażliwości), które stosowano do doboru wstępnego oraz przycinania struktury sieci neuronowej. Pod koniec rozdziału zaproponowano metodę detekcji uszkodzeń, w której część decyzyjna zrealizowana jest za pomocą reguł opartych na wartościach miar obliczanych podczas analizy ilościowej diagramów rekurencyjnych (wyznaczanych dla generowanych residuów). W rozdziale zawarto również formuły matematyczne wybranych miar określających dokładność modeli procesów oraz miar oceniających sprawność systemu detekcji uszkodzeń.

Rozdział piąty dotyczy weryfikacji tezy. Badania weryfikacyjne rozdzielone zostały na dwa stadia. Weryfikacja wstępna dotyczyła uzasadnienia celowości wprowadzenia nowych struktur neuronowych. Zaprezentowane badania prowadzone były dla danych pozyska-

nych z symulacji układów dynamicznych oraz dla danych zgromadzonych na obiektach rzeczywistych. Weryfikację zaproponowanej metodyki budowy modelu neuronowego procesu dla celów jego diagnostyki prowadzono z zastosowaniem problemu testowego udostępnionego w ramach europejskiego projektu DAMADICS.

Pracę kończy podsumowanie oraz wnioski i zarysowane kierunki dalszych badań.

Rozdział 2

Wybrane koncepcje diagnostyki procesów technicznych

Diagnostyka techniczna to interdyscyplinarna dziedzina nauki, która w głównej mierze zajmuje się rozpoznawaniem stanu technicznego obiektu diagnozowania (maszyny, urządzenia, procesu) na podstawie wybranych informacji o tym obiekcie zarówno aktualnych jak i/lub historycznych (Cempel, 1989; Cholewa i Moczulski, 1993; Żółtowski, 1996). Diagnostyka procesów technicznych uważana jest za jedną z gałęzi diagnostyki technicznej (Korbicz *i in.*, 2004). W dalszej części rozdziału przedstawiono podstawowe zagadnienia dotyczące diagnostyki procesów.

2.1. Pojęcia elementarne

W diagnostyce technicznej, podobnie jak ma to miejsce w wielu dziedzinach nauki, przyjmuje się, że *istotą procesu jest zmiana*. Tak więc proces jest uporządkowanym w czasie ciągiem stanów będących statycznym opisem własności i właściwości środka technicznego (nośnika procesu) (Moczulski, 2002). W diagnostyce procesów celem diagnozowania jest rozpoznawanie aktualnego stanu procesu. Działanie to ukierunkowane jest na wykrywanie, rozróżnianie oraz oszacowanie rozmiaru uszkodzeń obiektu w wyniku gromadzenia, przetwarzania, analizy oraz oceny sygnałów diagnostycznych (Kościelny, 2001). W niniejszej pracy przyjmuje się, że obiektami badań diagnostycznych mogą być następujące procesy (Korbicz *i in.*, 2004):

- proces technologiczny,
- proces eksploatacji maszyn i urządzeń.

Stan pierwszego z wymienionych procesów określany jest jako zbiór ocen jego odchyleń od procesu wzorcowego. Z drugiej strony, stan procesu eksploatacji definiowany jest jako suma stanów elementarnych instalacji obiektu wraz z oprzyrządowaniem pomiarowym i urządzeniami wykonawczymi (Korbicz *i in.*, 2004).

R. Isermann i P. Ballé (1997) zaproponowali definicje podstawowych pojęć stosowanych w diagnostyce procesów. Terminologia ta została przyjęta przez wiele zespołów

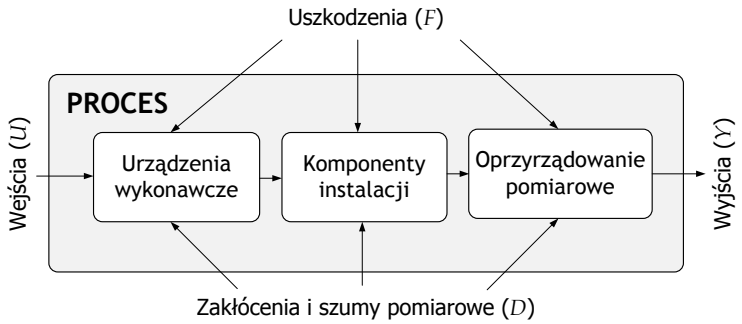
badawczych prowadzących zaawansowane badania w dziedzinie diagnostyki technicznej (Blanke *i in.*, 2006; Caccavale i Villani, 2003; Isermann, 2006; Korbicz *i in.*, 2004; Moczulski, 2002; Patton, Frank i Clark, 2000). Zgodnie z przyjętymi standardami wyróżniamy trzy różne etapy badania stanu:

- detekcja uszkodzenia (ang. *fault detection*),
- lokalizacja uszkodzenia (ang. *fault isolation*),
- identyfikacja uszkodzenia (ang. *fault identification*).

Stosowane jest również pojęcie diagnostyki uszkodzenia (ang. *fault diagnosis*) jako działania łączącego etapy lokalizacji i identyfikacji uszkodzenia. Jak można zauważyć, faza identyfikacji występuje rzadko, a proces diagnozowania obejmuje wyłącznie detekcję i lokalizację uszkodzenia. Należy w tym miejscu nadmienić, że w wielu przypadkach zamiast fazy lokalizacji występuje faza rozpoznawania stanu obiektu lub klasy stanów (Cholewa *i in.*, 2008; Korbicz *i in.*, 2004).

2.1.1. Obiekt diagnozowania

Na Rys. 2.1 przedstawiono ogólny schemat obiektu diagnozowania, który przyjęty został przez wiele zespołów prowadzących badania w obszarze diagnostyki procesów (Blanke *i in.*, 2006; Isermann, 2005; Korbicz *i in.*, 2004). W obiekcie tym wyróżnić możemy szereg wejść, za pomocą których istnieje możliwość oddziaływania otoczenia na elementy składowe obiektu oraz zbiór wyjść umożliwiających oddziaływanie obiektu na otoczenie. Ponadto wyróżniamy dwa typy wejść niejawnych, które reprezentują: (a) uszkodzenia elementów obiektu, (b) zakłócenia niemierzalne i szumy pomiarowe. Podczas diagnozowania procesu zbiór rozpatrywanych uszkodzeń obejmuje uszkodzenia urządzeń wykonawczych, komponentów instalacji oraz oprzyrządowania pomiarowego.

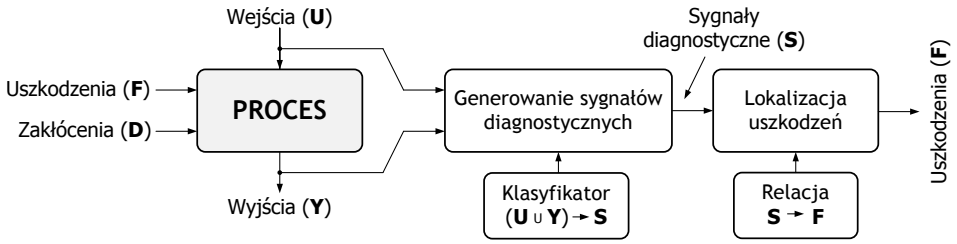


Rys. 2.1: Schemat ideowy obiektu diagnozowania (Kościelny, 2001)

Należy w tym miejscu zauważyć, że w procesie diagnozowania nie uwzględnia się systemu/układu sterowania. Dzieje się tak, ponieważ zazwyczaj detekcja i lokalizacja uszkodzeń elementów sterowników programowalnych, regulatorów, stacji procesowych i operatorskich oraz sieci przemysłowych łączących te urządzenia realizowana jest niezależnie przez te układy (Blanke *i in.*, 2006; Kościelny, 2001).

2.1.2. Diagnozowanie bezpośrednie

Diagnozowanie bezpośrednie to postępowanie, w którym detekcja, lokalizacja i identyfikacja uszkodzeń prowadzona jest na podstawie sygnałów diagnostycznych wygenerowanych wyłącznie w wyniku kontroli i analizy podstawowych i nadmiarowych zmiennych procesowych (Korbicz *i in.*, 2004; Patton, Frank i Clark, 2000). Najczęściej proces ten realizowany jest zgodnie ze schematem przedstawionym na Rys. 2.2. Wyróżniamy tu dwa najważniejsze elementy: blok detekcyjny (wykonujący określone testy diagnostyczne) oraz blok rozróżniający uszkodzenia (realizujący wnioskowanie diagnostyczne). Wejściem bloku detekcyjnego są zmienne procesowe, a wyjściem sygnały diagnostyczne. Zadaniem bloku, w którym następuje lokalizacja uszkodzeń, jest rozróżnienie uszkodzeń na podstawie analizy cech sygnałów diagnostycznych.



Rys. 2.2: Schemat blokowy procesu diagnozowania bezpośredniego z fazą detekcji i lokalizacji uszkodzeń (Korbicz *i in.*, 2004)

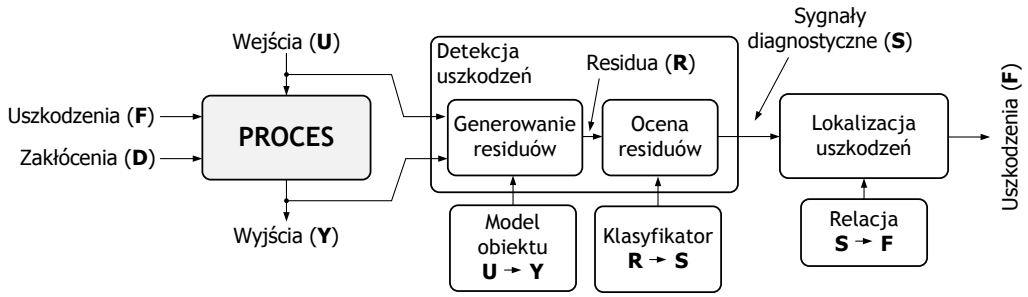
Czasem zamiast fazy lokalizacji występuje faza rozpoznawania stanu obiektu lub klasy stanu. W niektórych sytuacjach stosowane jest podejście łączące fazę detekcji i lokalizacji uszkodzeń. Wymaga to środków umożliwiających odwzorowanie przestrzeni wartości zmiennych procesowych (wstępnie przetworzonych) w przestrzeń wartości sygnatur uszkodzeń lub stanów obiektu (Cholewa *i in.*, 2008; Korbicz *i in.*, 2004). Na powyższym rysunku celowo nie wskazano fazy identyfikacji, co zostanie wyjaśnione w dalszej części tego rozdziału.

2.1.3. Diagnozowanie z zastosowaniem modelu procesu

Idea diagnozowania wspartego modelem procesu zrodziła się na podstawie doświadczeń dotyczących redundancji sprzętowej. Koncepcja diagnozowania (patrz Rys. 2.3) jest tu podobna jak w poprzednim przykładzie. Różnica polega jedynie na wprowadzeniu dodatkowego bloku służącego do generowania tzw. residuów, na podstawie których wyznaczane są sygnały diagnostyczne. Zazwyczaj residua rozumiane są jako sygnały będące różnicą pomiędzy rejestrowanymi wartościami zmiennych procesowych a sygnałami otrzymywanymi z modelu (Blanke *i in.*, 2006; Korbicz *i in.*, 2004). Znane są jednak podejścia, w których residua określane są jako różnice pomiędzy wartościami nominalnymi współczynników fizycznych modelu wyznaczonego dla stanu zdatności a wartościami bieżącymi współczynników modelu wyznaczonego w chwili diagnozowania (Isermann, 2005).

Dokonując syntezy metod diagnozowania bazujących na modelu procesu, wyróżnić można trzy podstawowe sposoby generacji residuów (Isermann, 2005; Korbicz *i in.*, 2004).

Pierwszy sposób polega na zastosowaniu modelu, który utworzono dla danych zgromadzonych podczas nominalnej pracy obiektu (dane takie są zazwyczaj ogólnodostępne). Podczas diagnozowania w sytuacji braku uszkodzeń oczekuje się, że wyjścia modelu oraz obiektu nie będą się znacząco różnić od siebie (wartości residuów oscylują na poziomie zera). Wystąpienie uszkodzenia powinno powodować znaczący wzrost wartości residuów.



Rys. 2.3: Schemat blokowy procesu diagnozowania z fazą detekcji i lokalizacji uszkodzeń z zastosowaniem redundancji analitycznej (Korbicz *i in.*, 2004)

Drugi ze sposobów generacji residuów polega na zastosowaniu banku modeli, które zostały zbudowane zarówno na danych pozyskanych w trakcie działania obiektu w stanie pełnej zdatności, jak również podczas działania obiektu z występującym uszkodzeniem. W tej sytuacji dostaje się dodatkowe residua, których wartości będą oscylować wokół zera, gdy wystąpią odpowiadające im uszkodzenia. Ten sposób generacji residuów znacząco ułatwia drugą fazę diagnozowania. Niemniej jednak pozyskanie danych z różnych stanów działania obiektu przysparza wielu problemów, powodując, że schemat ten stosowany jest głównie przy projektowaniu urządzeń z wbudowanymi modułami samo-diagnozującymi, które produkowane będą wielkoseryjne.

Nieco odmiennym sposobem generacji residuów jest zastosowanie identyfikacji online. Ten sposób wymaga budowy statycznego lub dynamicznego modelu (teoretycznego) reprezentującego obiekt oraz jego własności (tj. rezystancje, masy, sztywności, itp.) w stanie zdatności. Współczynniki określające własności fizyczne obiektu zawarte są w parametrach modelu obiektu. Jeśli na podstawie prowadzonej w czasie rzeczywistym identyfikacji modelu procesu określi się wartości tych współczynników i porówna z wartościami nominalnymi (tzn. dla modelu wyznaczonego dla stanu nominalnego), to uzyskane różnice są różnicami niosącymi informację o uszkodzeniach.

Niezależnie od sposobu generacji residuów niezbędna jest odpowiednia ich ocena. Ta część zadania detekcji uszkodzenia nazywana jest częścią decyzyjną. Stosuje się wiele różnych technik oceny residuów, z czego najważniejsze to: metody z arbitralnie przyjmowaną wartością progową, metody z adaptacyjnie wyznaczaną wartością progową, metody z zastosowaniem oceny rozmytej residuów i inne (Chen i Patton, 1998). Wygenerowane w taki sposób sygnały diagnostyczne przetwarzane są w bloku rozróżniającym uszkodzenia. Możliwe jest również podejście, w którym wnioskowanie diagnostyczne prowadzone jest bezpośrednio na residuach, co tym razem wymaga założenia o dostępności środków umożliwiających odwzorowanie przestrzeni wartości residuów (zazwyczaj wstępnie przetworzonych) w przestrzeń wartości sygnatur uszkodzeń lub stanów obiektu.

2.1.4. Metody detekcji i lokalizacji uszkodzeń

W diagnostyce procesów detekcja uszkodzeń jest na ogół pierwszym etapem diagnozowania realizowanym przez system diagnostyczny. Od algorytmów detekcyjnych wymaga się dużej szybkości działania (detekcja on-line) oraz wysokiej sprawności określanej poprzez takie wskaźniki ilościowe jak odsetek fałszywych alarmów, odsetek prawidłowo wskazanych uszkodzeń, prawidłowo wskazanego momentu detekcji uszkodzenia itp (Bartyś *i in.*, 2006). Najczęściej stosowanymi metodami detekcji w układach diagnostyki opartych na schematach omówionych w poprzednich punktach wg (Korbicz *i in.*, 2004; Kościelny, 2001) są:

- metody kontroli ograniczeń wartości zmiennych procesowych,
- metody analizy sygnałów,
- metody wykorzystujące proste związki między zmiennymi procesowymi,
- metody wykorzystujące modele obiektu.

Wystąpienie uszkodzenia powoduje przejście obiektu ze stanu bez uszkodzenia do stanu z uszkodzeniem. Metody lokalizacji uszkodzeń realizują zadanie wnioskowania diagnostycznego, różniąc się między innymi formami zapisu relacji symptomy - uszkodzenia oraz sposobem pozyskiwania wiedzy o tych relacjach (Isermann, 2006; Korbicz *i in.*, 2004; Kościelny, 2001). Od algorytmów stosowanych do lokalizacji uszkodzeń przede wszystkim wymaga się, aby umożliwiały wysoką rozróżnialność uszkodzeń i rozpoznawanie uszkodzeń wielokrotnych oraz pozwalały w łatwy sposób modyfikować procedurę lokalizacji uszkodzeń przy ewentualnych zmianach struktury lub parametrów obiektu.

Ogólny podział metod lokalizacji uszkodzeń, ze względu na sposób pozyskiwania wiedzy o relacjach diagnostycznych jest następujący (Kościelny, 2001):

- metody bazujące na relacji symptomy - uszkodzenia wynikające ze struktury sprzętowej obiektu,
- metody bazujące na relacji symptomy - uszkodzenia wyprowadzanej ze struktury modeli teoretycznych obiektu,
- metody wymagające określenia relacji symptomy - uszkodzenia w procesie uczenia,
- metody wymagające określenia relacji symptomy - uszkodzenia przez eksperta.

Należy w tym miejscu wyraźnie zaznaczyć, że często zamiast fazy lokalizacji uszkodzeń (szczególnie w diagnostyce symptomowej maszyn i urządzeń) pojawia się faza rozpoznania stanu/klasy stanów. Relacje diagnostyczne (symptomy - stan techniczny) w tym przypadku budowane są np. z wykorzystaniem odwrotnych modeli diagnostycznych (Cholewa i Kiciński, 1997; Cholewa *i in.*, 2008). Ponadto istnieją próby połączenia zalet diagnostyki wspartej modelem obiektu oraz diagnostyki symptomowej (Cholewa i Rogala, 2008), co może prowadzić do poprawy sprawności projektowanych systemów diagnostycznych.

2.1.5. Metody identyfikacji uszkodzeń

Identyfikacja uszkodzenia to działanie mające na celu oszacowanie rozmiaru uszkodzenia oraz czasu jego występowania (Isermann i Ballé, 1997). Etap ten jest kolejnym stadium, które niezbędne jest do postawienia pełnej diagnozy. Jak wspomniano wcześniej, faza identyfikacji łączona jest z fazą lokalizacji, tworząc blok, w którym realizowana jest diagnostyka uszkodzenia. Taki sposób postępowania, jak do tej pory, w praktycznych zastosowaniach jest bardzo trudny do zrealizowania. Z drugiej strony, zastosowania przemysłowe pokazują, że detekcja i lokalizacja uszkodzenia jest podejściem wystarczającym (Blanke *i in.*, 2006; Korbicz *i in.*, 2004). Powoduje to, że często używa się pojęcia diagnostyki uszkodzenia w odniesieniu do działania obejmującego wyłącznie lokalizację uszkodzenia (Isermann, 2006).

Niezależnie od przytoczonych powyżej argumentów coraz ważniejszą rolę odgrywają koncepcje diagnozowania podejmujące problem identyfikacji uszkodzenia (Ding, 2008), gdzie stosowane są zarówno metody opierające się na modelach diagnozowanego obiektu (Lehtoranta i Koivo, 2005; Patan *i in.*, 2008), jak i metody zaliczane do koncepcji znanych z diagnostyki symptomowej (Qipeng *i in.*, 2003).

2.2. Modelowanie w diagnostyce procesów

W diagnostyce procesów, podobnie jak w wielu innych dziedzinach nauki, stosowane są różne sposoby modelowania otaczającej nas rzeczywistości, które najogólniej możemy podzielić na (Isermann, 2006; Łęski, 2008):

a) modelowanie teoretyczne, gdzie:
   - znane są prawa fizyki rządzące zachowaniem obiektu/zjawiska,
   - znane są parametry obiektu/zjawiska,
   - wynikiem modelowania jest tzw. model w postaci białej skrzynki;

b) modelowanie eksperymentalne, gdzie:
   - brak jest wiedzy na temat zachowania obiektu/zjawiska,
   - trzeba przyjąć założenia odnośnie do struktury modelu,
   - można mierzyć sygnały wejścia i wyjścia obiektu/zjawiska,
   - wynikiem modelowania jest tzw. model w postaci czarnej skrzynki;

c) modelowanie mieszane, gdzie:
   - przeważa wiedza teoretyczna o obiekcie/zjawisku:
     - znane są prawa fizyki rządzące zachowaniem obiektu/zjawiska,
     - nieznane są parametry obiektu/zjawiska,
     - można mierzyć sygnały wejścia i wyjścia obiektu/zjawiska,
     - wynikiem modelowania jest tzw. model w postaci jasno-szarej skrzynki;

- przeważa wiedza eksperta o obiekcie/zjawisku:
  - znane są reguły fizyki rządzące zachowaniem obiektu/zjawiska,
  - konieczne jest przyjęcie założenia odnośnie do struktury modelu,
  - nieznane są parametry obiektu/zjawiska,
  - można mierzyć sygnały wejścia i wyjścia obiektu/zjawiska,
  - wynikiem modelowania jest tzw. model w postaci ciemno-szarej skrzynki.

Stosując pierwszy z wymienionych sposobów modelowania zakłada się, że rzeczywistość jest opisywalna za pomocą zasad i praw fizyki. Wynikiem takiego postępowania najczęściej jest model analityczny w postaci np. równań fizycznych, równań stanu, transmitancji operatorowej (Gutenbaum, 2003; Kaczorek, 1999; Osowski, 2006a).

Drugi sposób modelowania zgodny jest z ogólną teorią identyfikacji systemów. Model, którego parametry zazwyczaj nie mają interpretacji fizycznej, tworzony jest z użyciem danych pochodzących z eksperymentu. Obiekt poddaje się doświadczeniom, a następnie dobiera się parametry modelu tak, aby pasował on do danych doświadczalnych (innym sposobem może być wykorzystanie danych zgromadzonych w bazie danych podczas działania obiektu). W podejściu tym stosuje się najczęściej modele parametryczne lub sieci neuronowe (Söderström i Stoica, 1997; Korbicz *i in.*, 1994).

W ostatnim z przytoczonych powyżej przypadków, struktura modelu jest określana na podstawie znajomości praw i zasad fizyki opisujących zachowanie obiektu lub na podstawie wiedzy o obiekcie zgromadzonej przez eksperta. Model taki ma wiele nieznanych, ale zazwyczaj interpretowalnych parametrów, których wartości wyznaczane są na podstawie danych pomiarowych. Eksperci są w stanie opisywać złożone układy, nie znając odpowiednich teorii. Ich wiedza reprezentowana jest w formie zbioru stwierdzeń sformułowanych w języku naturalnym. Najczęściej stosowanymi modelami są tu: modele analityczne, których parametry zostały dostrojone na podstawie danych pomiarowych, systemy wnioskowania oparte na zbiorach reguł (ostrych, przybliżonych, rozmytych), systemy neuronowo-rozmyte, sieci przekonań (Cholewa, 2008; Korbicz *i in.*, 2004; Łęski, 2008; Ovaska *i in.*, 2006).

2.3. Modele procesów technicznych

W dalszej części rozprawy opisano różnego rodzaju modele procesów technicznych stosowane do detekcji uszkodzeń. Podział ten zgodny jest z klasyfikacją modeli zaproponowaną w (Korbicz *i in.*, 2004; Kościelny i Bartyś, 2004; Kościelny, 2001). Modele występujące w tej grupie charakteryzuje to, że w przeważającej liczbie są to modele odwzorowujące zależności dynamiczne. Można dokonać ich podziału ze względu na strukturę, wyróżniając modele o strukturach statycznych oraz modele o strukturach dynamicznych. Aby struktury statyczne mogły odwzorowywać własności dynamiczne obiektu, muszą być stosowane w szeregowo-równoległej procedurze identyfikacji (Korbicz *i in.*, 1994; Narendra i Parthasarathy, 1990). Ograniczenie to nie dotyczy drugiej grupy modeli.

2.3.1. Modele teoretyczne

Pierwszym sposobem zapisu dynamiki obiektu jest postępowanie polegające na teoretycznym wyprowadzeniu równań fizycznych (tj. równania ruchu, bilansowe itp.), które następnie można przekształcić do różnych postaci, stosując znane przekształcenia. Dynamikę obiektu o wielu wejściach i wielu wyjściach, przy odpowiednio poczynionych założeniach i uproszczeniach, można zazwyczaj opisać za pomocą układu nieliniowych równań różniczkowych.

Poza modelami w postaci równań różniczkowych zwyczajnych wyróżnić można równania różniczkowe cząstkowe, równania różnicowe, równania algebraiczne, równania całkowe oraz ich połączenia. Modele w postaci równań fizycznych najpełniej opisują związki między zmiennymi procesowymi. Istnieje wtedy możliwość wykrywania uszkodzeń o niewielkich rozmiarach. Często rzeczywisty obiekt jest jednak zbyt złożony, aby opisać go stosując dobrze znane prawa fizyki (lub prawa te są jeszcze nieodkryte). Jest to również przyczyną uzyskiwania modeli tak złożonych, że nie jest możliwe ich zastosowanie np. w diagnostyce on-line.

Modelowanie teoretyczne stosuje się również do opracowywania modelu obiektu z uwzględnieniem uszkodzeń. Wpływa to na zwiększenie złożoności modelu nawet dla prostych obiektów, przez co zazwyczaj nie jest możliwe uzyskanie rozwiązania układu równań w postaci jawnej (Kościelny, 2001). W tym przypadku układy równań opisujące zachowanie obiektu rozwiązywane są drogami symulacji komputerowej (metodami numerycznymi), a wyniki oraz analizy rozwiązań służą do pozyskania niezbędnych relacji diagnostycznych (Cholewa i Kiciński, 1997).

Równania stanu możemy uzyskać z równań fizycznych przez wprowadzenie nowych zmiennych określanych jako zmienne stanu oraz formułując równanie wyjścia. Generacja residuów z zastosowaniem modeli w postaci równań zmiennych stanu realizowana jest za pomocą tzw. redundancji czasowej (Korbicz *i in.*, 2004; Kościelny, 2001). Pojawia się tu dodatkowe ograniczenie dla modelu zlinearyzowanego, ponieważ staje się on czuły na zmiany punktu pracy obiektu.

Z liniowych równań różniczkowych lub liniowych równań stanu można przejść do zapisu modelu typu wejścia-wyjścia z zastosowaniem transmitancji operatorowych. Zastosowanie modeli procesów w postaci transmitancji operatorowych, podobnie jak to było w przypadku liniowych równań stanu, jest ograniczone z uwagi na trudności związane z uzyskaniem odpowiednich residuów, które powinny być czułe na uszkodzenia, a niewrażliwe na naturalne zakłócenia procesu, szumy pomiarowe, błędy modelowania, zmiany punktu pracy (Korbicz *i in.*, 2004).

Jeżeli znane są równania stanu, to możliwe jest konstruowanie tzw. obserwatorów stanu. Obserwator jest algorytmem stosowanym do oszacowania stanu obiektu dynamicznego (procesu) na podstawie sygnałów wejściowych i wyjściowych (Isermann, 2006). Jeżeli równania zmiennych stanu uwzględniają wpływ zakłóceń na obiekt, to do estymacji stanu używane jest podejście oparte na równaniach filtru Kalmana (Isermann, 2006; Patton, Frank i Clark, 2000).

2.3.2. Klasyczne modele parametryczne

Modele parametryczne są wynikiem rozwoju metod i środków ściśle związanych z modelowaniem eksperymentalnym (Söderström i Stoica, 1997). Identyfikacja takich modeli prowadzona jest w czterech głównych etapach polegających na: zgromadzeniu danych (podczas eksperymentu czynnego lub biernego), zdefiniowaniu struktury modelu, estymacji nieznanych parametrów modelu na podstawie zgromadzonych danych, oceny dokładności modelu w świetle określonych kryteriów. W tej grupie modeli możemy wyróżnić:

- modele szeregów czasowych, jak np: modele naiwne, AR - model autoregresyjny, MA - model w postaci średniej ruchomej, modele wygładzania wykładniczego (Browna, Holta) i inne;
- modele obiektów (których parametry nie mają interpretacji), jak np: MAX - model średniej ruchomej z zewnętrznym wejściem, ARX - model autoregresyjny z zewnętrznym wejściem , ARMAX - model autoregresyjny średniej ruchomej z zewnętrznym wejściem, model Boxa-Jenkinsa, modele Wienera i Hammersteina, NARX - nieliniowy model autoregresyjny z zewnętrznym wejściem;
- modele obiektów (których parametry mają interpretację), jak np. modele procesów w postaci funkcji przejścia i modele przestrzeni stanu.

Ciekawym przykładem modeli parametrycznych są modele Wienera i Hammersteina, które stosuje się do opisu nieliniowych systemów dynamicznych. W tym przypadku zakłada się jedynie, że dynamika obiektu może być przedstawiana za pomocą modelu liniowego, a jego właściwości nieliniowe za pomocą statycznego elementu nieliniowego. Przetwarzanie nieliniowe następuje na wejściu lub wyjściu systemu (odpowiednio dla modelu Hammersteina, Wienera).

Model Hammersteina stosowany jest głównie do identyfikacji właściwości dynamicznych elementów wykonawczych. Zaś model Wienera dobrze identyfikuje nieliniowe właściwości obiektu wynikające z charakterystyk czujników pomiarowych (Janczak, 2003). Rozważania teoretyczne i zastosowania praktyczne modeli Wienera i Hammersteina w zadaniach diagnostyki procesów były przedmiotem rozważań np. w (Janczak, 2003; Korbicz *i in.*, 2004).

2.3.3. Modele neuronowe

Jeżeli utworzenie modelu obiektu tradycyjnymi technikami modelowania jest zbyt trudne lub wręcz niemożliwe, to w celu budowy modelu procesu bardzo często stosuje się sztuczne sieci neuronowe o różnorodnej strukturze. Spowodowane jest to faktem, że sieci neuronowe uważane są za uniwersalne aproksymatory różnego rodzaju zależności funkcyjnych (Osowski, 2006b; Hagan *i in.*, 1995; Haykin, 1999; Rutkowski, 2005; Rutkowska *i in.*, 1997; Tadeusiewicz, 1993). W szczególnym przypadku, gdy struktura sieci zawiera połączenia rekurencyjne, można traktować ją jako uniwersalny aproksymator zależności czaso-przestrzennych (Garzon i Botelho, 1999; Gupta *i in.*, 2003; Liang Jin

i in., 1995; Patan, 2008a). Sposób identyfikacji modelu jest taki sam jak dla modeli parametrycznych.

Głównymi zaletami sieci są: możliwość adaptacji, szybkość przetwarzania danych (również dużych zbiorów danych), możliwość odwzorowania zależności silnie nieliniowych, prostota implementacji. Do najważniejszych wad tej techniki należą: potrzeba zgromadzenia odpowiednio reprezentatywnego zbioru przykładów uczących i trudność interpretacji uzyskanego modelu (Cholewa, 1996; Korbicz *i in.*, 1994). Ze względu na dużą popularność obliczeń neuronowych trudno jest wymienić i scharakteryzować wszystkie możliwe typy sieci, dlatego też, poniżej omówione zostaną wybrane topologie, które szczególnie dobrze nadają się do detekcji uszkodzeń (Korbicz *i in.*, 2004; Patan *i in.*, 2008).

1. **Sieci z liniami opóźniającymi** mają statyczną strukturę, która wymaga odpowiedniego przetworzenia przykładów uczących (szeregowo-równoległy sposób identyfikacji). W tej grupie najważniejsze struktury to: wielowarstwowe sieci perceptronowe (ang. *multi-layer perceptrons*), sieci funkcyjne (ang. *functional networks*), sieci typu GMDH (ang. *Group Method of Data Handling*), sieci realizujące regresję uogólnioną GRNN (ang. *General Regression Neural Networks*).
2. **Sieci globalnie rekurencyjne**, które złożone są ze statycznych jednostek przetwarzających i mają połączenia rekurencyjne jedynie pomiędzy neuronami. W tej grupie wyróżniamy: sieci perceptronowe ze sprzężeniem zwrotnym (ang. *recurrent multi-layer perceptrons*), sieci Jordana i Elmana oraz ich wielowarstwowe modyfikacje, wielokontekstowe sieci Elmana i Jordana (ang. *multi-context Jordan/Elman networks*), sieci z czasem ciągłym, sieci typu NNARX.
3. **Sieci lokalnie rekurencyjne**, których topologia jest podobna do sieci jednokierunkowych, a połączenia rekurencyjne zawarte są w neuronach. Istnieje wiele dynamicznych modeli neuronu, z czego najważniejsze to jednostki takie jak: model neuronu ze sprzężeniem w synapsie, model neuronu z filtrem IIR, model neuronu ze sprzężeniem wyjściowym, dynamiczny model neuronu typu GMDH.

Podczas modelowanie neuronowego pojawia się kilka problemów. Pierwszym z nich jest odpowiedni dobór struktury sieci (określenie liczby neuronów w warstwie ukrytej, określenie typu funkcji aktywacji neuronów, ewentualne przycinanie sieci). Kolejnym problemem jest określenie wartości parametrów charakteryzujących opóźnienie sygnałów wejściowych i wyjściowych modelu neuronowego. Innym bardzo ważnym problemem jest odpowiedni podział zgromadzonych danych.

2.3.4. Modele rozmyte

Modelowanie rozmyte stosowane jest wówczas, gdy wiedza o obiekcie diagnozowania jest nieprecyzyjna lub niepełna i równolegle istnieje potrzeba przetwarzania informacji o takim charakterze. Modele rozmyte (systemy rozmyte) oparte są na teorii zbiorów rozmytych zaproponowanej przez L. Zadeha w 1965 roku. Koncepcja ta jest ciągle rozwijana, co przejawia się szerokim spektrum publikacji związanych z tą tematyką w różnych dziedzinach nauki.

Systemy, o których jest mowa, składają się z bazy wiedzy zapisanej w postaci rozmytych reguł warunkowych oraz z bloku wnioskowania wyposażonego w mechanizm wnioskowania przybliżonego (Duch *i in.*, 2000; Rutkowska *i in.*, 1997). W zależności od zastosowania na wejścia systemów rozmytych podaje się wartości numeryczne lub wartości lingwistyczne. Gdy na wejście podawane są wartości numeryczne, niezbędne jest przeprowadzenie operacji rozmywania. Na wyjściu systemu rozmytego otrzymuje się wartości lingwistyczne. Jeżeli jest to konieczne, wartości numeryczne wyjścia uzyskuje się za pomocą operacji wyostrzania, dla zbioru rozmytego uzyskanego w procesie wnioskowania. Blok wnioskowania rozmytego realizowany jest na wiele sposobów z użyciem różnych operatorów stosowanych w logice rozmytej. Jego zadaniem jest stosowanie wiedzy zawartej w regułach w celu wypracowania konkluzji na podstawie przesłanek zbudowanych na wartościach danych. Opisane działania realizowane są zgodnie z metodami wnioskowania przybliżonego.

Kluczowym zadaniem podczas tworzenia modeli rozmytych jest budowa bazy wiedzy. Rozróżnia się trzy różne strategie pozyskania reguł "*jeżeli-to*": metody bezpośrednie (źródłem reguł jest wiedza ekspercka), metody automatycznego generowania reguł na podstawie numerycznych danych o wejściach i wyjściach modelowanego obiektu, oraz metody mieszane (wiedza ekspercka służy do określenia struktury oraz początkowych wartości parametrów modelu, natomiast dane pomiarowe do strojenia modelu).

Do najczęściej stosowanych systemów rozmytych w układach detekcji uszkodzeń można zaliczyć (Korbicz *i in.*, 2004; Rutkowski, 2007) system Mamdaniego-Assilana oraz system rozmyty Takagi-Sugeno-Kanga (TSK). Znane są również inne realizacje systemów rozmytych, takie jak (Łęski, 2008): system rozmyty z parametrycznymi konkluzjami, system Tsukamoto, system Baldwina i inne. W układach diagnostycznych różnych klas obiektów modele rozmyte umożliwiają, poza wyliczaniem wartości zmiennych procesowych, takie działania jak: rozmytą ocenę wartości residuów, zapis relacji symptomy - uszkodzenia, zapis wiedzy deklaratywnej oraz prowadzenie wnioskowania diagnostycznego (Cholewa, 1983; Cholewa i Pedrycz, 1987; Frank i Koppen-Seliger, 1997; Korbicz *i in.*, 2004; Korbicz, 2006; Kościelny, 2001; Kościelny i Bartyś, 2004; Moczulski i Szulim, 2004).

2.3.5. Modele hybrydowe

Modele mieszane są wynikiem łączenia elementarnych technik modelowania. Bardzo popularnymi przykładami tego typu modeli są: falkowe sieci neuronowe, bayesowskie sieci neuronowe, systemy neuronowo-rozmyte, systemy ewolucyjno-rozmyte, przybliżone sieci neuronowe itp. (Lingras, 1998; Liu *i in.*, 2004; Łęski, 2008; MacKay, 1992; Rutkowski, 2005). Zintegrowanie metod używanych oddzielnie ma wiele uzasadnień. Celem takiego działania jest połączenie zalet różnych technik elementarnych przy jednoczesnej minimalizacji wad.

Poniżej omówiono idee połączenia sieci neuronowych i systemów rozmytych oraz podano przykład sposobu generacji uszkodzeń z zastosowaniem tego podejścia. Systemy

tego typu stanowią połączenie struktur opartych na teorii zbiorów rozmytych ze sztucznymi sieciami neuronowymi (Jang *i in.*, 1997; Łęski, 2008; Osowski, 2006b; Rutkowska *i in.*, 1997; Rutkowski, 2005). Dzięki takiemu zestawieniu otrzymano doskonałe narzędzie umożliwiające włączenie wiedzy eksperckiej do procesu przetwarzania informacji. Wiedza ta może posłużyć do określenia wstępnej struktury systemu oraz do inicjowania początkowych wartości jego parametrów. Następnie w sposób automatyczny generuje się reguły dodatkowe i stroi parametry modelu na podstawie danych pomiarowych. Dąży się jednak do tego, aby systemy neuronowo-rozmyte automatycznie pozyskiwały rozmyte reguły warunkowe jedynie na podstawie danych (Łęski, 2008; Rutkowski, 2007). Bardzo ważną zaletą omawianych modeli jest możliwość ich interpretacji.

Jednym z najbardziej znanych systemów wnioskowania rozmytego opartego na sieciach neuronowych jest system ANFIS (ang. *Adaptive-Network-based Fuzzy Inference System*) zaproponowany przez J. Janga. Sieć ta złożona jest z warstwy wejściowej, warstw ukrytych i warstwy wyjściowej realizujących odpowiednie obliczenia rozmyte. Taki zapis struktury systemu rozmytego umożliwia identyfikację nieznanych parametrów modelu za pomocą algorytmów używanych do uczenia sieci neuronowych. W pracach (Lee i Ching-Cheng, 2000; Kowal, 2005; Mastorocostas i Theocharis, 2002) opisano bardziej zaawansowane systemy neuronowo-rozmyte, które posiadają w swej strukturze połączenia rekurencyjne umożliwiające pełniejsze odtworzenie dynamiki obiektu.

Sposób detekcji uszkodzeń za pomocą modeli neuronowo-rozmytych jest taki sam jak poprzednio. Rozważania dotyczące układów diagnostycznych zbudowanych na bazie systemów neuronowo-rozmytych można znaleźć np. w pracach (Ayoubi i Isermann, 1997; Korbicz, 2006; Korbicz i Kowal, 2007; Kościelny, 2001; Moczulski i Hanzel, 2007).

2.3.6. Modele heurystyczne

Modelowanie heurystyczne to postępowanie wywodzące się z miękkich obliczeń, będących, jak sugerował L. Zadeh (1994), połączeniem logiki rozmytej, obliczeń neuronowych i wnioskowania bayesowskiego. Szczegółowe rozważania dotyczące metodyki modelowania heurystycznego, uwzględniające różne sposoby odkrywania wiedzy, opisano w pracach (Moczulski, 2005a; Moczulski, 2005b). Modelowanie heurystyczne obiektów jest fuzją różnego rodzaju metod obliczeniowych, gdzie proces obliczeń bazuje na pojęciach fragmentarycznej prawdy, przybliżenia oraz niepewności. Metodyka modelowania heurystycznego obiektów ma wiele podobieństw z procesem odkrywania wiedzy w bazach danych. Modele heurystyczne budowane są głównie na podstawie danych zgromadzonych na obiekcie rzeczywistym lub otrzymanych w wyniku eksperymentu numerycznego przeprowadzonego za pomocą odpowiedniego symulatora. Nośnikami globalnych modeli heurystycznych mogą być (Moczulski, 2005b): sieci neuronowe, systemy neuronowo-rozmyte, systemy wnioskowania opartego na przykładach, zależności funkcyjne w postaci równań i inne. Jak wskazuje autor przytoczonych opracowań - automatyczny wybór typu/typów modeli dla rozpatrywanego problemu jest trudnym zadaniem. W tym celu należy uwzględnić wiele czynników, które zawrzeć można w trzech grupach zagadnień obejmujących:

znajomość procesu, charakterystykę danych, cel wyznaczania modelu. Jako jedno z możliwych rozwiązań tego problemu zaproponowano inteligentną procedurę bazującą na sieci przekonań, która umożliwiała wnioskowanie na podstawie informacji niepewnych, nieprecyzyjnych i niekompletnych (Ciupke, 2007).

Dynamiczny rozwój metodyki pociąga za sobą udoskonalenie elementarnych technik modelowania. Badania prowadzone w tym zakresie można znaleźć między innymi w pracach (Moczulski i Szulim, 2004; Przystałka, 2007a; Wachla i Moczulski, 2007). W przytoczonych pracach uzyskano wyniki pozwalające na zastosowanie modeli w układach diagnostyki i sterowania.

2.4. Modele do lokalizacji uszkodzeń i rozpoznawania stanów obiektu

Podstawową właściwością modeli do lokalizacji uszkodzeń i rozpoznawania stanów obiektu jest to, że umożliwiają one odwzorowanie przestrzeni wartości sygnałów diagnostycznych (symptomów) w przestrzeń wartości sygnatur uszkodzeń lub stanów obiektu. Wejściowymi sygnałami diagnostycznymi w procesie lokalizacji uszkodzeń lub rozpoznawania stanów obiektu są (Korbicz *i in.*, 2004):

- residua generowane na podstawie modeli obiektów,
- binarne lub wielowartościowe sygnały powstałe w wyniku kwantowania wartości residuów,
- binarne lub wielowartościowe sygnały generowane w wyniku zastosowania różnych metod detekcji uszkodzeń,
- parametry statystyczne (cechy) opisujące właściwości sygnałów zmiennych procesowych lub residuów,
- zmienne procesowe.

Istnieje wiele różnego rodzaju metod lokalizacji uszkodzeń i rozpoznawania stanu, które najogólniej można podzielić na metody wnioskowania automatycznego i metody klasyfikacji (Isermann, 2006). W metodach tych stosowane są różnorodne modele umożliwiające zapis relacji diagnostycznych. Do najważniejszych należy zaliczyć:

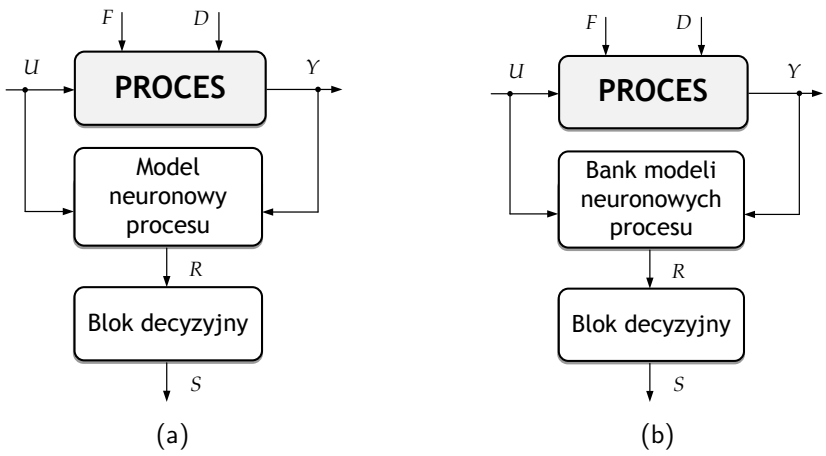
- modele odwzorowujące (Kościelny, 2001) takie jak: binarna macierz diagnostyczna, binarne drzewa diagnostyczne, reguły diagnostyczne, system informacyjny, przybliżony system informacyjny, tablice decyzyjne;
- modele pozyskane metodami uczenia maszynowego lub metodami rozpoznawania obrazów (Korbicz *i in.*, 2004; Moczulski, 2002);
- modele neuronowe, modele rozmyte i neuronowo-rozmyte, komitety modeli, wielosieci (Duch *i in.*, 2000; Frank i Koppen-Seliger, 1997; Korbicz *i in.*, 2004; Patton, Uppal i Lopez-Toribio, 2000; Qipeng *i in.*, 2003; Ruiz *i in.*, 1999; Satoh *i in.*, 2001);

- sieci stwierdzeń (Cholewa, 2006; Cholewa, 2008) i sieci przekonań (Bednarski *i in.*, 2004);
- modele i wielomodele diagnostyczne: modele wielowarstwowe, modele wieloaspektowe, modele wielomodułowe (Cholewa i Rogala, 2008; Cholewa *i in.*, 2008; Rzydzik, 2007; Skupnik, 2008; Wojtusik, 2006).

Na szczególne wyróżnienie zasługuje koncepcja wielomodeli diagnostycznych (Wojtusik, 2006), która pozwala na znacznie szersze wykorzystanie zalet technik elementarnych. Podobnie jak ma to miejsce w przypadku komitetów modeli, wielomodele mogą być tworzone z zastosowaniem różnorodnych technik modelowania (np. z modeli interpolacyjnych, sieci neuronowych, modeli regresyjnych, sieci Bayesa i wielu innych). Można powiedzieć, że jest to kolejny krok poczyniony w kierunku eliminacji niedoskonałości metod elementarnych.

2.5. Modele neuronowe w detekcji uszkodzeń

Różne strategie stosowania sieci neuronowych w systemach detekcji uszkodzeń, które zgodne są z ogólną koncepcją diagnozowania z zastosowaniem modelu procesu, można znaleźć m.in. w pracach (Korbicz *i in.*, 2004; Kościelny, 2001; Kościelny i Bartyś, 2004; Patan *i in.*, 2008). Wyróżnia się dwa podstawowe sposoby diagnozowania (patrz Rys. 2.4): a) na podstawie modelu neuronowego procesu utworzonego dla stanu nominalnego; b) na podstawie banku modeli neuronowych, które odpowiadają różnym stanom obiektu.

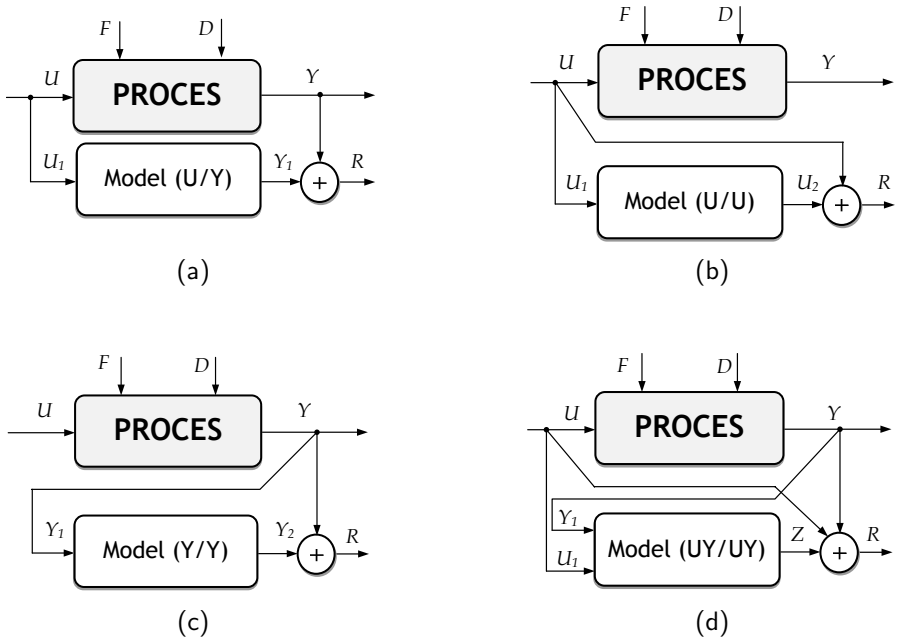


Rys. 2.4: Koncepcje detekcji uszkodzeń z zastosowaniem modeli neuronowych procesów (Korbicz *i in.*, 2004)

Notacja przyjęta na Rys. 2.4 jest następująca: U - zbiór wejść obiektu, Y - zbiór wyjść obiektu, R - zbiór residuów, S - zbiór sygnałów diagnostycznych. Rys. 2.5 przedstawia cztery sposoby budowy modeli neuronowych relacji pomiędzy zmiennymi procesowymi, które mogą zostać zastosowane do detekcji uszkodzeń. Zgodnie z nim wyróżniamy:

a) model wejściowo-wyjściowy, gdzie wejście sieci stanowi podzbiór wejść obiektu ($U_1 \subset U$), a wyjście sieci stanowi podzbiór wyjść obiektu ($Y_1 \subset Y$),
b) model wejściowo-wejściowy, gdzie wejście sieci stanowi podzbiór wejść obiektu ($U_1 \subset U$) i wyjście sieci stanowi podzbiór wejść obiektu ($U_2 \subset U$),
c) model wyjściowo-wyjściowy, gdzie wejście sieci stanowi podzbiór wyjść obiektu ($Y_1 \subset Y$) i wyjście sieci stanowi podzbiór wyjść obiektu ($Y_2 \subset Y$),
d) model mieszany, którego szczególnym przypadkiem jest model odwrotny, gdzie wejście sieci stanowi podzbiór wejść i wyjść obiektu $(U_1 \cup Y_1) \subset (U \cup Y)$ i wyjście sieci stanowi podzbiór wejść i wyjść obiektu $Z \subset (U \cup Y)$.

W przypadku schematu diagnozowania z Rys. 2.4a, w którym zastosowano model neuronowy utworzony zgodnie z dowolnie wybraną konfiguracją pokazaną na Rys. 2.5, oczekuje się, że wartości generowanych residuów będą nieznacznie różnić się od zera w sytuacjach braku uszkodzenia, a w sytuacji, gdy uszkodzenie występuje, różnica ta będzie znacząca. W zastosowaniach praktycznych, niezależnie od przyjętej techniki modelowania, utworzony model zawsze będzie w mniejszym lub większym stopniu odbiegał od rzeczywistości (nie jest możliwe utworzenie dokładnego modelu obiektu), co powoduje, że nie zawsze uzyska się pożądany efekt.



Rys. 2.5: Podstawowe konfiguracje wejść/wyjść modeli neuronowych procesów stosowanych do detekcji uszkodzeń

Wartości residuum, generowane z zastosowaniem modelu neuronowego procesu, są więc uzależnione od kilku czynników. Ogólnie zależność tę możemy zapisać jako:

$$r = y_{\mathrm{ob}} - y = \hbar\left(f, \delta_z, \delta_m\right), \tag{2.1}$$

gdzie f - uszkodzenie, $\hbar$ - zazwyczaj nieznana nieliniowa zależność funkcyjna, δ_z - niepewność powiązana z sygnałem y_{ob} zarejestrowanym na obiekcie, która wynika z za-

kłóceń niemierzalnych i szumów pomiarowych, δ_m - niepewność związana z sygnałem y wyliczanym z zastosowaniem modelu wynikająca z błędów modelowania.

Na niepewność modelu neuronowego składają się błędy strukturalne i błędy parametryczne (Korbicz i Witczak, 2005; Korbicz, 2006). Pierwszy typ błędu wynika z potrzeby doboru określonego typu sieci oraz jej struktury (nieposiadającej interpretacji i często nieodpowiadającej strukturze modelowanego obiektu). Błędy parametryczne powodowane są niedoskonałościami metod estymacji nieznanych parametrów sieci. Na błędy tego rodzaju wpływa również jakość i liczebność zbioru danych, za pomocą których model był utworzony.

Biorąc pod uwagę przytoczone powyżej fakty, należy w taki sposób projektować i tworzyć modele neuronowe, aby residua uzyskiwane za ich pomocą w jak najmniejszym stopniu uzależnione były od zakłóceń, szumów i błędów modelowania, a z drugiej strony były czułe na uwzględniane uszkodzenia.

2.6. Odporna detekcja uszkodzeń

Przytoczone w poprzednim punkcie argumenty powodują, że celowe jest rozwijanie odpornych (ang. *robust*) metod detekcji uszkodzeń (Chen i Patton, 1998). Problemy te nasilają się zwłaszcza wtedy, kiedy rozpatruje się uszkodzenia drobne (ang. *soft faults*) lub uszkodzenia będące w początkowej fazie (ang. *incipient faults*). Na przykład przyjęcie zbyt małej wartości progów decyzyjnych służących do oceny wartości residuów prowadzi zazwyczaj do generacji dużej liczby fałszywych alarmów. Dla zbyt dużych wartości progów system detekcji będzie niewrażliwy na uszkodzenia drobne i narastające (jeżeli celem diagnostyki jest wczesne wykrywanie uszkodzeń, to wówczas rośnie prawdopodobieństwo występowania fałszywych alarmów).

Od wielu lat rozwijane są liczne koncepcje diagnozowania gwarantujące odporność algorytmów detekcyjnych na różnego rodzaju błędy i niepewności. Metody prezentowane w literaturze możemy podzielić na metody aktywne i pasywne (Korbicz, 2006; Patan *i in.*, 2008). Pierwsza grupa metod związana jest z samym generowaniem residuów i polega na odpowiednich modyfikacjach struktury modelu i sposobu jego identyfikacji tak, aby uzyskać residua niewrażliwe na różnego rodzaju błędy i niepewności, a z drugiej strony wrażliwe na występujące w obiekcie uszkodzenia. W wyniku takich działań możliwe jest ograniczenie wpływu zakłóceń oraz minimalizacja niepewności strukturalnej i parametrycznej. W tej grupie wyróżniamy głównie:

- obserwatory stanu o nieznanym wejściu (Korbicz *i in.*, 2004; Korbicz, 2006; Witczak, 2003; Witczak, 2007),
- metody oparte na filtracji optymalnej (Chen i Patton, 1998; Korbicz *i in.*, 2004),
- optymalne odporne równania parzystości (Chen i Patton, 1998; Korbicz *i in.*, 2004; Kościelny, 2001).

Druga grupa metod związana jest z częścią decyzyjną systemu detekcji, gdzie idea oceny residuów zazwyczaj polega na wykorzystaniu adaptacyjnych progów decyzyjnych. Metody

wyznaczania progów możemy podzielić na:

- metody polegające na estymacji niepewności (Korbicz, 2006; Korbicz i Mrugalski, 2008; Witczak *i in.*, 2006),
- metody polegające na modelowaniu niepewności (Patan, 2008b).

Jako sposób redukcji niepewności wynikających z błędów modelowania można podać następujący przykład. Stosując tradycyjne zbiory rozmyte (typu 1.), trudno jest modelować bezpośrednio niepewność wynikającą z faktu, że wiedza pozyskana od kilku ekspertów dotycząca przesłanek i konkluzji rozmytych reguł warunkowych nie jest jednakowa. Z drugiej strony automatyczna budowa bazy wiedzy niesie ze sobą konieczność korzystania z zakłóconych danych numerycznych. Zbiory rozmyte typu 2. są idealnym środkiem do modelowania tego typu niepewności, gdyż wartości funkcji przynależności są zbiorami rozmytymi. Bazę wiedzy złożoną z I reguł warunkowych stosowanych w systemie rozmytym Mamdaniego-Assilana przedstawić można w formie kanonicznej:

$$\mathcal{R} = \left\{\mathcal{R}^{(i)}\right\}_{i=1}^{I} = \left\{ \text{Jeżeli } \left(\bigwedge_{n=1}^{N} X_n \text{ jest } \widehat{A}_n^{(i)} \right), \text{ to } Y \text{ jest } \widehat{B}^{(i)} \right\}_{i=1}^{I}, \tag{2.2}$$

gdzie $\widehat{A}_1^{(i)}$, $\widehat{A}_2^{(i)}$, ..., $\widehat{A}_N^{(i)}$, $\widehat{B}^{(i)}$ to wartości lingwistyczne reprezentowane przez zbiory rozmyte typu 2. Rozumując analogicznie, można rozważać system rozmyty Takagi-Sugeno-Kanga ze zbiorami rozmytymi typu 2. Wyczerpujący przegląd zagadnień budowy systemów rozmytych typu 2. omówiono w pracy (Łęski, 2008).

Przykładem należącym do drugiej grupy metod może być podejście omówione w pracy (Patan, 2008b), które polega na wyliczaniu progów decyzyjnych za pomocą modelu błędu modelu obiektu (ang. *model error model*). Procedura projektowania tak pomyślanych progów jest następująca. Dla utworzonego modelu obiektu w stanie nominalnym generuje się residua r_i dla bieżącego stanu nominalnego (jednocześnie rejestrując wejścia modelu u_i). Dla tak zgromadzonych danych (u_i,r_i) buduje się model z wykorzystaniem wybranej metody identyfikacji parametrycznej, co prowadzi do utworzenia modelu błędu. Do detekcji uszkodzeń wykorzystuje się sygnały generowane przez oba modele. Podejście to nasuwa jednak pytanie co do wpływu niepewności modelu błędu modelu na przebieg procesu diagnozowania.

2.7. Teoria chaosu w diagnostyce procesów

Teoria chaosu deterministycznego uznawana jest, obok teorii względności oraz mechaniki kwantowej, za jedno z trzech monumentalnych odkryć dwudziestego wieku (Li, 2006; Li *i in.*, 2006). Teoria chaosu w głównej mierze koncentruje swoją uwagę na analizie dynamicznych układów nieliniowych, których zachowanie jest nieregularne. Układy chaotyczne uważa się za pośrednie ogniwo stanowiące połączenie pomiędzy układami deterministycznymi oraz stochastycznymi (Morrison, 1996). Dzieje się tak, ponieważ układy tego

typu są deterministyczne, lecz posiadają własności charakterystyczne dla układów stochastycznych. W matematyce, podstawowe cechy wyróżniające układy chaotyczne to: silna zależność zachowania układu od warunków początkowych (wrażliwość na warunki początkowe), duża czułość układu na zmiany jego parametrów, występowanie co najmniej jednego dodatniego wykładnika Lapunowa. Wykładniki Lapunowa są podstawową miarą chaotyczności i mierzą średnią szybkość rozchodzenia się trajektorii leżących bardzo blisko siebie (Ott, 1997; Wiggins, 2003; Anishchenko *i in.*, 2007). Do liczbowego opisania chaosu służą również takie miary jak entropia metryczna i entropia topologiczna (Ott, 1997).

Okazuje się, że chaos to stan dynamiki powszechnie występujący w wielu układach opisujących różnego rodzaju zjawiska (procesy). W ostatnich latach opisano wiele rodzajów układów fizycznych, w których zaobserwowano nieregularne zachowania dynamiczne nazwane oscylacjami chaotycznymi. Przykładami tego typu układów są (Moon, 2004; Schuster, 1993): wahadło z siłą wymuszenia, płyny w pobliżu progu turbulencji, akceleratory cząstek, biologiczne modele dynamiki populacji, pobudzane komórki serca i wiele innych.

Istnieje kilka głównych powodów zainteresowania tymi układami. Podstawowym czynnikiem jest wzgląd poznawczy. Układy rzeczywiste są z natury nieliniowe, dlatego wielu badaczy, próbując znaleźć przyczyny warunków generacji ruchu chaotycznego, konstruuje narzędzia umożliwiające ich analizę (Awrejcewicz i Mosdorf, 2003; Abarbanel, 1996; Kantz i Schreiber, 1999; Morrison, 1996; Wiggins, 2003).

Z drugiej strony, ten powszechnie występujący stan dynamiki w otaczającej nas rzeczywistości jest niepożądany w niektórych sytuacjach (np. podczas pracy urządzeń elektrycznych lub elektronicznych). Pojawiło się wiele ciekawych prac w obszarze dotyczącym synchronizacji (sterowania) układów chaotycznych (Ott, 1997). Dla przykładu można wymienić np. prace dotyczące stabilizacji rytmu serca (Garfinkel *i in.*, 1992) lub pracy lasera (Roy *i in.*, 1992). Innym przykładem może być zastosowanie diagramów bifurkacyjnych wyznaczanych za pomocą modelu neuronowego procesu do określenia aktualnego stanu tego procesu w celu wypracowania odpowiedniej strategii sterowania (Krishnaiah *i in.*, 2006a). Szczegółowy przegląd zagadnień dotyczących sterowania układów chaotycznych przedstawiono np. w pracach (Li, 2006; Fradkov i Evans, 2005).

W latach dziewięćdziesiątych ubiegłego wieku pojawiła się nowa koncepcja wykorzystania tego zjawiska nazwana *"dobrym chaosem"* (ang. *good chaos*) lub też inżynierią chaosu (ang. *chaos engineering*), w której uzyskanie zachowania chaotycznego w rozważanym układzie jest pożądane (Moon, 2004). Można w tym obszarze wymienić co najmniej kilka zastosowań praktycznych, jak na przykład: wprowadzenie drgań chaotycznych o niewielkiej amplitudzie podczas procesu cięcia metalu w celu uzyskania powierzchni cięcia o wysokiej jakości (Moon i Kalmár-Nagy, 2001), zastosowanie systemów chaotycznych do transmisji sygnałów, przesyłu danych i kodowania informacji (Stavroulakis, 2005; Perruquetti i Barbot, 2005), zastosowania systemów chaotycznych jako generatorów liczb pseudolosowych (Li *i in.*, 2006). Powstało również wiele ciekawych patentów urządzeń codziennego użytku (takich firm jak Sanyo, Panasonics, Goldstar), w których zasto-

sowanie elementów teorii chaosu umożliwiło poprawę ich sprawności lub użyteczności. Przykładami takich urządzeń są pralki, zmywarki, mikrofalówki, podgrzewacze powietrza i wiele innych (Hirota, 1995; Moon, 2004).

Istnieją również innowacyjne koncepcje zastosowania teorii chaosu w układach mikroprocesorowych. Ciekawym pomysłem dotyczącym tej idei jest projekt *ChaoLogix* (Ditto *i in.*, 2009). Przewiduje się, że w ramach rozwoju tego projektu zostanie opracowany nowy typ układu scalonego opartego na chaotycznych elementach przetwarzających, realizujących operacje logiczne (bez potrzeby rozwoju nowej technologii wytwarzania). Powinno to przyczynić się do uzyskania komputerowych jednostek obliczeniowych (procesorów), których szybkość obliczeń szacuje się na poziomie 100 GHz i większym (Graham-Rowe, 2009).

Teoria chaosu znalazła również swoje zastosowanie w szeroko rozumianej diagnostyce. Początkowo największe zainteresowanie rozwojem metod opartych na teorii chaosu wiązano z diagnostyką medyczną (West, 1991). Bezpośrednie zastosowanie teorii chaosu w diagnostyce chorób serca znajdują na przykład: analiza fraktalna trajektorii fazowej rytmu serca (Goldberger, 1992), ocena ryzyka nagłego zatrzymania krążenia na podstawie miar złożoności (trajektorii interwałów RR sygnałów EKG) bazujących na entropii Shannona oraz dynamice symbolicznej (Żebrowski *i in.*, 2000).

Badania teoretyczne i eksperymentalne związane z zastosowaniem teorii chaosu dla celów diagnostyki technicznej, w tym diagnostyki procesów, pozwalają na wydzielenie dwóch głównych kierunków działań badaczy. Pierwszy kierunek dotyczy rozwoju metod pasywnych, natomiast drugi związany jest z rozwojem metod aktywnych.

W metodach pasywnych w celu diagnozowania stosowane są różnego rodzaju miary wypracowane na gruncie teorii chaosu (estymaty wykładników Lapunowa, estymaty wymiaru fraktalnego). Miary te wyznaczane są głównie dla odpowiednio przetworzonych zmiennych procesowych. Ich zmiany traktowane są jako symptomy pojawiających się w obiekcie uszkodzeń. Ciekawa jest koncepcja badania zmian stanu układu z zastosowaniem tzw. diagramów rekurencyjnych (ang. *recurrence plots*), których podstawy opisane są w pracy (Eckmann *i in.*, 1987). Już w tej pracy autorzy wskazywali, że narzędzie to może służyć do diagnozowania układów dynamicznych. Początkowo diagramy rekurencyjne stosowano do wizualizacji zachowania trajektorii badanego układu w przestrzeni fazowej. Wprowadzenie przez C. Zbiluta i C. Webbera tzw. analizy ilościowej diagramów rekurencyjnych (ang. *Recurrence Quantification Analysis*, RQA) pozwoliło na ilościowy opis złożoności struktur (np. linii diagonalnych i pionowych) zawartych w diagramie, co znacznie rozszerzyło możliwości tej metody. Analiza układów dynamicznych z zastosowaniem tego narzędzia pozwala na identyfikację np. punktów bifurkacyjnych, przejść intermitencyjnych, stanów laminarnych itp. (Marwan, Romano, Thiel i Kurths, 2007). Przykłady dotyczące metod pasywnych znaleźć można na przykład w pracach (Boguś i Merkisz, 2005; Bi-qiang Du *i in.*, 2008; Nichols *i in.*, 2006; Tykierko, 2008).

Diagnozowanie z zastosowaniem metod aktywnych polega na bezpośrednim wykorzystaniu układów chaotycznych. Najbardziej popularnym przykładem tego typu podejścia jest odpowiednie zastosowanie oscylatorów chaotycznych (np. oscylatora Duffinga).

Wykorzystuje się w tym celu dwie podstawowe własności układów chaotycznych, a mianowicie ich wrażliwość na zmiany parametrów oraz ich wrażliwość na zmiany warunków początkowych (Birx i Pipenberg, 1992). Okazuje się, że w niektórych układach chaotycznych przekłada się to na dużą wrażliwość tych układów na sygnały wymuszenia o określonej postaci przy jednoczesnej odporności tych układów na zakłócenia (Wang *i in.*, 1999). Symptomy zmiany stanu obiektu rozważań obserwowane są pośrednio poprzez zmiany zachowania układu chaotycznego, którego wejściem jest obserwowana na obiekcie zmienna procesowa. Zmiany zachowania układu chaotycznego polegają na przejściach ze stanu układu określanego jako chaotyczny do stanu laminarnego (zachowanie w przybliżeniu okresowe przerywane przez *wybuchy* o skończonym czasie trwania). Stan laminarny świadczy o określonej postaci sygnału wymuszenia, a co za tym idzie o stanie obiektu. Można znaleźć przykłady zastosowania tego podejścia w diagnostyce maszyn (Li i Qu, 2007), jak również w diagnostyce procesów (Song *i in.*, 2009).

Jak dotąd istnieje niewiele prac przeglądowych, wykazujących praktyczne możliwości wykorzystania teorii chaosu w rzeczywistych układach diagnostyki obiektów technicznych (Aihara, 2002; Hirota, 1995; Iokibe, 1997). Na podstawie własnego przeglądu literatury w tym zakresie autor rozprawy zauważa, że praktycznych zastosowań przemysłowych teorii chaosu w zakresie diagnostyki obiektów technicznych jest ciągle niewiele, ponieważ teoria chaosu rozwijana jest od niedawna.

2.8. Podsumowanie

W rozdziale przedstawiono podstawowe koncepcje diagnostyki procesów ze szczególnym uwzględnieniem metod diagnozowania wspartego modelami. Scharakteryzowano grupy modeli procesów do detekcji uszkodzeń uzyskiwane metodami klasycznymi (tj. równania fizyczne, równania zmiennych stanu, transmitancje, obserwatory stanu, modele parametryczne) oraz z zastosowaniem metod sztucznej inteligencji (np. sieci neuronowe, systemy rozmyte, systemy hybrydowe). Dokonano również syntetycznego przeglądu metod i środków dotyczących lokalizacji uszkodzeń i rozpoznawania stanu obiektu. Poddano szczegółowej analizie sposób detekcji uszkodzeń z zastosowaniem modelu neuronowego procesu, wskazując słabe punkty tej techniki. Przedstawiono podstawowe koncepcje odpornej detekcji uszkodzeń oraz omówiono wybrane sposoby redukcji niepewności wynikających z zakłóceń niemierzalnych, szumów pomiarowych i błędów modelowania. W końcowej części rozdziału omówiono najważniejsze sposoby diagnozowania wykorzystujące elementy teorii chaosu.

Biorąc pod uwagę trudności, jakie związane są z budową modeli procesów do detekcji uszkodzeń metodami klasycznymi, można zauważyć, że coraz częściej sięga się do metod obliczeń miękkich. Jednym z najbardziej skutecznych sposobów budowy modeli procesów są niewątpliwie metody bazujące na sztucznych sieciach neuronowych. Istnieje wiele klas struktur neuropodobnych umożliwiających tworzenie przybliżonych modeli procesów, które szczególnie dobrze nadają się do detekcji uszkodzeń. Jak zostało to zauważone, modele tego typu wymagają dalszego rozwoju, co może przyczynić się do ograniczenia

błędów strukturalnych i parametrycznych modelowania. Z drugiej strony brak możliwości całkowitej eliminacji tego rodzaju błędów (oraz błędów innych typów) zmusza projektantów systemów diagnostycznych do opracowywania pasywnych metod detekcji uszkodzeń uwzględniających niepewności pochodzące z różnych źródeł.

W opinii autora rozprawy szczególnie duże nadzieje można wiązać z zastosowaniem teorii chaosu w układach diagnostycznych. Ciekawym podejściem może być budowa systemów diagnostycznych (wykorzystujących model obiektu) z zastosowaniem tzw. inżynierii chaosu, w której w bezpośredni sposób wykorzystuje się własności układów chaotycznych. Można również zauważyć, że brak jest np. metod detekcji uszkodzeń bazujących na modelu procesu, w których do oceny residuów stosowane byłyby analizy za pomocą narzędzi wywodzących się z teorii chaosu (np. analizy diagramów rekurencyjnych wyznaczanych dla residuów). Opisane w dalszej części pracy badania autora są próbą zastosowania wybranych odkryć teorii chaosu w diagnostyce procesów technicznych.

Rozdział 3

Rekurencyjne sieci neuronowe

Przedmiotem rozważań niniejszego rozdziału są rekurencyjne sieci neuronowe, których wykorzystanie stanowi jedno z możliwych podejść w zagadnieniach modelowania obiektów technicznych. Działanie, którego wynikiem jest model neuronowy obiektu, zgodne jest z ogólną teorią identyfikacji systemów (Söderström i Stoica, 1997). Jest to podejście, w którym model tworzony jest z użyciem danych pochodzących z eksperymentu. System poddaje się szeregowi doświadczeń, a następnie dobiera się parametry modelu tak, aby pasował on do danych doświadczalnych. Innym źródłem danych mogą być bazy danych historycznych zgromadzonych na obiekcie rzeczywistym (Wang, 2000).

Podstawową właściwością rekurencyjnych sieci neuronowych jest ich zdolność gromadzenia informacji oraz późniejsze jej przetwarzanie (Duch *i in.*, 2000; Gupta *i in.*, 2003; Maydl i Sick, 2000; Medsker i Jain, 1999; Osowski, 2006b; Żurada *i in.*, 1996). Z technicznego punktu widzenia właściwość tę uzyskano, wprowadzając do struktury sieci neuronowej połączenia rekurencyjne. Ze względu na różnorodność sposobów realizacji połączeń pomiędzy neuronowych o charakterze sprzężeń zwrotnych, przyjęto następujący podział sieci dynamicznych (Korbicz *i in.*, 2004; Patan, 2008b; Sinha *i in.*, 2000; Tsoi i Back, 1994):

a) **Sieci globalnie rekurencyjne** – głównym założeniem przyjmowanym przy projektowaniu tego typu struktur jest możliwość połączenia każdego neuronu z dowolnym innym. Dynamika sieci reprezentowana jest przez połączenia sprzężone lub skrośne w dwóch podstawowych wariantach: architektury w pełni rekurencyjnej lub architektury częściowo rekurencyjnej (ang. *fully/partially recurrent neural networks*);

b) **Sieci lokalnie rekurencyjne** – struktury o tej topologii złożone są z neuronów dynamicznych pogrupowanych w warstwy, w których sygnał przepływa w jednym kierunku od wejść poprzez warstwy ukryte do wyjść. Jest to powodem, że dla architektur tego typu często używa się następującej nazwy: sieci lokalnie rekurencyjne globalnie jednokierunkowe (ang. *locally recurrent globally feedforward networks*).

Zarówno struktury globalnie, jak i lokalnie rekurencyjne mają swoje reprezentacje w postaci nieliniowych równań stanu i wyjścia (Griño *i in.*, 2000; Gupta *i in.*, 2003; Pa-

tan, 2008b). Taki zapis struktury sieci umożliwia identyfikację szerokiej klasy nieliniowych układów dynamicznych. Pozwala to również na bezpośrednią adaptację metod analizy stabilności systemów dynamicznych w odniesieniu do struktur neuronowych.

3.1. Struktury w pełni i częściowo rekurencyjne

Wyróżnić można dwa podstawowe przykłady sieci w pełni rekurencyjnych (Osowski, 2006b): sieć Hopfielda (ang. *Hopfield's neural network*) i jej modyfikacje oraz sieć czasu rzeczywistego (ang. *real time neural network*) (Williams i Zipser, 1989). Pierwsza z nich stosowana jest do pełnienia funkcji pamięci asocjacyjnych. Drugą zaprojektowano do przetwarzania sygnałów w czasie rzeczywistym. Ponieważ sieci tego typu mają połączenia rekurencyjne pomiędzy wszystkimi neuronami, pociąga to za sobą kilka niepożądanych efektów, takich jak: złożoność procesu dostrajania parametrów i jego wolna zbieżność, problem stabilności modelu, niski rząd modelu. Te wady powodują, że sieci tego typu nie znajdują aprobaty w praktycznych zastosowaniach (Patan, 2008b). W dalszej części rozdziału omówione zostaną wyłącznie te sieci globalnie rekurencyjne, które autor stosował w swoich badaniach (Przystałka, 2007a; Przystałka, 2008; Przystałka, 2009).

3.1.1. Sieci Jordana i Elmana

Struktury Jordana i Elmana w swojej pierwotnej formie mają budowę podobną do dwuwarstwowych struktur jednokierunkowych, z tą różnicą, że rozbudowane są o elementy rekurencyjne w tak zwanej warstwie kontekstowej. Neurony zrealizowane są za pośrednictwem jednostkowych członów opóźniających, będąc odpowiednikami elementów w warstwie wyjściowej (sieć Jordana) lub w warstwie ukrytej (sieć Elmana). Zadaniem dodatkowej warstwy jest opóźnienie sygnałów związanych ze stanem wewnętrznym sieci lub stanem jej wyjść wyłącznie o jeden takt. Znane są również implementacje bardziej rozbudowanych struktur tych sieci, np. hierarchiczne sieci Elmana lub wielowarstwowe sieci Jordana. Niemniej jednak nie rozwiązują one problemu dotyczącego jednostkowego opóźnienia sygnałów docierających do warstwy ukrytej.

W. Wilson zaproponował wielokontekstowe architektury będące modyfikacjami struktur sieci Jordana i Elmana, które mają możliwość pamiętania stanów wewnętrznych lub wyjść z i poprzednich kroków czasowych (Wilson, 1993; Wilson, 1995). Struktury tego typu pokazano na Rys. 3.1 i 3.2.

Wyjścia sieci wyznacza się z zależności:

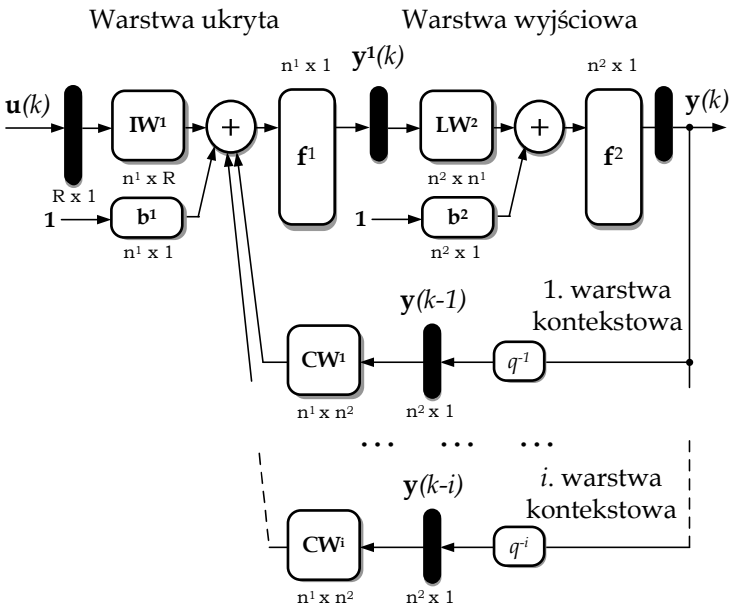
a) wielokontekstowe sieci Jordana:

$$\begin{aligned} \mathbf{y}(k) &= \mathbf{LW}^2\mathbf{f}^1(\mathbf{IW}^1\mathbf{u}(k) + \mathbf{CW}^1\mathbf{y}(k-1) + \cdots \\ &\quad \cdots + \mathbf{CW}^i\mathbf{y}(k-i) + \mathbf{b}^1) + \mathbf{b}^2, \end{aligned} \tag{3.1}$$

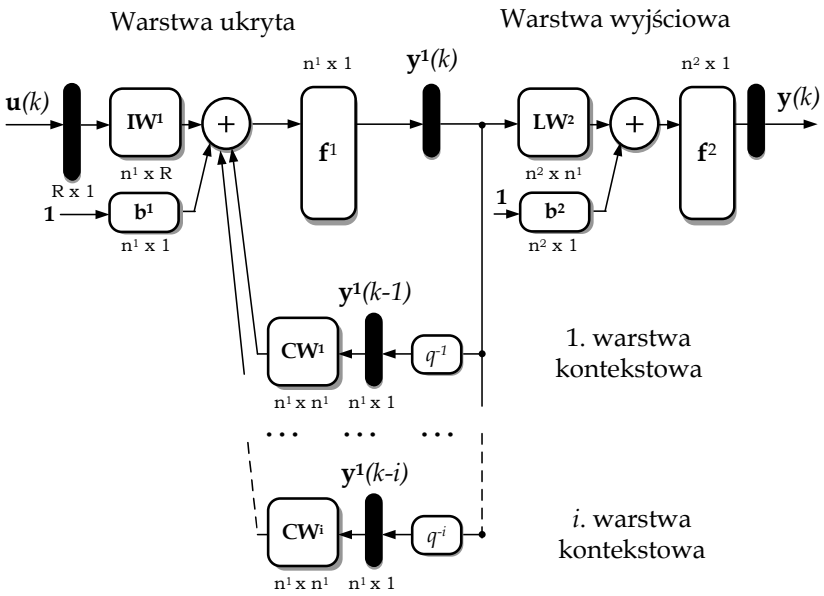
b) wielokontekstowe sieci Elmana:

$$\begin{aligned}\mathbf{y}(k) &= \mathbf{LW}^2\mathbf{f}^1(\mathbf{IW}^1\mathbf{u}(k) + \mathbf{CW}^1\mathbf{y}^1(k-1) + \cdots \\ &\quad \cdots + \mathbf{CW}^i\mathbf{y}^1(k-i) + \mathbf{b}^1) + \mathbf{b}^2,\end{aligned} \tag{3.2}$$

gdzie $\mathbf{IW}^i$, $\mathbf{LW}^i$ - macierze wag odpowiednio warstw wejściowych oraz ukrytych, $\mathbf{CW}^i$ - macierze wag warstw kontekstowych, $\mathbf{b}^i$ - wektor wartości progowych neuronów i-tej warstwy, $\mathbf{f}^i$ - wektor funkcji wyjścia neuronów i-tej warstwy.



Rys. 3.1: Uogólniona struktura sieci Jordana



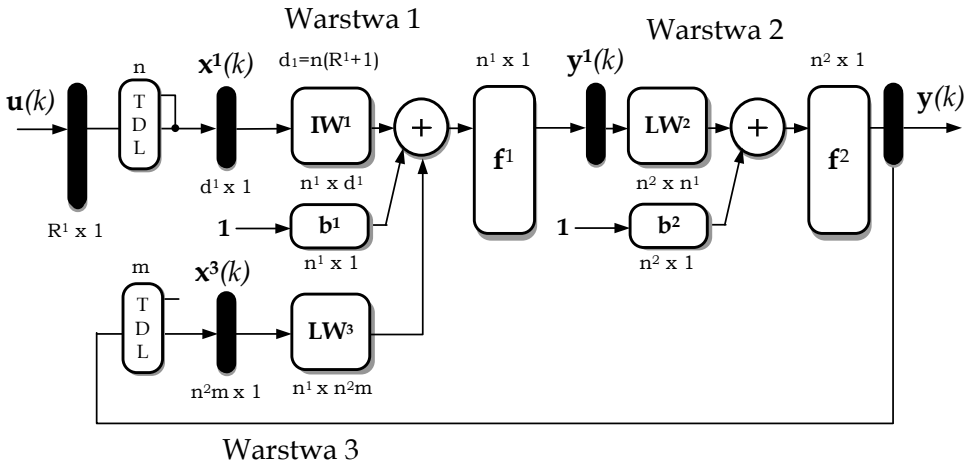
Rys. 3.2: Uogólniona struktura sieci Elmana

Sieci tego typu projektowane były głównie do zadań rozpoznawania sekwencji słów języka naturalnego (przewidywanie kolejnych wyrazów w zdaniu). Dobre właściwości aproksymacyjne oraz cechy umożliwiające gromadzenie informacji i późniejsze ich przetwarzanie umożliwiły zastosowanie tych sieci w typowym zadaniu identyfikacji systemów dynamicznych (Tomanek *i in.*, 2006; Tomanek *i in.*, 2007).

3.1.2. Sieci typu NNARX

Innym sposobem uzyskania własności dynamicznych modelu neuronowego jest stosowanie statycznych struktur sieci oraz specjalnie przygotowanych wzorców uczących (Korbicz *i in.*, 2004; Korbicz *i in.*, 1994; Kuschewski *i in.*, 1993; Narendra i Parthasarathy, 1990; Tipsuwanporn *i in.*, 2001). Umożliwia to uwzględnienie sygnałów zarówno wejściowych jak i wyjściowych identyfikowanego obiektu z aktualnej i poprzednich chwil czasu. W ten sposób do struktury jednokierunkowej wprowadza się sprzężenie zwrotne z wyjścia, przez co sieć statyczna uczy się zależności rekurencyjnej np. metodą wstecznej propagacji błędu (szeregowo−równoległy model identyfikacji). Zakładając, że różnica pomiędzy wyjściem obiektu i modelu neuronowego jest bliska zeru, model w układzie szeregowo − równoległym można zastąpić modelem w układzie równoległym (Korbicz *i in.*, 1994; Narendra i Parthasarathy, 1990).

Strukturę sieci NNARX, umożliwiającą identyfikację szerokiej gamy układów dynamicznych, przedstawia Rys. 3.3 (The MathWorks, 2007). Sieci rekurencyjne tego typu są wynikiem rozważań zapoczątkowanych przez (Narendra i Parthasarathy, 1990), a następnie ogólnie stosowanych przez wiele zespołów badawczych, m.in. (Korbicz *i in.*, 2004; Korbicz *i in.*, 1994; Maydl i Sick, 2000; Siegelmann *i in.*, 1997; Tsung-Nan Lin *i in.*, 1999).



Rys. 3.3: Globalnie rekurencyjna sieć neronowa typu NNARX (The MathWorks, 2007)

Wyjście sieci w dyskretnej chwili k można wyznaczyć z równania

$$\mathbf{y}(k) = \mathbf{LW}^2\mathbf{f}^1\left(\mathbf{IW}^1\mathbf{x}^1(k) + \mathbf{LW}^3\mathbf{x}^3(k) + \mathbf{b}^1\right) + \mathbf{b}^2, \tag{3.3}$$

przy czym wektory stanu sieci przyjmuje się jako

$$\mathbf{x}^1(k) = \mathbf{q}^{-\tau_n}\mathbf{u}(k) = [\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-n)]^T, \tag{3.4}$$

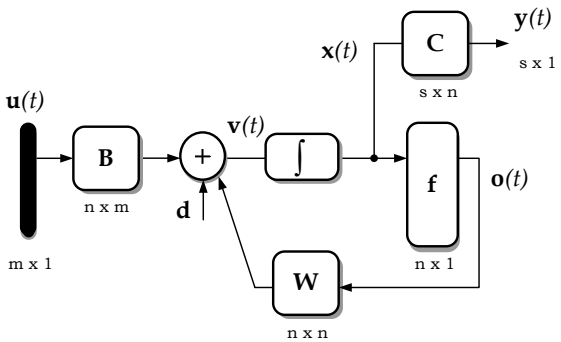
$$\mathbf{x}^3(k) = \mathbf{q}^{-\tau_m}\mathbf{y}(k) = [\mathbf{y}(k-1), \mathbf{y}(k-2), \ldots, \mathbf{y}(k-m)]^T. \tag{3.5}$$

gdzie $\mathbf{q}^{-\tau_n}$ i $\mathbf{q}^{-\tau_m}$ - reprezentują wektorowy zapis operatora opóźnienia.

Struktura neuronowa typu NNARX umożliwia realizacje szeregowo – równoległego oraz równoległego modelu identyfikacji. Jednakże zalecane jest stosowanie pierwszego podejścia i następnie przełączanie do układu równoległego, tak jak w przypadku struktur statycznych. W przypadku identyfikacji w układzie równoległym niezbędne jest stosowanie gradientowych algorytmów uczenia opartych na dynamicznej wstecznej propagacji błędu (Bajramovic *i in.*, 2004; De Jesus i Hagan, 2001; Gruber i Sick, 2003; Werbos, 1990).

3.1.3. Sieci rekurencyjne z czasem ciągłym

Wspólną cechą prezentowanych powyżej modeli neuronowych jest to, że ich stan i wyjścia wyznaczane są w dyskretnych chwilach czasu. Prowadzi to do silnej zależności uzyskiwanych modeli od okresu próbkowania oraz do utraty informacji o zmianach trajektorii modelu pomiędzy punktami przestrzeni fazowej (Gupta *i in.*, 2003). W celu eliminacji tych ograniczeń R. Griño i współpracownicy opracowali architekturę w pełni rekurencyjną opartą na dynamicznych neuronach z czasem ciągłym (Griño *i in.*, 2000). Zaproponowany przez nich model neuronowy można przedstawić jak na Rys. 3.4.



Rys. 3.4: Schemat globalnie rekurencyjnej sieci neuronowej z czasem ciągłym

Opis formalny modelu neuronowego z czasem ciągłym jest następujący:

$$\dot{\mathbf{x}}(t) = -\mathbf{U}(\alpha \otimes \mathbf{x}(t)) + \mathbf{Wf}(\mathbf{x}(\mathbf{t})) + \mathbf{Bu}(t) + \mathbf{d}, \tag{3.6}$$

$$\mathbf{y}(t) = \mathbf{Cx}(t), \tag{3.7}$$

gdzie $\mathbf{u}(t)$ – wejście sieci w chwili t, $\mathbf{x}(t)$ – wektor stanu w chwili t, $\mathbf{d}$ – wektor wartości progowych, $\mathbf{y}(t)$ – wyjście sieci w chwili t, $\mathbf{W}$ – macierz wag wejść sprzężonych, $\mathbf{B}$ – macierz wag wejść modelu, $\mathbf{C}$ – macierz wag wyjść modelu, $\mathbf{U}$ – macierz umożliwiająca wektorowy zapis stałych czasowych, $\mathbf{f}$ – nieliniowy operator transformacji.

Jeśli przyjmie się liniową funkcję wyjścia oraz zrezygnuje ze specjalnego zapisu macierzy stałych czasowych, to otrzyma się klasyczne równania stanu z czasem ciągłym. Ponadto, przyjmując $\mathbf{W} = 0$, otrzymujemy strukturę charakterystyczną dla lokalnie rekurencyjnych sieci neuronowych. Proponowane przez (Griño *i in.*, 2000) podejście umożliwia adaptację metod uczenia sieci neuronowych do wyznaczania parametrów równań przestrzeni stanów, jak również adaptację znanych metod do analizy stabilności modelu neuronowego. Modele neuronowe zapisane w postaci równań stanu były również rozważane w pracach (Gupta *i in.*, 2003; Patan, 2008b).

3.2. Sieci lokalnie rekurencyjne

Dynamiczne właściwości neuronu uzyskuje się, rozszerzając znaną strukturę modelu zaproponowanego przez W. McCullocha i W. Pittsa przez wprowadzenie w odpowiednie miejsca wewnętrznych sprzężeń zwrotnych. Podejście z użyciem lokalnego sprzężenia zwrotnego w jednostce przetwarzającej rozwijane jest przez wiele zespołów badawczych. Do najważniejszych pozycji, w których można znaleźć wyniki badań dotyczących modelowania neuronowego opartego na takich elementach, należy zaliczyć między innymi prace (Ayoubi, 1994; Gupta *i in.*, 2003; Korbicz *i in.*, 2004; Patan *i in.*, 2008; Patan, 2008b; Sinha *i in.*, 2000; Tsoi i Back, 1994). Autorzy opracowań przytaczają kilka sposobów umożliwiających wprowadzenie dynamiki do struktury podstawowego elementu przetwarzającego, co prowadzi do uzyskania dokładniejszego odwzorowania dynamiki obiektu (procesu).

Liczba prac poświęconych tej tematyce jest znacząca, co powoduje, że nie zawsze jest możliwe usystematyzowanie i zwięzłe przedstawienie opublikowanych wyników badań. Jedną z najciekawszych pozycji literaturowych, która dotyczy projektowania sieci lokalnie rekurencyjnych w zadaniach diagnostyki procesów, jest monografia K. Patana (2008). Autor porusza w niej wiele problemów dotyczących projektowania tego typu sieci, przedstawiając równocześnie szereg metod umożliwiających zastosowanie struktur neuronów dynamicznych w systemach diagnostyki procesów.

W niniejszym przeglądzie wyróżnione zostaną wybrane dynamiczne modele neuronu, które autor rozprawy uznał za szczególnie istotne dla omawianego podejścia.

3.2.1. Modele neuronu wg A. Yazdizadeha i K. Khorasaniego

A. Yazdizadeh i K. Khorasani zaproponowali dwie struktury sieci neuronowych (Yazdizadeh i Khorasani, 1997) dla typowego zadania identyfikacji systemów dynamicznych, które zaliczane są do systemów dynamicznych pierwszego typu (Narendra

i Parthasarathy, 1990). Wyjście tego systemu rozważane jest jako liniowa kombinacja wyjściowych wartości z poprzedniej chwili czasu oraz pewnej nieliniowej funkcji wejścia i jego przeszłych wartości zgodnie z równaniem różnicowym:

$$y(k) = \sum_{i=1}^{N_s} \alpha_i y(k-i) + g(u(k-1), u(k-2), \ldots, u(k-M_s)), \tag{3.8}$$

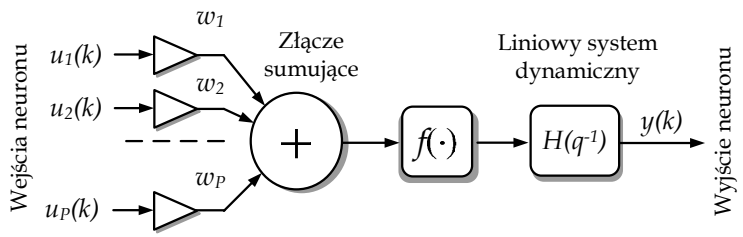
gdzie $u(k), y(k)$ są odpowiednio wejściem i wyjściem systemu. Struktury dynamiczne umożliwiające identyfikację opisanego systemu bazują na dwóch dynamicznych modelach neuronu zaproponowanych przez A. Yazdizadeha i K. Khorasaniego: jednostki przetwarzającej z filtrem rekurencyjnym w bloku wyjściowym oraz neuronu z operatorem opóźnienia w synapsach (patrz Rys. 3.5 i 3.6).

Neuron z filtrem autoregresyjnym

A. Yazdizadeh i K. Khorasani uzyskali model neuronu z filtrem rekurencyjnym, umieszczając liniowy system dynamiczny za funkcją wyjścia w typowym neuronie. Pozwoliło to na realizację dynamicznej relacji pomiędzy wejściami i wyjściem jednostki przetwarzającej. Związek wejściowo-wyjściowy może być reprezentowany przez następujące równanie:

$$y(k) = f\left(\sum_{i=1}^{N} w_i u_i(k)\right) + \sum_{j=1}^{M} a_j y(k-j), \tag{3.9}$$

gdzie $u_i(k)$, $y(k)$, $i = 1, 2, \ldots, N$ reprezentują odpowiednio wejścia i wyjście neuronu w dyskretnej chwili k, $f(\cdot)$ – nieliniowa funkcja wyjścia, w_i – wagi synaptyczne odpowiednio przetwarzające sygnały wejściowe, a_j, $j = 1, 2, \ldots, M$ są współczynnikami charakterystycznymi filtru autoregresyjnego, M jest rzędem filtru.



Rys. 3.5: Neuron z filtrem autoregresyjnym (Yazdizadeh i Khorasani, 1997)

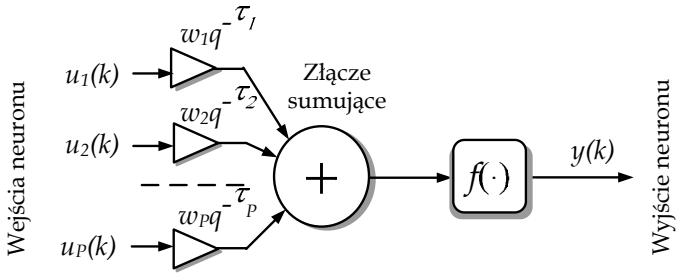
Neuron z operatorem opóźnienia w synapsach

Drugi, proponowany przez przytoczonych powyżej autorów, model jest adaptacją neuronu stosowanego w sieciach z opóźnieniem czasowym (ang. *time delay neural networks*). Tego typu sieci stosowane były zazwyczaj w zadaniach rozpoznawania szeregów czasowych oraz sterowania (Lin *i in.*, 1992; Waibel *i in.*, 1989).

W tym przypadku stan neuronu w dyskretnej chwili k jest reprezentowany przez zależność:

$$y(k) = f\left(\sum_{i=1}^{N} w_i q^{\tau_i} u_i(k)\right) = f\left(\sum_{i=1}^{N} w_i u_i(k-\tau_i)\right), \tag{3.10}$$

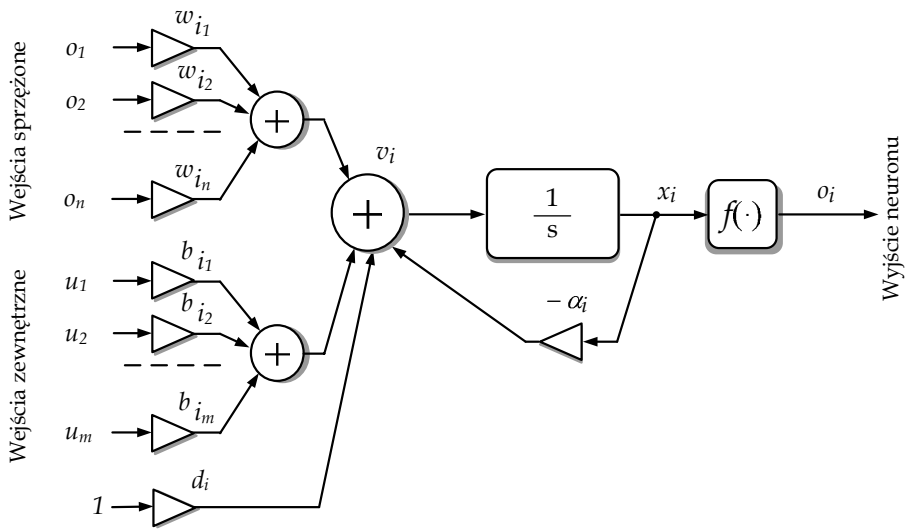
gdzie q^{τ_i} oznacza opóźnienie i-tego wejścia. Wyjście neuronu $y(k)$ ostatniej warstwy dodatkowo przekazywane jest na wejście filtru rekurencyjnego tak jak w poprzednim przykładzie. Jest to niezbędne działanie, aby uzyskać strukturę reprezentatywną dla systemów dynamicznych pierwszego rodzaju (Narendra i Parthasarathy, 1990). Jak wynika z równań (3.9) i (3.10), na bieżący stan neuronu wpływają jego wewnętrzne stany z r poprzednich kroków czasowych. Neurony tego typu stosowane są w jednokierunkowych sieciach neuronowych, przez co umożliwiają adaptację znanych metod uczenia.



Rys. 3.6: Neuron z operatorem opóźnienia w synapsach (Yazdizadeh i Khorasani, 1997)

3.2.2. Neuron z czasem ciągłym

Ciekawe rozwiązanie umożliwiające identyfikację on-line zaproponowali (Griño *i in.*, 2000). Głównym elementem przetwarzającym struktur dynamicznych był model neuronu prezentowany na Rys. 3.7.



Rys. 3.7: Neuron z liniowym systemem dynamicznym typu SISO (Griño *i in.*, 2000)

Jednostki tego typu mogą działać zarówno w sieciach globalnie, jak i lokalnie rekurencyjnych (poprzez odpowiednie zerowanie wag połączeń wejść sprzężonych). Stan i-tej jednostki modelu neuronowego wyrażony jest jako:

$$v_i = \sum_{j=1}^{n} w_{ij} o_j + \sum_{k=1}^{m} b_{ik} u_k + d_i, \tag{3.11}$$

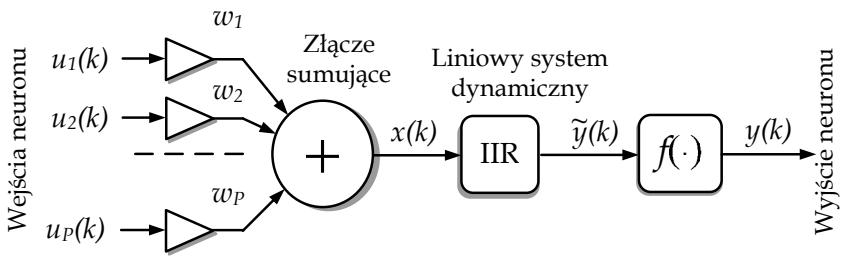
gdzie ważona suma v_i jest liniową kombinacją wyjść o_j jednostek sieci, zewnętrznych wejść u_k oraz progu funkcji wyjścia. Suma v_i stanowi wejście liniowego systemu dynamicznego (pierwszego rzędu) z wyjściem x_i:

$$\dot{x}_i + \alpha_i x_i = v_i, \tag{3.12}$$

przy czym przyjmuje się stałą czasową $\tau_i = 1/\alpha_i$ i wzmocnienie o wartości α_i dla każdego węzła w sieci.

3.2.3. Neuron z filtrem IIR

Innym sposobem uzyskania właściwości dynamicznych dla prostego elementu przetwarzającego było wprowadzenie filtru o nieskończonej odpowiedzi impulsowej pomiędzy blok sumacyjny a blok aktywacji (Ayoubi, 1994; Korbicz *i in.*, 2004). W ten sposób uzyskano uogólnienie modelu neuronu ze sprzężeniem aktywacyjnym (Frasconi *i in.*, 1992). Model ten był przedmiotem wielu ciekawych prac dotyczących systemów diagnostycznych odpornych na niepewność modelu (Patan *i in.*, 2008; Patan, 2008b).



Rys. 3.8: Neuron z filtrem IIR w bloku aktywacji (Korbicz *i in.*, 2004)

Zachowanie takiego neuronu opisać można za pomocą następujących zależności:

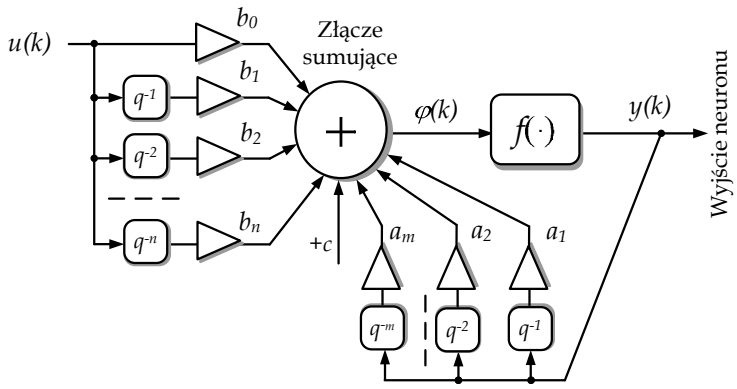
$$x(k) = \sum_{p=1}^{P} w_p u_p(k), \tag{3.13}$$

$$y(k) = f\left(-g \sum_{i=1}^{n} a_i \tilde{y}(k-i) + g \sum_{i=0}^{n} b_i x(k-i) + c\right), \tag{3.14}$$

gdzie $u_p(k)$, $p = 1, \ldots, P$ są wejściami neuronu, P jest liczbą wejść, w_p oznaczają wagi wejściowe, $\tilde{y}(k)$ jest wyjściem filtru, $a_i, i = 1, \ldots, n$ oraz $b_i, i = 0, \ldots, n$ są parametrami filtru, a n oznacza rząd filtru. Ponadto $f(\cdot)$ jest nieliniową funkcją wyjścia, $y(k)$ - sygnałem wyjściowym neuronu, g - współczynnikiem nachylenia, oraz c - progiem funkcji wyjścia.

3.2.4. Neuron G-FGS

Przykładem bardziej złożonego modelu neuronu, w którym istnieje pamięć zarówno poprzednich stanów jego wejść, jak i wyjścia, jest dynamiczny model jednostki przetwarzającej zaproponowany przez P. Frasconiego i współpracowników (Frasconi *i in.*, 1992), a następnie zmodyfikowany i stosowany w pracy (Mastorocostas i Theocharis, 2002).



Rys. 3.9: Uogólniony model neuronu Frasconiego–Gori–Soda (G-FGS)

Ograniczając rozważania do jednego wejścia, co wynika wyłącznie z potrzeby uproszczenia prezentacji na schemacie, działanie tego neuronu można opisać zależnością:

$$y(k) = f\left(\sum_{j=0}^{n} b_j u(k-j) + \sum_{i=1}^{m} a_i y(k-i) + c\right), \tag{3.15}$$

gdzie $u(k)$ oznacza wejście neuronu, $y(k)$ – wyjście neuronu, b_j są współczynnikami połączeń synaptycznych opisujących moc opóźnionych wartości wejść neuronu, a_i są współczynnikami połączeń synaptycznych opisujących wpływ poprzednich wartości wyjść neuronu na jego stan bieżący, c – próg określający poziom aktywności neuronu.

Neuron G-FGS (ang. *generalized Frasconi–Gori–Soda neuron*) stosowany był w dynamicznych rozmytych sieciach neuronowych (ang. *dynamic fuzzy neural networks*), umożliwiając połączenie mechanizmu wnioskowania Takagi-Sugeno-Kanga oraz rozwiązania bazującego na lokalnie rekurencyjnych sieciach neuronowych.

3.2.5. Neuron chaotyczny

Lata 90. poprzedniego wieku zapoczątkowały burzliwe zainteresowanie klasą układów dynamicznych charakteryzujących się deterministycznym chaosem (Kapitaniak i Wojewoda, 1994; Morrison, 1996; Ott, 1997; Peitgen *i in.*, 1995; Radziszewski *i in.*, 2005). Przełożyło się to na wiele prac z zakresu identyfikacji dynamicznych systemów chaotycznych bazujących na miękkich technikach obliczeniowych (Krishnaiah *i in.*, 2006b; Poznyak *i in.*, 1999; Sato i Nagaya, 1996; Song-Ming *i in.*, 2004; Tokuda *i in.*, 2001).

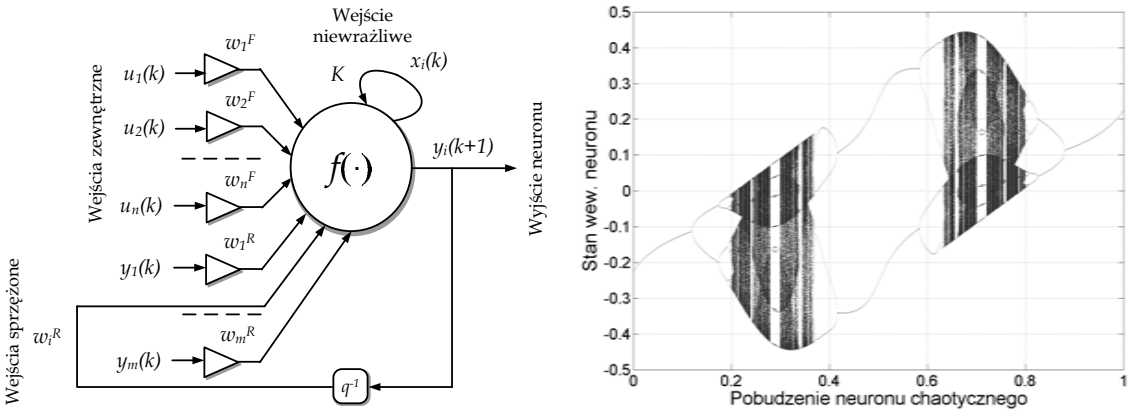
Innym pomysłem było rozwinięcie struktur neuronowych, które charakteryzowały się dynamiką chaotyczną (Aihara *i in.*, 1990; Pasemann, 1997; Sang-Hee Kim *i in.*, 2001).

Jednym ze sposobów wprowadzenia właściwości chaotycznych do struktur neuronowych jest stosowanie neuronów chaotycznych. Przykład takiej jednostki pokazuje Rys. 3.10a (Sang-Hee Kim *i in.*, 2001). Neuron ten jest uproszczonym wariantem jednostki zaproponowanej przez K. Aiharę i współpracowników (Aihara *i in.*, 1990). Jego stan reprezentowany jest przez zależności:

$$x_i(k+1) = Kx_i(k) + \sum_{j=1}^{n} w_{ij}^F u_j(k) + \sum_{j=1}^{m} w_{ij}^R y_i(k), \tag{3.16}$$

$$y_i(k+1) = f(x_i(k+1), \epsilon), \tag{3.17}$$

gdzie w_{ij}^F i w_{ij}^R – wartości połączeń synaptycznych z j-tego zewnętrznego neuronu (lub j-tego wejścia) oraz j-tego sprzężonego neuronu do i-tego neuronu, $f(\cdot)$ – logistyczna funkcja wyjścia neuronu, K – parametr refrakcji określający wpływ poprzednich stanów neuronu na stan bieżący.



(a) Schemat blokowy neuronu (b) Diagram bifurkacyjny

Rys. 3.10: Neuron o dynamice chaotycznej (Sang-Hee Kim *i in.*, 2001)

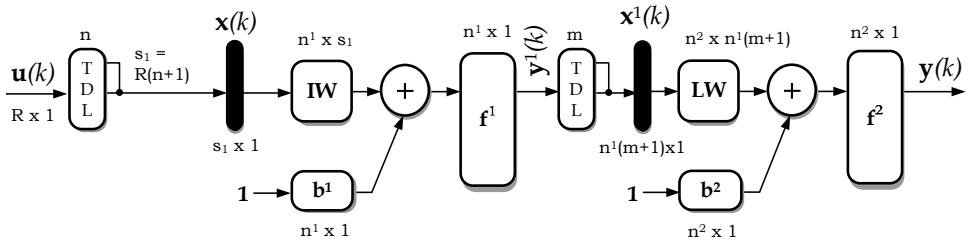
Diagram bifurkacyjny z Rys. 3.10b obrazuje proces podwajania okresu przy krytycznych wartościach wymuszenia $u(k)$ neuronu chaotycznego (3.16-3.17) dla $n = m = 1$, $w_{11}^R = -1$, $w_{11}^F = 1$, $K = 0.9$, $\epsilon = 0.06$. Wygląd diagramu bifurkacyjnego dla tego neuronu uwidacznia kolejne krytyczne wartości wymuszenia, które prowadzą do zachowania charakterystycznego dla ruchu chaotycznego. W zakresie tym pojawiają się również przedziały wartości wymuszenia, dla których otrzymywany jest ruch okresowy (tzw. okna okresowe). Diagram opracowano na podstawie wyników pokazanych w pracy (Sang-Hee Kim i Won-Woo Park, 2003).

3.3. Przykłady sieci neuronów dynamicznych

Poniżej przedstawiono dwa przykłady realizacji sieci neuronów dynamicznych. Pierwszy przykład pokazuje sposób reprezentacji sieci lokalnie rekurencyjnej w zapisie wejściowo-wyjściowym. Drugi przykład prezentuje strukturę sieci lokalnie rekurencyjnej zapisanej w postaci równań stanu i wyjścia.

3.3.1. Sieć z liniami opóźniającymi

A. Waibel oraz jego współpracownicy po raz pierwszy użyli struktur jednokierunkowych z liniami opóźniającymi (Waibel *i in.*, 1989). Stosowana przez nich architektura sieci projektowana była dla problemu dotyczącego rozpoznawania mowy. Struktura Waibela bazowała na modelu neuronu z filtrami o skończonej odpowiedzi impulsowej w synapsach. Szczególnym przypadkiem takiej architektury jest dwuwarstwowa sieć jednokierunkowa pokazana na Rys. 3.11. Linie opóźniające występują w warstwie wejściowej i wyjściowej. Wymaga to użycia jednej z dynamicznych metod wyznaczenia gradientów funkcji błędu dla potrzeb algorytmów trenujących. Jeżeli ograniczy się opóźnienie jedynie do warstwy wejściowej, to bardzo łatwo można zaadaptować gradientowe metody uczenia oparte na statycznej metodzie propagacji wstecznej błędu.



Rys. 3.11: Perceptron wielowarstwowy z liniami opóźniającymi

Wyjście sieci w dyskretnej chwili k wyznaczane jest w następujący sposób:

$$\mathbf{y}(k) = \mathbf{f}^2\left(\mathbf{LWx}^1(k) + \mathbf{b}^2\right), \tag{3.18}$$

przy czym

$$\mathbf{x}^1(k) = \mathbf{q}^{-\tau_m}\mathbf{y}^1(k) = \mathbf{q}^{-\tau_m}\mathbf{f}^1\left(\mathbf{IWx}(k) + \mathbf{b}^1\right), \tag{3.19}$$

oraz

$$\mathbf{x}(k) = \mathbf{q}^{-\tau_n}\mathbf{u}(k) = \left[\mathbf{u}(k), \mathbf{u}(k-1), \ldots, \mathbf{u}(k-n)\right]^T, \tag{3.20}$$

gdzie $\mathbf{f}$ - nieliniowe/liniowe operatory transformacji, $\mathbf{q}^{-\tau}$ – wektorowy zapis operatora opóźnienia przekształcającego wejścia warstwy ukrytej i wyjściowej sieci.

Struktury tego typu stosowane są głównie w zadaniach szeroko rozumianego rozpoznawania (Clouse *i in.*, 1997; Lin *i in.*, 1992), choć znaleźć można również zastosowania dotyczące identyfikacji systemów dynamicznych (Marques *i in.*, 2005; Maydl i Sick, 2000).

3.3.2. Sieć neuronów z filtrem IIR

Kolejnym przykładem sieci lokalnie rekurencyjnej może być sieć o jednej warstwie ukrytej złożonej z neuronów z filtrem o nieskończonej odpowiedzi impulsowej, które opisywane

były w poprzedniej części rozdziału. Zapis formalny sieci w postaci równań stanu zaproponowany w pracy (Patan, 2008b) jest następujący:

$$\mathbf{x}(k+1) = \mathbf{A}\mathbf{x}(k) + \mathbf{W}\mathbf{u}(k), \tag{3.21}$$

$$\mathbf{y}(k) = \mathbf{C}\mathbf{f}\left[\mathbf{G}_2\left(\mathbf{B}\mathbf{x}(k) + \mathbf{D}\mathbf{u}(k) - \mathbf{g}_1\right)\right]^T, \tag{3.22}$$

gdzie $N = v \times r$ oznacza liczbę stanów sieci, $\mathbf{x}(k) \in \mathbb{R}^N$ jest wektorem stanu, $\mathbf{u}(k) \in \mathbb{R}^n$, $\mathbf{y}(k) \in \mathbb{R}^m$ - wektory wejściowe i wyjściowe, $\mathbf{A} \in \mathbb{R}^{N\times N}$ jest macierzą stanu sieci (diag$(\mathbf{A}) = [\mathbf{A}_1, \ldots, \mathbf{A}_v]$), $\mathbf{W} = \left[\mathbf{w}_1\mathbf{1}^T, \ldots, \mathbf{w}_v\mathbf{1}^T\right]^T \in \mathbb{R}^{N\times n}$, gdzie $\mathbf{w}_i$ jest wektorem wag i - tego neuronu warstwy ukrytej, $\mathbf{C} \in \mathbb{R}^{m\times v}$ jest macierzą wyjścia, $\mathbf{B} \in \mathbb{R}^{v\times N}$ jest diagonalną macierzą parametrów filtru (diag$(\mathbf{B}) = [\mathbf{b}_1, \ldots, \mathbf{b}_v]$), $\mathbf{D} = \left[b_{0_1}\mathbf{w}_1^T, \ldots, b_{0_v}\mathbf{w}_v^T\right]^T \in \mathbb{R}^{v\times n}$ jest macierzą przejścia, $\mathbf{g}_1 = [\mathbf{g}_{1_1}, \ldots, \mathbf{g}_{1_v}]^T$ to wektor zawierający progi funkcji aktywacji, $\mathbf{G}_2 \in \mathbb{R}^{v\times v}$ jest diagonalną macierzą wartości współczynników nachylenia funkcji aktywacji neuronów (diag$(\mathbf{G}_2) = [\mathbf{g}_{2_1}, \ldots, \mathbf{g}_{2_v}]$), oraz $f : \mathbb{R}^v \longrightarrow \mathbb{R}^v$ jest nieliniowym operatorem transformacji.

Jak już wspominano wcześniej, struktura tego typu stosowana była z powodzeniem w systemach diagnostycznych w zadaniach detekcji, lokalizacji oraz identyfikacji uszkodzeń (Korbicz *i in.*, 2004; Patan, 2008b). Badania dotyczące rozwoju tego rodzaju sieci w połączeniu z zastosowaniem odpornej detekcji i lokalizacji uszkodzeń potwierdzają, że możliwe jest uzyskanie środka o dużej przydatności praktycznej w zakresie diagnozowania.

3.4. Projektowanie sieci rekurencyjnych

Podczas projektowania struktur rekurencyjnych niezbędne jest rozważenie kilku podstawowych zadań. Pierwszym etapem projektowania jest przyjęcie odpowiedniego dla danego problemu sposobu reprezentacji struktury neuropodobnej (w formie równań stanu lub zapisu wejściowo-wyjściowego). Podobnie, jak ma to miejsce w sieciach jednokierunkowych, niezbędne jest opracowanie skutecznego algorytmu uczenia sieci oraz sposobu doboru jej struktury. Dodatkowym problemem, który należy uwzględnić, projektując aplikacje bazujące na sieciach rekurencyjnych, jest odpowiednia analiza stabilności tych sieci.

3.4.1. Uczenie sieci rekurencyjnych

Podobnie jak dla sieci statycznych, głównym celem uczenia sieci rekurencyjnych jest wyznaczenie wektora parametrów sieci, dla którego określona funkcja celu (np. w postaci sumy kwadratów błędu) przyjmuje wartość minimum. Iteracyjne metody poszukiwania optymalnych wartości parametrów sieci najogólniej możemy podzielić na algorytmy globalne i lokalne. Bardziej szczegółowy podział pozwala na rozróżnienie algorytmów pod względem sposobu wyznaczenia nowego przybliżenia funkcji celu. Możemy tu wyróżnić metody zdeterminowane oraz metody stochastyczne.

Do najczęściej stosowanych metod globalnej optymalizacji parametrów sieci zaliczamy: modyfikacje metody Monte-Carlo, metody przeszukiwania systematycznego, metody adaptacyjnego przeszukiwania ślepego (Patan, 2008b), algorytmy ewolucyjne (Obuchowicz i Korbicz, 2004). Metody te charakteryzują się zazwyczaj wolną zbieżnością i wymagają dużej liczby wyliczeń funkcji celu.

Z drugiej strony, skuteczne przy optymalizacji lokalnej są metody gradientowe, które opierają się na liniowym lub kwadratowym rozwinięciu funkcji celu w pobliżu bieżącego rozwiązania. Do najpopularniejszych metod gradientowych można zaliczyć: modyfikacje metody największego spadku, metody quasi-newtonowskie oraz metody gradientów sprzężonych (Bajramovic *i in.*, 2004; Chang i Mak, 1999; Mandic i Chambers, 2001; Medsker i Jain, 1999; Patan, 2008b). Głównym problemem przy stosowaniu tego typu metod jest wyznaczenie gradientu, jakobianu i hesjanu funkcji błędu. Najczęściej używane do tego celu są dwie modyfikacje metody wstecznej propagacji błędu (De Jesus i Hagan, 2007): metoda BPTT (ang. *BackPropagation Through Time*) oraz metoda RTRL (ang. *Real Time Recurrent Learning*). Ponadto do ich wyznaczenia używane są metody grafów przepływowych lub metody obliczeń symbolicznych (Osowski i Cichocki, 1999). Jeżeli przyjęta postać struktury sieci zbytnio komplikuje wyliczenia analityczne gradientu (jakobianu, hesjanu), to możliwe jest zastosowanie metod numerycznej lub stochastycznej aproksymacji gradientu (jakobianu, hesjanu) (Patan i Parisini, 2002). W grupie algorytmów lokalnych stosowanych do strojenia parametrów sieci rekurencyjnych popularne są również: modyfikacje rekurencyjnej metody najmniejszych kwadratów, metody heurystyczne oraz metody z losowym wyborem kierunku (Duch *i in.*, 2000).

Bardzo często używa się metod mieszanych (Duch *i in.*, 2000; Obuchowicz i Korbicz, 2004), będących połączeniem poprzednio wymienionych algorytmów globalnych i lokalnych, co pozwala na osiągnięcie zakładanej dokładności rozwiązania przy zachowaniu stosunkowo dużej szybkości zbieżności oraz ograniczonej złożoność obliczeniowej. Należy jednak zaznaczyć, że wybór odpowiedniego sposobu trenowania sieci uzależniony jest od ograniczeń wynikających z rozpatrywanego problemu. Sposób trenowania oraz jakość dostępnych danych wpływają na niepewność parametryczną modelu neuronowego.

3.4.2. Dobór struktury w sieciach rekurencyjnych

W zadaniu identyfikacji obiektu przy użyciu modelu neuronowego, obok estymacji nieznanych parametrów tego modelu, niezbędne jest poprawne określenie struktury zastosowanej sieci neuronowej. Dobór architektury sieci neuronowej jest jednym z czynników wpływających na zdolność uogólniania (generalizacji) tej sieci. Wybór typu modelu oraz jego architektury ma więc zasadniczy wpływ na niepewność strukturalną modelu.

Dobór topologii sieci polega na: odpowiednim zdefiniowaniu liczby warstw ukrytych, określeniu liczby jednostek w warstwach, ustanowieniu lub pominięciu określonych połączeń synaptycznych, określeniu postaci funkcji wyjściowej neuronów oraz określeniu struktury systemów dynamicznych zagnieżdżonych w neuronach dynamicznych (dla sieci neuronów dynamicznych). Zadaniem równoległym podczas budowania modelu neurono-

wego jest określenie wejść i wyjść istotnych dla rozważanego problemu, przy czym ten etap budowy modelu zazwyczaj realizowany jest oddzielnie.

Wyróżnia się następujące klasy algorytmów optymalizujących architekturę sieci neuronowej (Duch *i in.*, 2000; Jankowski, 2003):

- metody wzrostu (np. algorytm kafelkowy, algorytm dynamicznej kreacji neuronów, algorytm kaskadowej korelacji),
- metody redukcji (tj. np. metody badania wrażliwości, metody funkcji kary, metody analizy statystycznej),
- metody optymalizacji dyskretnej (np. algorytmy ewolucyjne, algorytm A*, algorytmy przeszukiwania systematycznego).

Jak się okazuje, nie istnieje żaden sposób umożliwiający określenie, które z wymienionych podejść pozwoli osiągnąć najlepsze rezultaty w przypadku rozpatrywanego problemu. Jedynie dla metod wzrostu struktury wykazano, że zbieżność do rozwiązania jest gwarantowana (Duch *i in.*, 2000). Metody te jednak wymagają zbyt dużego nakładu obliczeń wówczas, gdy rozważany problem wymaga do jego opisu struktury o dużej złożoności. Algorytmy bazujące na metodach redukcji wymagają określenia początkowej postaci struktury sieci, co nie jest zadaniem łatwym. Jest to zazwyczaj realizowane z zastosowaniem prostych reguł heurystycznych (Duch *i in.*, 2000). W trzeciej grupie algorytmów zadanie optymalizacji sprowadza się do przeszukania przestrzeni dyskretnej architektur sieci, co zazwyczaj prowadzi do problemu NP-trudnego. W opinii autora najbardziej skutecznym sposobem optymalizacji topologii sieci mogą być metody będące połączeniem kilku metod należących do omówionych klas.

3.4.3. Stabilność i stabilizacja sieci rekurencyjnych

Bardzo ważnym aspektem projektowania sieci zarówno lokalnie jak i globalnie rekurencyjnych jest rozważenie problemu ich stabilności. Rozważania te dotyczą zarówno struktury modelu, jak i procesu uczenia. Przeważająca liczba prac dotyczy analizy stabilności wytrenowanych modeli neuronowych (Lisheng Wang i Zongben Xu, 2006; Mandic i Chambers, 2001; Weimin Shen *i in.*, 2004; Wilson, 1995). Opierają się one głównie na pierwszej i drugiej metodzie Lapunowa. Znane są również prace, w których podejmowany jest problem stabilizacji modeli neuronowych podczas procesu uczenia (Drapała *i in.*, 2008; Patan, 2007; Patan, 2008b). W przypadku sieci lokalnie rekurencyjnych problem analizy stabilności sprowadza się do analizy stabilności pojedynczych neuronów w sieci (Gupta *i in.*, 2003; Patan, 2008b).

3.5. Podsumowanie

Przedmiotem rozważań rozdziału były wybrane struktury globalnie i lokalnie rekurencyjne. Omówiono takie topologie globalnie rekurencyjne jak: wielokontekstowe sieci Jordana i Elmana, sieci NNARX, sieci globalnie rekurencyjne w postaci równań stanu. Wy-

różniono wybrane dynamiczne modele neuronu, które autor uznał za jednostki wnoszące wielki wkład w rozwój struktur lokalnie rekurencyjnych globalnie jednokierunkowych. Szczególną rolę w tej grupie stanowią modele: neuronu z filtrem IIR, neuronu G-FGS oraz neuronu chaotycznego. Ponadto skrótowo omówiono trzy ważne problemy dotyczące projektowania struktur sieci tego typu: uczenie, analizę stabilności i stabilizację struktur rekurencyjnych.

Dokonany przegląd dostępnych autorowi publikacji dotyczących różnego rodzaju metodyk modelowania neuronowego w metodologii diagnostyki procesów pozwala na stwierdzenie, że zastosowanie struktury sieci lokalnie rekurencyjnej wyposażonej w nowe modele neuronu może być sposobem na uzyskanie środka do generacji residuów wrażliwych na pojawiające się w obiekcie uszkodzenia, przy jednoczesnej minimalizacji wpływu niepożądanych czynników (tj. zakłócenia i szumy pomiarowe). Wprowadzenie jednostki przetwarzającej nowego typu do tego rodzaju sieci może umożliwić również ograniczenie w pewnym stopniu niepewności strukturalnej.

Rozdział 4

Metodyka modelowania neuronowego w diagnostyce procesów

W poniższym rozdziale przedstawiono sposób tworzenia modeli neuronowych (o strukturach lokalnie rekurencyjnych) oraz ich zastosowania do diagnozowania różnego rodzaju procesów. Najważniejszymi ogniwami metodyki są:

- sposób reprezentacji danych,
- opis formalny zaprojektowanych struktur lokalnie rekurencyjnych,
- sposób trenowania zaproponowanych struktur sieci,
- sposób oceny uzyskiwanych modeli procesów,
- sposób analizy stabilności asymptotycznej modelu,
- sposób doboru struktury modelu,
- metoda detekcji uszkodzeń oparta o zaproponowaną strukturę sieci,
- sposób oceny sprawności systemu diagnostycznego.

Metodyka uwzględnia wybrane elementy teorii chaosu deterministycznego, które stosowane są podczas budowy modelu neuronowego procesu oraz podczas budowy części decyzyjnej systemu detekcji uszkodzeń.

4.1. Koncepcja metodyki

W proponowanej metodyce można wyróżnić dwie główne fazy: faza budowy modelu procesu oraz faza projektowania bloku decyzyjnego systemu detekcji uszkodzeń. Proponowany sposób budowy modelu neuronowego zbieżny jest z ogólną teorią identyfikacji systemów. Również zasada działania systemu detekcji uszkodzeń zgodna jest z ogólną koncepcją diagnozowania na podstawie modelu. Nowym elementem w proponowanym podejściu, w stosunku do istniejących rozwiązań, jest zastosowanie elementów teorii chaosu deterministycznego w obu etapach. Zastosowanie teorii chaosu dotyczy kwestii:

- wykorzystania podstawowych własności układów chaotycznych do uzyskania modeli neuronowych procesów wrażliwych na uszkodzenia,

- wykorzystania narzędzi nieliniowej analizy danych (opracowanych dla celów badania układów chaotycznych) w zadaniu detekcji uszkodzeń.

Brano pod uwagę następujące przesłanki. Wprowadzenie nowych elementów przetwarzających (do struktur lokalnie rekurencyjnych), które posiadają własności charakterystyczne dla układów chaotycznych, może przyczynić się do uzyskania struktur neuronowych wrażliwych na pojawiające się uszkodzenia (szczególnie o niewielkim rozmiarze). Takie stwierdzenie jest uzasadnione, ponieważ:

- chaotyczne układy dynamiczne cechuje duża wrażliwość na dowolnie małe zaburzenie parametrów (układu lub otoczenia),
- układy dynamiczne tego typu w skali czasu mikro zachowują się deterministycznie, natomiast w skali czasu makro zachowanie układów chaotycznych jest nieprzewidywalne.

Z drugiej strony, zastosowanie do oceny residuów (generowanych za pomocą zaproponowanych modeli neuronowych) metody analizy ilościowej diagramów rekurencyjnych (RQA), która opracowana została w celu diagnozowania układów chaotycznych uważanych za najtrudniejsze z układów deterministycznych do analizowania, powinno umożliwić budowę odpornych systemów detekcji uszkodzeń.

4.2. Zastosowana sieć neuronów dynamicznych

W dalszej części pracy rozpatrywana będzie sieć neuronów dynamicznych złożona z neuronów, których zasada działania została opisana przez autora w pracy (Przystałka, 2008), jako wynik wcześniejszych rozważań zaprezentowanych w (Przystałka, 2007a; Przystałka, 2007b).

4.2.1. Uogólniony model neuronu dynamicznego

Zaproponowany przez autora model neuronu pokazany jest na Rys. 4.1. Można go uważać za uogólnienie dwóch modeli: neuronu z filtrem o nieskończonej odpowiedzi impulsowej w bloku aktywacji (Ayoubi, 1994; Korbicz *i in.*, 2004; Patan, 2008b) oraz modelu neuronu ze sprzężeniem wyjściowym (Frasconi *i in.*, 1992). Strukturę takiej jednostki przetwarzającej uzyskano przez wbudowanie liniowych systemów dynamicznych typu ARMAX w blok aktywacji oraz w blok wyjściowego sprzężenia zwrotnego. Działanie to pozwoliło otrzymać element przetwarzający, na który poza sygnałami wymuszenia wpływ mają również zakłócenia. Rozpatrywany neuron umożliwia modelowanie procesów technicznych zdeterminowanych, na które oddziałują zakłócenia stacjonarne.

Formalny opis rozpatrywanego modelu jest następujący. Stan w złączu sumującym oblicza się z równania

$$\xi_1(k) = \sum_{i=1}^{r_n} w_i u_i(k) + w_{(r_n+1)} \xi_3(k), \tag{4.1}$$

przy czym wewnętrzny stan neuronu w bloku aktywacyjnym może zostać sformułowany jako:

$$A\left(q^{-1}\right)\xi_2\left(k\right) = B\left(q^{-1}\right)\xi_1\left(k\right) + C\left(q^{-1}\right)\phi_A\left(k\right), \tag{4.2}$$

oraz łączne sprzężone pobudzenie neuronu jako:

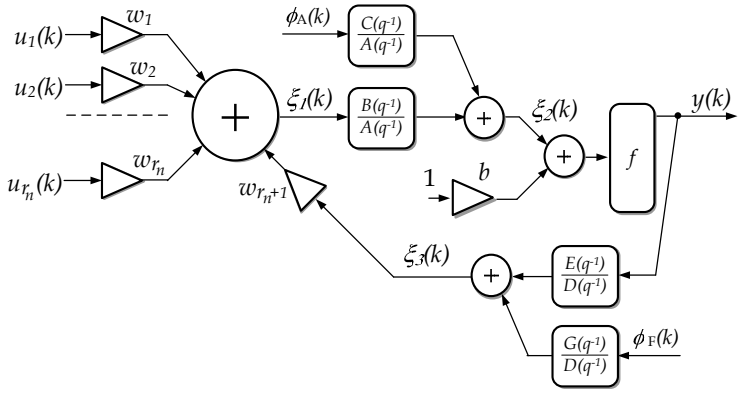
$$D\left(q^{-1}\right)\xi_3\left(k\right) = E\left(q^{-1}\right)y\left(k\right) + G\left(q^{-1}\right)\phi_F\left(k\right). \tag{4.3}$$

I ostatecznie, wyjście jednostki w dyskretnej chwili k otrzymuje się z równania:

$$y\left(k\right) = f\left(\xi_2\left(k\right) + b\right), \tag{4.4}$$

gdzie r_n - liczba wejść zewnętrznych neuronu, $u_i(k)$ oznacza i-te wejście neuronu w dyskretnej chwili k, w_i oznacza i-tą wagę neuronu, $\xi_i\left(k\right)$ reprezentuje stan wewnętrzny jednostki, ϕ_A, ϕ_F reprezentują procesy losowe o założonym rozkładzie, $f(\cdot)$ oznacza funkcję wyjścia neuronu, b jest progiem jednostki.

W zdecydowanej przewadze przypadków w pracy domyślnie przyjmuje się, że czynnik ϕ ma charakter procesu stochastycznego o rozkładzie normalnym $\mathcal{N}(0, \sigma_\phi^2 = 1)$. Jeżeli sytuacja będzie wymagała innych założeń, co do charakteru zakłócenia, wtedy zostanie to wyraźnie podane.



Rys. 4.1: Schemat ideowy modelu neuronu z liniowymi systemami dynamicznymi w bloku aktywacyjnym i w bloku sprzężenia wyjściowego

Wielomiany zmiennej q^{-1} zdefiniowane są następująco:

$$\begin{aligned}
A\left(q^{-1}\right) &= 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na}, \\
B\left(q^{-1}\right) &= b_0 + b_1 q^{-1} + \ldots + b_{nb} q^{-nb}, \\
C\left(q^{-1}\right) &= c_0 + c_1 q^{-1} + \ldots + c_{nc} q^{-nc}, \\
D\left(q^{-1}\right) &= 1 + d_1 q^{-1} + \ldots + d_{nd} q^{-nd}, \\
E\left(q^{-1}\right) &= e_0 + e_1 q^{-1} + \ldots + e_{ne} q^{-ne}, \\
G\left(q^{-1}\right) &= g_0 + g_1 q^{-1} + \ldots + g_{ng} q^{-ng}.
\end{aligned} \tag{4.5}$$

W pracy brano pod uwagę następujące typy funkcji wyjścia neuronów:

- funkcja liniowa:

$$y = x, \tag{4.6}$$

- funkcja logistyczna:

$$y = \frac{1}{1 + e^{-\alpha_f x}}, \tag{4.7}$$

- funkcja w postaci tangensa hiperbolicznego:

$$y = \tanh(x/\alpha_f) = \frac{1 - e^{-x/\alpha_f}}{1 + e^{-x/\alpha_f}}, \tag{4.8}$$

- funkcja o kształcie sigmoidalnym:

$$y = \frac{\alpha_f x}{\sqrt{1 + \alpha_f^2 x^2}}, \tag{4.9}$$

gdzie α_f to parametr określający nachylenie funkcji wyjścia neuronu. Funkcja (4.6) stosowana jest głównie w ostatniej warstwie sieci, której zadaniem jest denormalizacja danych.

Wybór funkcji (4.7)-(4.9) sprawia, że współczynnik zbieżności estymacji nieznanych parametrów modelu neuronowego nie zależy od rozmiaru przestrzeni wejściowej modelowanego problemu (Jankowski, 2003). Funkcja (4.9) pozwala z kolei na szybsze wyliczenie wartości wyjścia neuronu niż w przypadku klasycznych funkcji sigmoidalnych takich jak (4.7)-(4.8).

Wektorem parametrów neuronu jest:

$$\begin{array}{cccccccccccccc} & & w_1 & w_2 & & b & a_1 & a_2 & & b_0 & b_1 & & \alpha_f & \sigma_\phi^2 \\ & & \shortparallel & \shortparallel & & \shortparallel & \shortparallel & \shortparallel & & \shortparallel & \shortparallel & & \shortparallel & \shortparallel \\ \boldsymbol{\omega} & = & [\ \omega_1 & \omega_2 & \dots & \omega_j & \omega_k & \omega_{k+1} & \dots & \omega_l & \omega_{l+1} & \dots & \omega_{m-1} & \omega_m\ ], \end{array} \tag{4.10}$$

gdzie liczba parametrów modelu neuronu $m = r_n + na + nb + nc + nd + ne + ng + 8$, przy czym ostatni czynnik zależy od struktury zagnieżdżonych systemów dynamicznych.

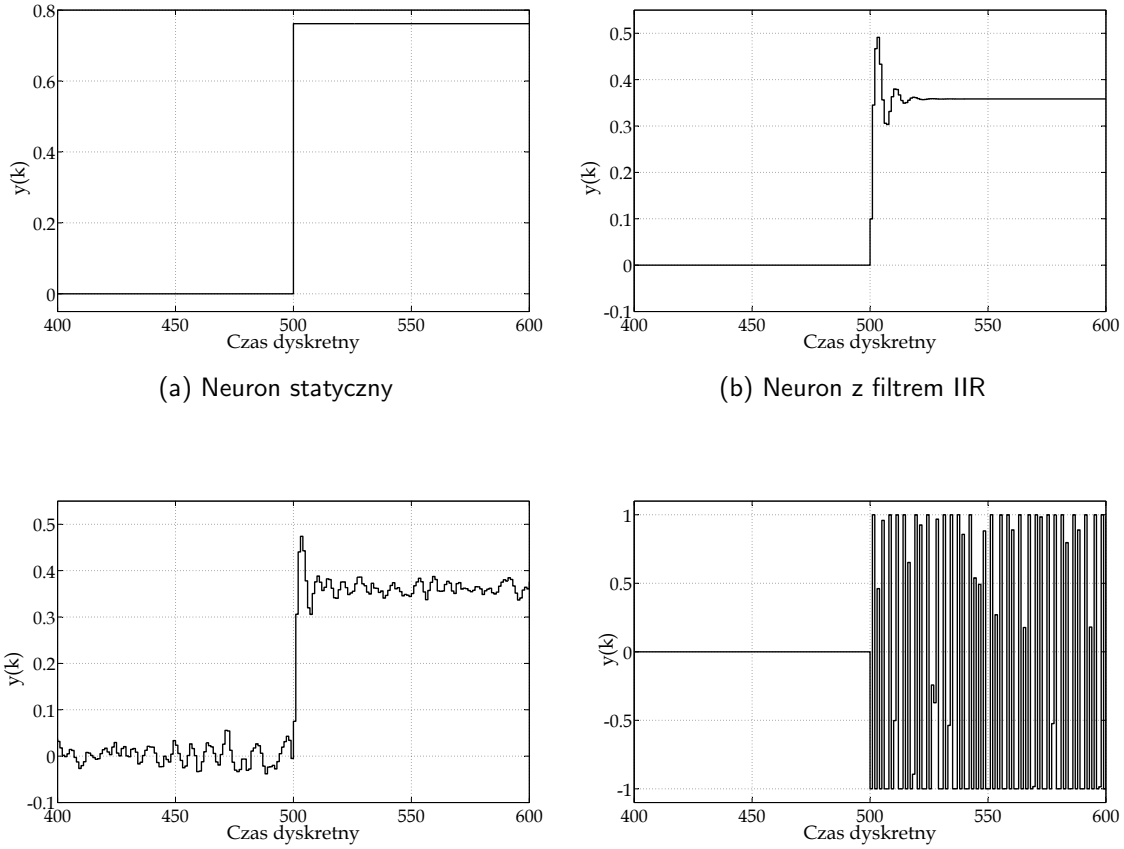
4.2.2. Właściwości neuronu

Model rozpatrywanego neuronu otrzymano, sprowadzając ogólny model matematyczny systemu typu SISO (Söderström i Stoica, 1997) do postaci struktury typu ARMAX i następnie zagnieżdżając go w blokach aktywacji i sprzężenia zwrotnego neuronu. W ramach struktury ARMAX można opisać dowolny liniowy system skończonego rzędu o stacjonarnych zakłóceniach (Box i Jenkins, 1983).

Pouczające wydaje się rozpatrzenie, w jaki sposób związane są ze sobą poszczególne warianty uogólnionego modelu neuronu. W Tab. 4.1 zamieszczono szczególne przypadki jednostki przetwarzającej o strukturze typu SISO ($r_n = 1$, $w_1 = 1$, $b = 0$) z funkcją wyjścia w postaci (4.8). Na Rys. 4.2 (a)-(d) zamieszczono odpowiadające im charakterystyki skokowe wygenerowane dla zerowych warunków początkowych.

Tab. 4.1: Warianty jednostki przetwarzającej

Nr	Oznaczenie	Założenia dotyczące struktury	Wartości parametrów
1	$\begin{smallmatrix}1\\0\end{smallmatrix}$	$A\left(q^{-1}\right)=B\left(q^{-1}\right)=1,\ C\left(q^{-1}\right)=0,$ $D\left(q^{-1}\right)=E\left(q^{-1}\right)=G\left(q^{-1}\right)=0$	$\alpha_f=1$
2	$\begin{smallmatrix}(3,2,0)\\0\end{smallmatrix}$	$C\left(q^{-1}\right)=0,$ $D\left(q^{-1}\right)=E\left(q^{-1}\right)=G\left(q^{-1}\right)=0$	$\alpha_f=1,\ b_0=0.1,\ b_1=0.2,$ $a_1=-0.6,\ a_2=0.1,\ a_3=0.3$
3	$\begin{smallmatrix}(3,2,0)\\(0,0,1)\end{smallmatrix}$	$C\left(q^{-1}\right)=0,$ $D\left(q^{-1}\right)=1,\ E\left(q^{-1}\right)=0$	$\alpha_f=1,\ b_0=0.1,\ b_1=0.2,$ $a_1=-0.6,\ a_2=0.1,\ a_3=0.3,\ g_0=0.1$
4	$\begin{smallmatrix}(1,1,0)\\(0,1,0)\end{smallmatrix}$	$C\left(q^{-1}\right)=0,$ $D\left(q^{-1}\right)=1,\ G\left(q^{-1}\right)=0$	$\alpha_f=0.5,\ b_0=-1,\ w_2=5$ $a_1=-0.95,\ e_0=0.8$



(a) Neuron statyczny

(b) Neuron z filtrem IIR

(c) Neuron z filtrem IIR i procesem resztowym

(d) Neuron chaotyczny

Rys. 4.2: Charakterystyki skokowe wybranych wariantów neuronu

Przyjmuje się, że oznaczenie struktur systemów dynamicznych zawartych w bloku aktywacyjnym i w bloku sprzężenia zwrotnego jest następujące: $\begin{smallmatrix}(na,nb,nc)\\(nd,ne,ng)\end{smallmatrix}$.

Neuron statyczny

Pierwszym szczególnym przypadkiem neuronu uogólnionego jest jednostka nr 1, w której dynamika nie występuje. Pobudzenie neuronu nie zależy w tym przypadku od jego stanów

z poprzednich chwil czasu. Odpowiedź tego typu neuronu na wymuszenie w postaci skoku jednostkowego pokazano na Rys. 4.2 (a).

Neuron z filtrem IIR

Innym przypadkiem postaci neuronu, znanego z literatury, może być wariant modelu neuronu nr 2. Jest to neuron z filtrem o nieskończonej odpowiedzi impulsowej w bloku aktywacyjnym (Ayoubi, 1994; Korbicz *i in.*, 2004). Jak można zauważyć, wyjście neuronu w tym wariancie zależy również od jego poprzednich stanów. Dodatkowo filtr pełni rolę kompensatora zakłóceń będących składowymi sygnałów wejściowych. Charakterystykę skokową neuronu z filtrem IIR dla przykładowych wartości parametrów pokazano na Rys. 4.2 (b).

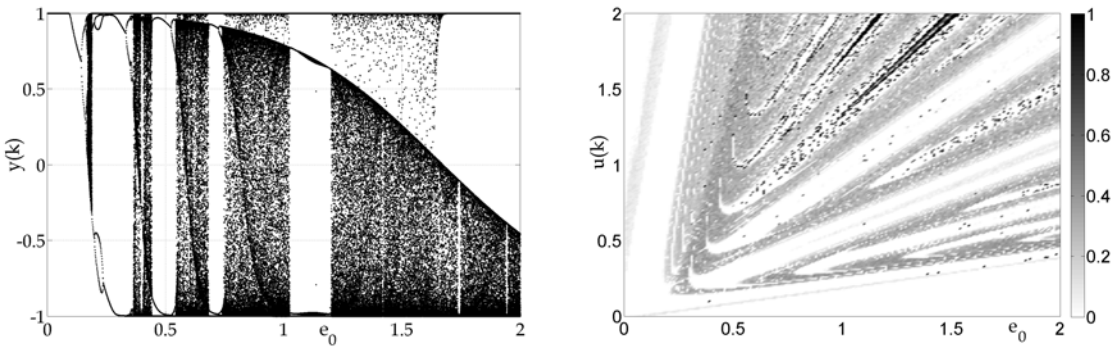
Neuron ze sprzężeniem wyjściowym

Bardziej złożoną jednostkę otrzymuje się, aktywując wpływ wyjściowego sprzężenia zwrotnego, które w tym przypadku można uznać za sygnał związany z procesem resztkowym (jednostka nr 3). Umożliwia to np. modelowanie zjawisk, na które mają wpływ zakłócenia będące składowymi pochodzącymi od niepożądanych zachowań obiektu, takich jak drgania maszyny lub hałas. Ponadto czynnik ϕ może nieść informacje o nieznanych lub niemierzalnych składowych wejść i wyjść identyfikowanego obiektu (np. nieobserwowalne skutki działania maszyny lub pomijalne warunki otoczenia). Odpowiedź skokową tego typu jednostki przedstawiono na Rys. 4.2 (c).

Neuron chaotyczny

Ciekawą własnością dyskutowanego neuronu jest to, że przy odpowiednich założeniach (przypadek nr 4) możliwe staje się uzyskanie własności charakterystycznych dla skrajnego przypadku układów niestabilnych (tzw. układów chaotycznych). Mówi się wtedy o tak zwanym chaosie deterministycznym. Równania opisujące model neuronu są wówczas równie deterministyczne, jak w przypadku modelu neuronu wg McCullocha-Pittsa, a mimo to mogą być stosowane do opisu zachowań o charakterze losowym. Rys. 4.2 (d) pokazuje przykładową odpowiedź neuronu o dynamice chaotycznej na sygnał wymuszenia w postaci skoku jednostkowego. Dodatkowo na Rys. 4.3 (a) i (b) rozpatrzono zachowanie neuronu z zastosowaniem diagramu bifurkacyjnego oraz analizy największego wykładnika Lapunowa. Pierwszy z wykresów uzyskano, zmieniając wartość parametru refrakcji e_0 dla stałego wymuszenia $u(k)$. Parametr refrakcji określa wpływ poprzednich stanów wyjścia neuronu na bieżący stan wewnętrzny. Drugi rysunek przedstawia wykres konturowy (wykres powierzchniowy widziany z góry) obrazujący zmiany największego wykładnika Lapunowa w funkcji parametru refrakcji oraz sygnału pobudzenia.

Oba rysunki potwierdzają złożony charakter dynamiki dla prostego przypadku struktury neuronu uogólnionego. Uwidacznia się wiele obszarów, dla których wykładnik Lapunowa przyjmuje wartości dodatnie, co potwierdza wrażliwość zachowania neuronu ze względu na warunki początkowe.



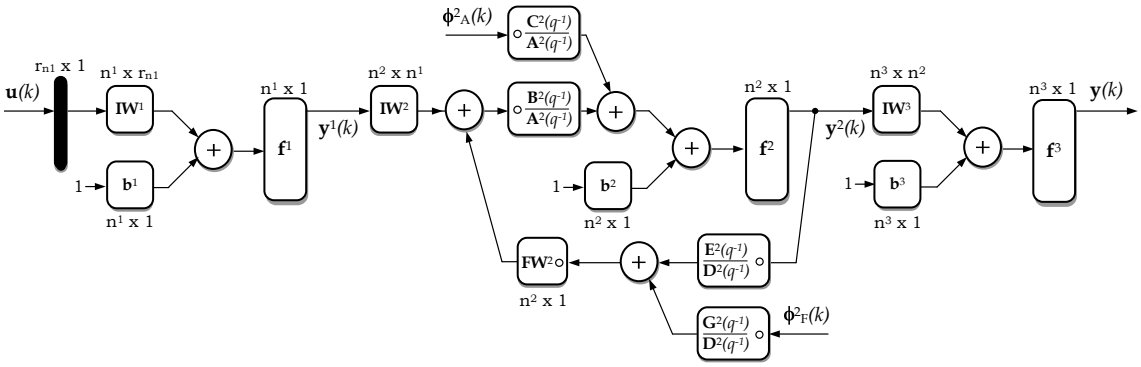
(a) Diagram bifurkacyjny dla $u(k) = 0.5$ (b) Największy wykładnik Lapunowa

Rys. 4.3: Analiza dynamiki neuronu uogólnionego

Przeprowadzona analiza potwierdza, że zmiany charakteru dynamiki neuronu zależą nie tylko od wartości jego parametrów, ale również od czynników zewnętrznych. W tym przypadku czynnikiem tym była amplituda sygnału pobudzenia. Podobne wyniki można uzyskać, przyjmując np. okresowy sygnał wymuszenia o stałej amplitudzie, gdzie czynnikiem wpływającym na charakter dynamiki będzie częstotliwość tego sygnału.

4.2.3. Sieć lokalnie rekurencyjna

Strukturę tego rodzaju przedstawia Rys. 4.4. Sieć neuronów dynamicznych złożona jest z trzech lub dwóch warstw. Pierwsza warstwa nazywana jest pierwszą warstwą ukrytą i pełni rolę bloku, w którym następuje wstępne przetwarzanie danych.



Rys. 4.4: Schemat blokowy lokalnie rekurencyjnej globalnie jednokierunkowej sieci neuronowej

Neurony w warstwie wejściowej mogą być zarówno elementami statycznymi jak i dynamicznymi z liniową lub nieliniową funkcją wyjścia. Zazwyczaj są to neurony z dynamiką wyłącznie w bloku aktywacji i nieliniową funkcją wyjścia (co umożliwia standaryzację i filtrowanie danych wejściowych).

Druga warstwa ukryta ma za zadanie odwzorowywanie pełnej dynamiki modelowanego obiektu. Neurony tej warstwy zazwyczaj posiadają nieliniową funkcję wyjściową

przy jednoczesnym stosowaniu systemów dynamicznych w blokach aktywacji i sprzężenia zwrotnego. Warstwa wyjściowa złożona jest ze statycznych jednostek z liniową funkcją wyjścia. Warstwa ta pełni rolę bloku, w którym następuje denormalizacja.

Poniżej wyprowadzono zależności macierzowe opisujące przetwarzanie neuronowe sieci neuronów dynamicznych. Przyjmując macierz wag wejść zewnętrznych neuronów i-tej warstwy w postaci:

$$\mathbf{IW}^i = \begin{bmatrix} w_{11}^i & w_{12}^i & \cdots & w_{1r_{n^i}}^i \\ w_{21}^i & w_{22}^i & \cdots & w_{2r_{n^i}}^i \\ \vdots & \vdots & \ddots & \vdots \\ w_{n^i1}^i & w_{n^i2}^i & \cdots & w_{n^ir_{n^i}}^i \end{bmatrix}, \tag{4.11}$$

oraz wektor wag wejść sprzężonych jako:

$$\mathbf{FW}^i = \begin{bmatrix} w_{1(r_{n^i}+1)}^i & w_{2(r_{n^i}+1)}^i & \cdots & w_{n^i(r_{n^i}+1)}^i \end{bmatrix}^T, \tag{4.12}$$

łączne wejściowe pobudzenie neuronów i-tej warstwy można wyrazić poprzez wektor:

$$\boldsymbol{\xi}_1^i(k) = \mathbf{IW}^i\mathbf{x}^i(k) + \mathbf{FW}^i \circ \boldsymbol{\xi}_3^i(k), \tag{4.13}$$

gdzie $\circ$ oznacza operator mnożenia tablicowego, zaś n^i to liczba neuronów w i-tej warstwie. Dla pierwszej warstwy przyjmuje się

$$\mathbf{x}^1(k) = \mathbf{u}(k) = \begin{bmatrix} u_1(k) & u_2(k) & \cdots & u_{r_{n^1}}(k) \end{bmatrix}^T, \tag{4.14}$$

a dla pozostałych warstw $(i > 1)$:

$$\mathbf{x}^i(k) = \mathbf{y}^{i-1}(k) = \begin{bmatrix} y_1^{i-1}(k) & y_2^{i-1}(k) & \cdots & y_{n^{i-1}}^{i-1}(k) \end{bmatrix}^T. \tag{4.15}$$

Składową reprezentującą sygnały wejść sprzężonych tej warstwy wyraża wektor:

$$\boldsymbol{\xi}_3^i(k) = \begin{bmatrix} \xi_{31}^i(k) & \xi_{32}^i(k) & \cdots & \xi_{3n^i}^i(k) \end{bmatrix}^T. \tag{4.16}$$

Stany wewnętrzne neuronów w bloku aktywacji i-tej warstwy zapisuje się stosując notację wielomianową w postaci wektorowej:

$$\mathbf{A}^i(q^{-1}) \circ \boldsymbol{\xi}_2^i(k) = \mathbf{B}^i(q^{-1}) \circ \boldsymbol{\xi}_1^i(k) + \mathbf{C}^i(q^{-1}) \circ \boldsymbol{\phi}_{\mathrm{A}}^i(k), \tag{4.17}$$

i podobnie dla bloku sprzężenia wyjściowego:

$$\mathbf{D}^i(q^{-1}) \circ \boldsymbol{\xi}_3^i(k) = \mathbf{E}^i(q^{-1}) \circ \mathbf{y}^i(k) + \mathbf{G}^i(q^{-1}) \circ \boldsymbol{\phi}_{\mathrm{F}}^i(k), \tag{4.18}$$

gdzie wielomiany można zapisać w postaci wektorowej:

$$\mathbf{A}^i(q^{-1}) = \begin{bmatrix} A_1^i(q^{-1}) & A_2^i(q^{-1}) & \cdots & A_{n^i}^i(q^{-1}) \end{bmatrix}^T, \tag{4.19}$$

przy czym wielomian A n-tego neuronu i-tej warstwy zapisuje się jako:

$$A_n^i(q^{-1}) = 1 + a_{n1}^i q^{-1} + a_{n2}^i q^{-2} + \cdots + a_{nna^i}^i q^{-na^i}. \tag{4.20}$$

Dla pozostałych wielomianów zmiennej q^{-1} postępuje się w analogiczny sposób.

Ostatecznie wyjście i-tej warstwy wyznaczane jest z zastosowaniem liniowego lub nieliniowego operatora transformacji:

$$\mathbf{y}^i(k) = \mathbf{f}^i\left(\boldsymbol{\xi}_2^i(k) + \mathbf{b}^i\right). \tag{4.21}$$

Wektor reprezentujący wszystkie parametry swobodne s - warstwowej sieci lokalnie rekurencyjnej wyrażony jest jako

$$\boldsymbol{\omega} = \begin{bmatrix} \boldsymbol{\omega}^1 & \boldsymbol{\omega}^2 & \cdots & \boldsymbol{\omega}^s \end{bmatrix}^T. \tag{4.22}$$

przy czym $p = \dim(\boldsymbol{\omega})$ jest liczbą parametrów swobodnych sieci. Zapis reprezentujący $s-$ warstwowy model neuronowy o r_{n^1} wejściach i n^s wyjściach jest następujący:

$$r_{n^1} \longrightarrow \ldots \longrightarrow n^{i\,(na^i, nb^i, nc^i)}_{(nd^i, ne^i, ng^i)} \longrightarrow \ldots \longrightarrow n^s. \tag{4.23}$$

gdzie na^i, nb^i, ... określają postaci struktur systemów dynamicznych zawartych w blokach aktywacji i wyjściowego sprzężenia zwrotnego i-tej warstwy, n^i określa liczbę neuronów w i-tej warstwie.

4.3. Hybrydowa metoda uczenia

Możliwość uczenia (trenowania) i adaptacji do zmieniających się warunków to podstawowa zaleta sieci neuronowych. Ich funkcjonalność w głównej mierze zależy od algorytmów uczenia, które umożliwią dostrajanie parametrów swobodnych sieci odpowiednio do rozwiązywanego problemu. Obszerny przegląd różnorodnych metod pozwalających na uczenie szerokiej gamy struktur neuronowych można znaleźć m.in. w (Korbicz *i in.*, 1994; Osowski, 1996; Rutkowska *i in.*, 1997; Tadeusiewicz, 1993).

Opisywane w literaturze metody strojenia parametrów swobodnych sieci posiadają zasadnicze ograniczenia i wady, które uwidaczniają się wraz ze zwiększeniem się złożoności sieci lub liczby wzorców uczących. Metody optymalizacji wymagające znajomości gradientu funkcji celu mogą zatrzymać się w optimum lokalnym. W sieciach rekurencyjnych dodatkowym problemem jest zapewnienie stabilności modelu w trakcie strojenia jego parametrów. Wymaga to zazwyczaj użycia reguł umożliwiających stabilizację modelu w trakcie uczenia (Patan, 2008b). Te same problemy dotyczą algorytmów z losowym wyborem kierunku poszukiwań, choć tutaj nie jest wymagane, aby funkcja celu była różniczkowalna. Z drugiej strony, metody optymalizacji globalnej nie nakładają na funkcję celu żadnych ograniczeń. Jednak nawet przeszukiwanie zrealizowane za pomocą algorytmu ewolucyjnego jest zbyt czasochłonne dla sieci o rozbudowanej strukturze.

Wymienione powyżej problemy były bezpośrednią przyczyną tego, że w niniejszej pracy do strojenia parametrów swobodnych struktur lokalnie rekurencyjnych zastosowano hybrydową metodę optymalizacji realizowaną według tzw. strategii dwufazowej. Wynikało to również z faktu, iż metody uczenia hybrydowego były z powodzeniem stosowane zarówno do strojenia parametrów struktur jednokierunkowych (Duch *i in.*, 2000; Prudêncio i Ludermir, 2001; Rutkowska *i in.*, 1997), jak również rekurencyjnych (Li i Cheng, 2007; Obuchowicz i Korbicz, 2004).

Głównym celem uczenia hybrydowego jest wyznaczenie wektora parametrów sieci, dla których zdefiniowana funkcja celu przyjmuje wartość minimum zgodnie z ogólną zależnością:

$$\boldsymbol{\omega}^* = \arg\min_{\omega\in\Omega} E\left(\boldsymbol{\mathcal{A}}, \boldsymbol{\omega}\right), \tag{4.24}$$

gdzie $\Omega \subseteq \mathbb{R}^p$ jest zbiorem ograniczeń definiujących zakres wartości parametrów sieci neuronowej, $\boldsymbol{\omega}^*$ jest wektorem optymalnych parametrów sieci o wymiarze $p \times 1$, $\boldsymbol{\mathcal{A}}$ reprezentuje strukturę sieci. W niniejszej pracy przyjęto ograniczenie, że struktura sieci nie zmienia się w trakcie procesu uczenia.

Jak już wspomniano, głównym celem uczenia jest wyznaczenie wektora parametrów sieci $\boldsymbol{\omega}$, dla których funkcja celu przyjmuje wartość minimum. W ramach procesu uczenia autor zastosował funkcję celu w postaci:

$$E\left(\boldsymbol{\omega}, k\right) = \sum_{i=q}^{Q}\sum_{j=1}^{n^s}\left|f_j^s\left(\boldsymbol{u}_i, \boldsymbol{\omega}\right) - y_{ij}\right|^R + \lambda\sum_{r=1}^{p}\frac{\omega_r^2/\omega_0^2}{1+\omega_r^2/\omega_0^2}, \tag{4.25}$$

gdzie $f_j^s\left(\cdot\right)$ jest j-tym wyjściem sieci uzyskanym dla i-tego wzorca wejściowego $\mathbf{u}_i$, y_{ij} jest pożądanym j-tym wyjściem sieci, p oznacza liczbę parametrów swobodnych sieci, ω_r jest r-tym parametrem sieci, λ i ω_0 określają moc regularyzacji. Pierwszy składnik sumy w równaniu (4.25) jest błędem Minkowskiego, natomiast drugi to czynnik regularyzacyjny. Przyjęta postać funkcji celu umożliwia prowadzenie procesu uczenia w trybie:

- *online*, gdzie w n-tym kroku algorytmu parametry aktualizowane są po prezentacji bieżącego wzorca: $k = \{1, 2, \ldots, \mathrm{card}(\mathcal{L}_T)\}$, $q = Q = k = n$;
- *quasi-online*, gdzie w n-tym kroku algorytmu parametry aktualizowane są po prezentacji wzorców określonych za pomocą przesuwnego okna Δq: $k = \{\Delta q + 1, \Delta q + 2, \ldots, \ \mathrm{card}(\mathcal{L}_T)\}$, $q = k - \Delta q$, $Q = k$;
- *offline*, gdzie w n-tym kroku algorytmu parametry aktualizowane są po prezentacji wszystkich wzorców: $q = 1$, $Q = \mathrm{card}(\mathcal{L}_T)$.

Funkcję celu w pełnej formie ($\lambda \neq 0$) stosowano jedynie w pierwszej fazie uczenia z zastosowaniem algorytmu globalnego. W drugiej fazie uczenia dla algorytmów lokalnych bazujących na metodach gradientowych przyjmowano $R = 2$ i $\lambda = 0$.

Ogólny schemat uczenia hybrydowego wg strategii dwufazowej został przedstawiony na Rys. 4.5 (a). Pokazane tam podejście polega na optymalizacji parametrów sieci w następujący sposób. Najpierw przeprowadza się wstępną optymalizację za pomocą algo-

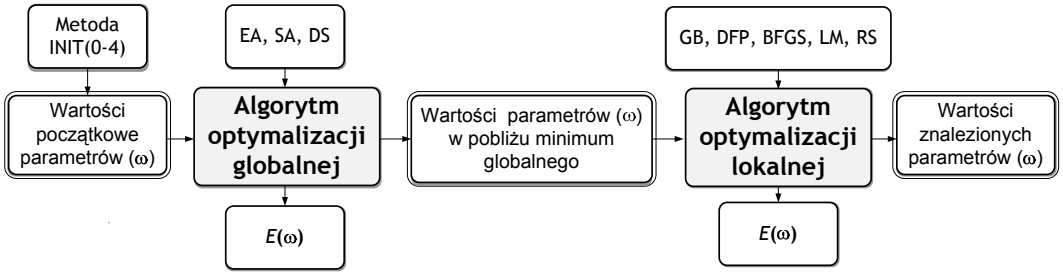
rytmu globalnego (np. metodą przeszukiwania bezpośredniego) w celu znalezienia początkowego zestawu parametrów sieci dla algorytmu lokalnego (np. metody najszybszego spadku). Następnie algorytm optymalizacji lokalnej, startując z tego punktu, doprowadza do poszukiwanego rozwiązania optymalnego. Na ogół przyjmuje się niewielką liczbę iteracji dla algorytmu globalnego, co sprzyja odnalezieniu rozwiązania dość odległego od optymalnego, lecz na tyle bliskiego, aby obliczenia według algorytm lokalnego nie utknęły w optimum lokalnym.

W niniejszej pracy rozważano różne schematy uczenia hybrydowego, przy czym ograniczono się do algorytmów globalnych takich, jak:

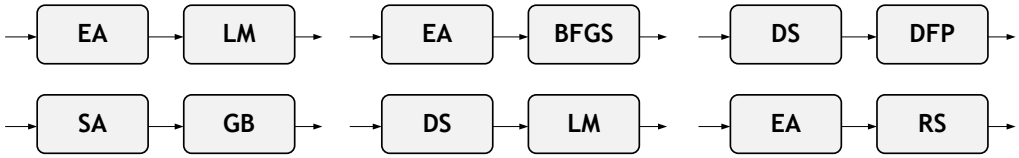
- algorytm ewolucyjny (EA),
- algorytm symulowanego wyżarzania (SA),
- algorytm przeszukiwania bezpośredniego (DS),

oraz poniższych algorytmów lokalnych:

- algorytm największego spadku (GB),
- algorytm zmiennej metryki z formułą DFP (DFP),
- algorytm zmiennej metryki z formułą BFGS (BFGS),
- algorytm Levenberga-Marquardta (LM),
- algorytm z losowym wyborem kierunku (RS) wg strategii: A1, B1, C1, D1.



(a) Ogólny schemat uczenia hybrydowego



(b) Wybrane strategie uczenia hybrydowego

Rys. 4.5: Hybrydowe strategie optymalizacji parametrów swobodnych sieci lokalnie rekurencyjnej

Na Rys. 4.5 (b) pokazano wybrane schematy uczenia hybrydowego takie jak: EA-LM, EA-BFGS, DS-DFP, SA-GB itd.

4.4. Selekcja wejść modelu neuronowego

Selekcja relewantnych wejść modelu neuronowego prowadzona może być na wiele sposobów. Zazwyczaj podczas budowy modelu neuronowego procesu istnieje dostęp do wiedzy eksperckiej, co umożliwia ograniczenie zbioru wszystkich możliwych kombinacji zmiennych procesowych. Automatyczna selekcja wejść ogranicza się tu do sytuacji, kiedy mamy do rozpatrzenia mniej niż 15 zmiennych procesowych. Wówczas gdy liczba atrybutów jest większa, niezbędne jest połączenie proponowanych metod z wybranym algorytmem przeszukiwania (np. algorytmem A*, algorytmem genetycznym).

4.4.1. Współrzędne z opóźnieniem

Odtworzenie przestrzeni fazowej jest zwykle pierwszym krokiem w analizie systemów dynamicznych (Ott, 1997). W dużej liczbie przypadków podczas obserwacji procesów (układów dynamicznych) nie jest możliwy jednoczesny pomiar wszystkich składowych wektora stanu $\mathbf{x}(k)$. Przypuśćmy, że możemy obserwować wyłącznie jedną wielkość charakteryzującą dany proces np. $g\,(k)$ taką, że:

$$g\,(k) = G\,(\mathbf{x}\,(k))\,. \tag{4.26}$$

Jeżeli celem jest analizowanie dynamiki procesu opisanej przez trajektorię fazową wyznaczaną dla wektora stanu $\mathbf{x}\,(k)$, to należy odtworzyć atraktor w wielowymiarowej przestrzeni fazowej, wykorzystując wyłącznie pomiary zapisane w postaci szeregu czasowego $g(k)$. Najbardziej znanym podejściem jest tu tak zwana metoda współrzędnych z opóźnieniem (Rosenstein *i in.*, 1993a; Rosenstein *i in.*, 1993b), polegająca na odpowiednim przetworzeniu szeregu czasowego $g(k)$ do postaci wektora $\mathbf{g}(k)$ takiego, że:

$$\mathbf{g}\,(k) = \left[\begin{array}{ccccc} g\,(k) & g\,(k+\tau) & g\,(k+2\tau) & \ldots & g\,(k+(d-1)\,\tau) \end{array}\right]^T, \tag{4.27}$$

gdzie d jest wymiarem zanurzenia, τ jest tzw. opóźnieniem czasowym. Pozwala to przyjąć, że wyjście modelu neuronowego w dyskretnej chwili k może stanowić zmienna stanu $g(k)$. Wejścia modelu stanowią pozostałe składowe $\mathbf{g}(k)$.

Jak łatwo zauważyć, wierność odwzorowania dynamiki silnie zależy od dwóch parametrów: d i τ. M. Rosenstein *i in.* (1993b) opisali kilka dróg postępowania, które umożliwiają dobre oszacowanie wartości czasu charakterystycznego (τ). Najczęściej stosowane są metody wykorzystujące funkcję autokorelacji liniowej lub miarę określaną jako wzajemna informacja (ang. *mutual information*). Do oszacowania wymiaru zanurzenia d najczęściej stosuje się metodę fałszywych najbliższych sąsiadów (ang. *false nearest neighbour method*) (Zbilut i Webber, 1992).

4.4.2. Test Gamma

Test Gamma jest procedurą umożliwiającą estymację wartości miary oceniającej poziom zakłóceń występujących w zbiorze danych (Kemp *i in.*, 2005; Stefánsson *i in.*, 1997).

W efekcie jej działania możliwe jest określenie podzbiorów danych reprezentujących różne konfiguracje wejść i wyjść modelu neuronowego oraz oszacowanie jakości danych potrzebnych do jego wytrenowania. Poziom zakłócenia rozpatrywanego podzbioru danych wejść i wyjść może być spowodowany następującymi czynnikami:

- niedokładnością pomiarów,
- pominięciem wejść, które mogą być istotne dla wybranego wyjścia,
- faktem, że poszukiwany związek wejściowo−wyjściowy nie jest funkcją ciągłą i różniczkowalną.

W podstawowej wersji testu Gamma rozpatrywany jest podzbiór danych w formie:

$$\{\mathbf{u}_i, y_i\}_{i=0}^{N}, \tag{4.28}$$

gdzie $\mathbf{u}_i \subset \mathbb{R}^{\mathsf{L}}$ jest wektorem potencjalnych wejść modelu, $y_i \subset \mathbb{R}^{\mathsf{M}}$ jest potencjalnym wyjściem modelu. Zależność pomiędzy wektorem wejściowym i wyjściem uwzględniająca wpływ zakłóceń wyrażana jest poprzez równanie:

$$y_i = f(\mathbf{u}_i) + r_i \tag{4.29}$$

gdzie f jest pewną nieznaną funkcją, r jest zmienną niezależną reprezentującą czynnik losowy.

Test Gamma oszacowuje wariancję σ_r w następujący sposób. W pierwszej kolejności tworzone jest k wymiarowe drzewo przy użyciu wektorów wejściowych $\mathbf{u}_i|_{(0\leq i\leq N)}$. Dla tak utworzonego drzewa wyznacza się $k = \{1, 2, \ldots, z\}$ najbliższych sąsiadów $\mathbf{u}'_{ik}|_{(0\leq i\leq N)}$ dla węzła $\mathbf{u}_i$. Następnie, dla każdego k obliczane są wielkości:

$$\delta_N(k) = \frac{1}{N}\sum_{i=0}^{N} \|\mathbf{u}'_{ik} - \mathbf{u}_i\|^2, \tag{4.30}$$

gdzie $\|\cdot\|$ oznacza jedną z podstawowych norm oraz

$$\gamma_N(k) = \frac{1}{2N}\sum_{i=0}^{N} \|y'_{ik} - y_i\|^2, \tag{4.31}$$

gdzie y_{ik} jest wyjściem przypisanym dla węzła $\mathbf{u}_{ik}$. Ostatecznie wyznaczana jest prosta regresji liniowej $\gamma_N(k) = \Gamma + A\delta_N(k)$. Wartość współczynnika Γ proporcjonalna jest do wariancji czynnika losowego. Parametr A niesie informację dotyczącą stopnia złożoności nieznanej zależności $y = f(\mathbf{u})$. Podstawy teoretyczne i zastosowania praktyczne testu (np. w doborze wejść sieci jednokierunkowych) opisano m.in. w pracach (Jarvis *i in.*, 2006; Jones, 2004; Jones *i in.*, 2007; Kemp *i in.*, 2005).

4.4.3. Metoda wskaźników pojemności informacyjnej

W prezentowanych badaniach autor zaproponował udoskonalenie koncepcji metody wskaźników pojemności informacyjnej (metoda Hellwiga). Metoda w swojej pierwot-

nej formie stosowana była w doborze zmiennych objaśniających modeli ekonometrycznych (Barczak i Biolik, 2003). W metodzie tej pojemność indywidualną nośnika informacji wyraża się w następujący sposób:

$$h_{jk}^{c} = \frac{\upsilon_{jk}^{2}}{1 + \sum\limits_{\substack{i=1 \\ i \neq j}}^{L_c} |\nu_{ij}|}, \tag{4.32}$$

gdzie υ_{jk} - współczynnik określający zależność pomiędzy k-tą zmienną wyjściową, a j-tą zmienną wejściową modelu, ν_{ij} współczynnik określający zależność pomiędzy i-tą i j-tą potencjalną zmienną wejściową modelu, $c = 1, 2, \ldots, 2^L - 1$ oznacza identyfikator kombinacji, L_c jest liczbą zmiennych w c-tej kombinacji. Pojemność integralną kombinacji potencjalnych wejść modelu neuronowego określa wyrażenie:

$$H_k^c = \sum_{j=1}^{L_c} h_{jk}^c. \tag{4.33}$$

Pojemność integralna stanowi kryterium wyboru odpowiedniego zestawu zmiennych wejściowych. Wybiera się tę kombinację zmiennych wejściowych, dla której H jest maksymalne. W odróżnieniu od standardowej metody opisanej w pracy (Barczak i Biolik, 2003), gdzie stosowano wyłącznie współczynnik korelacji liniowej Pearsona (r_{jk}), jako miary zależności υ_{jk} i ν_{ij}, w niniejszej pracy brano pod uwagę również:

- współczynnik korelacji rang Spearmana:

$$\rho_{jk} = 1 - 6\frac{\sum\limits_{i=0}^{N} \left[r\left(z_{ij}\right) - r\left(z_{ik}\right)\right]^2}{N\left(N^2 - 1\right)}, \tag{4.34}$$

- współczynnik korelacji rang Kendalla:

$$\tau_{jk} = \frac{2V}{N\left(N - 1\right)}, \tag{4.35}$$

- współczynnik korelacji nieliniowej:

$$nr_{jk} = \sqrt{1 - \frac{\sum\limits_{i=0}^{N} \left[Q_n\left(z_{ij}\right) - z_{ik}\right]^2}{\sum\limits_{i=0}^{N} \left(z_{ik} - \bar{Z}_k\right)^2}}, \tag{4.36}$$

- informację wzajemną:

$$I\left(Z_j, Z_k\right) = \sum_{s=0}^{N} \sum_{i=0}^{N} p\left(z_{ij}, z_{sk}\right) \log \frac{p\left(z_{ij}, z_{sk}\right)}{p\left(z_{ij}\right) p\left(z_{sk}\right)}, \tag{4.37}$$

gdzie $r(z_{ij})$, $r(z_{ik})$ - rangi badanych zmiennych, V to wyznaczona suma not dla wszystkich rang par, Q - wielomian stopnia n aproksymujący zależność pomiędzy badanymi zmiennymi, $p(z_{ij}, z_{sk})$ - prawdopodobieństwo w rozkładzie produktowym, a $p(z_{ij})$ i $p(z_{sk})$ - prawdopodobieństwa w rozkładzie zmiennych Z_j i Z_k.

Wymienione wyżej współczynniki pozwalają na badanie zależności statycznych. Wyznaczona na tej podstawie pojemność integralna zawiera jedynie informację o statycznej zależności nieliniowej dla danej kombinacji wejść modelu. Jako rozwiązanie tego problemu autor zaproponował rozszerzenie współczynników miary ν_{ij} o miary bazujące na analizie diagramów rekurencyjnych (umożliwiających wprowadzenie podobieństwa czasowego procesów).

Aby można było wyliczyć miary RQA dla wybranych zmiennych, niezbędne jest odpowiednie ich przekształcenie (w celu wyznaczenia poprzecznego diagramu rekurencyjnego). Przekształcenie rozpatrywanych zmiennych realizuje się stosując metodę współrzędnych z opóźnieniem (rozdział 4.4.1) otrzymując dwie trajektorie takie, że:

$$\begin{aligned} \mathbf{z}_{ij} &= \begin{bmatrix} z_{ij} & z_{(i+\tau_j)j} & z_{(i+2\tau_j)j} & \cdots & z_{(i+d\tau_j-\tau_j)j} \end{bmatrix}^T, \\ \mathbf{z}_{ik} &= \begin{bmatrix} z_{ik} & z_{(i+\tau_k)k} & z_{(i+2\tau_k)k} & \cdots & z_{(i+d\tau_k-\tau_k)k} \end{bmatrix}^T, \end{aligned} \tag{4.38}$$

gdzie $i = 0, 1, 2, \ldots, N - (d-1)\tau_j$, a czasy charakterystyczne τ_j, τ_k oraz wymiar zanurzenia d dobiera się za pomocą metod przytoczonych w rozdziale 4.4.1. Poprzeczny diagram rekurencyjny wyznaczamy z zastosowaniem cechy funkcyjnej (będącej miarą rekurencji) zdefiniowanej jako:

$$[\mathbf{CR}]_{nm} = H\left(\epsilon - \|\mathbf{z}_{nj} - \mathbf{z}_{mk}\|\right), \tag{4.39}$$

gdzie $n, m = 0, 1, 2, \ldots, N_1 = N - (d-1)\tau_j$, H to funkcja skokowa Heaviside'a, $\|\cdot\|$ jest wybraną metryką (w pracy rozważano metrykę euklidesową), próg $\epsilon = 5\sigma$, przy czym σ określa odchylenie standardowe szumu pomiarowego występującego w rozpatrywanych danych (Marwan, Romano, Thiel i Kurths, 2007; Thiel *i in.*, 2002). Dla utworzonego diagramu możemy wyliczyć znane miary RQA (Zbilut i Webber, 1992; Marwan, Kurths i Saparin, 2007). W pracy ograniczono się do miar RQA takich jak:

- Wskaźnik rekurencji - odsetek punktów rekurencyjnych w diagramie:

$$\mathrm{rr} = \frac{1}{N_1^2} \sum_{n,m=0}^{N_1} [\mathbf{CR}]_{nm}, \tag{4.40}$$

- Determinizm - zdefiniowany jako odsetek punktów rekurencyjnych, które należą do linii diagonalnych diagramu (o długościach co najmniej $l_{\min}$):

$$\det = \frac{\sum_{l=l_{\min}}^{N_1} lP(l)}{\sum_{l=1}^{N_1} lP(l)}, \tag{4.41}$$

- Entropia - entropia informacyjna Shannona wyznaczana dla rozkładu prawdopodobieństwa linii diagonalnych:

$$\mathrm{ent} = -\sum_{l=l_{\min}}^{N_1} p(l) \ln p(l) = -\sum_{l=l_{\min}}^{N_1} \frac{P(l)}{N_l} \ln \frac{P(l)}{N_l}, \tag{4.42}$$

- Laminarność - odsetek punktów należących do linii pionowych, których długość wynosi co najmniej $l_{\min}$:

$$\mathrm{lam} = \frac{\sum\limits_{l=l_{\min}}^{N_1} lP_v(l)}{\sum\limits_{l=1}^{N_1} lP_v(l)}, \tag{4.43}$$

- Czas pułapkowania - jest średnią długością linii pionowych diagramu:

$$\mathrm{tt} = \frac{\sum\limits_{l=l_{\min}}^{N_1} lP_v(l)}{\sum\limits_{l=l_{\min}}^{N_1} P_v(l)}, \tag{4.44}$$

gdzie $P(l)$ jest prawdopodobieństwem znalezienia linii diagonalnej o długości l na diagramie, N_l jest całkowitą liczbą linii diagonalnych diagramu.

Poza nowymi miarami dla υ_{jk} i ν_{ij} zastosowanymi przez autora, modyfikacja metody polegała na odpowiednim wskazywaniu potencjalnych wejść modelu. W oryginalnej metodzie jako zmienne wejściowe modelu wybiera się taką kombinację wejść, aby H_k^c było maksymalne. W niniejszych badaniach brano wartość średnią poszczególnych wejść wyliczoną dla pierwszych dziesięciu kombinacji, dla których uzyskano największe wartości H_k^c. W ten sposób uzyskano stopnie przekonania $[0, 1]$ wskazujące poziom istotności danego wejścia. Podobnie postępowano w przypadku testu Gamma.

4.5. Ocena działania modelu neuronowego procesu

Ocenę działania modeli neuronowych procesów można prowadzić na wiele sposobów. Najczęściej jednak stosuje się miary *ex post* opracowane w celu oceny poprawności prognoz dowolnych szeregów czasowych niezależnie od źródła ich pochodzenia.

Standardowe miary dokładności predykcji mogą być stosowane do oceny modeli procesów uzyskanych za pomocą dowolnej metody modelowania. Analizując literaturę dotyczącą zastosowań sieci neuronowych w diagnostyce i sterowaniu, można zauważyć, iż szczególne znaczenie w tej grupie mają następujące miary (Duch *i in.*, 2000; Korbicz *i in.*, 2004):

- pierwiastek z błędu średniokwadratowego (ang. *Root Mean Squared Error*):

$$\mathrm{RMSE} = \frac{1}{n^s} \sum_{j=1}^{n^s} \sqrt{\frac{1}{N_G} \sum_{i=0}^{N_G} (y_{ij} - \hat{y}_{ij})^2}, \tag{4.45}$$

- znormalizowany błąd RMSE (ang. *normalized RMSE*):

$$\mathrm{nRMSE} = \frac{1}{n^s} \sum_{j=1}^{n^s} \frac{1}{\sigma_{Y_j}} \sqrt{\frac{1}{N_G} \sum_{i=0}^{N_G} \left(y_{ij} - \hat{y}_{ij}\right)^2}, \tag{4.46}$$

- średni bezwzględny błąd procentowy (ang. *Mean Absolute Percentage Error*):

$$\mathrm{MAPE} = \frac{100\%}{n^s N_G} \sum_{j=1}^{n^s} \sum_{i=0}^{N_G} \left| \frac{y_{ij} - \hat{y}_{ij}}{y_{ij}} \right|, \tag{4.47}$$

- zmodyfikowany błąd MAPE (ang. *modified MAPE*):

$$\mathrm{mMAPE} = \frac{100\%}{n^s N_G} \sum_{j=1}^{n^s} \sum_{i=0}^{N_G} \left| \frac{y_{ij} - \hat{y}_{ij}}{\max\left(Y_j\right) - \min\left(Y_j\right)} \right|, \tag{4.48}$$

- współczynnik I^2 Theila (ang. *Theil's inequality coefficient*):

$$I^2 = \sum_{j=1}^{n^s} \frac{\sum\limits_{i=0}^{N_G} \left(y_{ij} - \hat{y}_{ij}\right)^2}{\sum\limits_{i=0}^{N_G} y_{ij}^2}, \tag{4.49}$$

gdzie y_{ij} to j-te wyjście modelu uzyskane dla i-tego wzorca wejściowego, $\hat{y}_{ij}$ oznacza oczekiwaną wartość j-tego wyjścia modelu, $N_G = \mathrm{card}\left(\mathcal{L}_G\right)$ jest licznością zbioru przykładów testowych.

Poza uniwersalnymi miarami wymienionymi powyżej bardzo często stosuje się oceny utworzone na podstawie innych klasycznych mierników takich, jak błąd średni, suma kwadratów błędów, błąd średniokwadratowy, współczynnik korelacji liniowej itp. Możliwe jest również zastosowanie testów statystycznych. Innym ważnym sposobem oceny modeli są względne miary dokładności. W tej grupie często stosowanymi miarami są (Duch *i in.*, 2000):

- względny błąd predykcji (ang. *Relative Prediction Error*):

$$\mathrm{RPE} = \sqrt{\frac{\sum\limits_{j=1}^{n^s} \sum\limits_{i=0}^{N_G} \left(y_{ij} - \hat{y}_{ij}\right)^2}{\sum\limits_{j=1}^{n^s} \sum\limits_{i=1}^{N_G} \left(y_{ij} - y_{ij}^w\right)^2}}, \tag{4.50}$$

- statystyka U Theila (ang. *Theil's U statistics*):

$$U = \sqrt{\frac{\sum\limits_{j=1}^{n^s} \sum\limits_{i=0}^{N_G} \left(y_{ij} - \hat{y}_{ij}\right)^2}{\sum\limits_{j=1}^{M} \sum\limits_{i=1}^{N_G} \left(y_{ij} - y_{ij}^w\right)^2}}, \tag{4.51}$$

gdzie y_{ij}^{w} to wartość j-tego wyjścia uzyskana dla modelu wzorcowego, np. modelu neuronowo-rozmytego, prognozy naiwnej itp.

Przytoczone powyżej miary oceny poprawności odwzorowania dynamiki procesu nie zawsze są dobrą podstawą oceny jakości uzyskanego modelu. Często w ocenie modelu niezbędne jest również uwzględnienie jego złożoności. Podczas tworzenia modeli neuronowych procesów autor brał pod uwagę kryteria informacyjne oceny modeli z uwzględnieniem ich złożoności, które znane są z ogólnej teorii identyfikacji oraz modelowania systemów (Konishi i Kitagawa, 2008; Söderström i Stoica, 1997):

- kryterium końcowego błędu predykcji (ang. *Final Prediction Error*):

$$\mathrm{FPE} = Q_G \left(1 + \frac{2p/\operatorname{card}(\mathcal{L}_G)}{1 - p/\operatorname{card}(\mathcal{L}_G)}\right), \tag{4.52}$$

- kryterium informacyjne Akaike (ang. *Akaike's Information Criterion*):

$$\mathrm{AIC} = \operatorname{card}(\mathcal{L}_G) \ln(Q_G) + 2p, \tag{4.53}$$

- kryterium Schwarza (ang. *Schwarz's Bayesian Criterion*):

$$\mathrm{BIC} = \operatorname{card}(\mathcal{L}_G) \ln(Q_G) + p \ln(\operatorname{card}(\mathcal{L}_G)), \tag{4.54}$$

- kryterium Hannana i Quinna (ang. *Phi Criterion*):

$$\mathrm{HQC} = \operatorname{card}(\mathcal{L}_G) \ln\left(\frac{Q_G}{\operatorname{card}(\mathcal{L}_G)}\right) + 2p \ln(\ln(\operatorname{card}(\mathcal{L}_G))), \tag{4.55}$$

- kryterium Jenkinsa-Wattsa (ang. *Jenkins and Watts Criterion*):

$$\mathrm{JEW} = Q_G \frac{(\operatorname{card}(\mathcal{L}_G) - p)}{\operatorname{card}(\mathcal{L}_G) - 2p - 1}, \tag{4.56}$$

gdzie $Q_G = \{\mathrm{RMSE}, \mathrm{nRMSE}, \ldots\}$ jest wybraną miarą jakości odwzorowania dynamiki procesu wyznaczoną dla zbioru danych testowych.

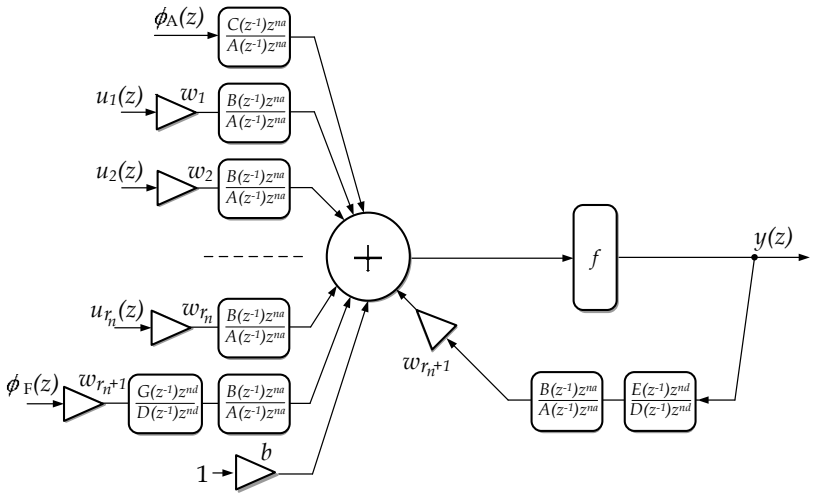
Kryteria informacyjne można stosować do doboru struktury modelu, np. stosując zasadę oszczędności lub ostrożności. Zazwyczaj dobór struktury realizuje się z zastosowaniem jednego wybranego kryterium. W sytuacjach, gdy zastosowanie jednego kryterium prowadzi do wskazania większej liczby modeli, należy użyć dodatkowych kryteriów do rozstrzygnięcia ostatecznego wyboru (Konishi i Kitagawa, 2008).

4.6. Analiza stabilności modelu neuronowego

Analizę stabilności zaproponowanych struktur lokalnie rekurencyjnych przeprowadzono, wykorzystując metodykę opracowaną w pracy (Patan, 2008b), przy czym stosowano zapis wejściowo-wyjściowy części dynamicznej neuronów. Jak wiadomo, struktura lokalnie rekurencyjna złożona jest z dynamicznych jednostek przetwarzających (podsystemów)

działających równolegle w warstwach, które połączone są ze sobą szeregowo (brak połączeń mających charakter sprzężeń zwrotnych i skrośnych).

Stabilność asymptotyczna sieci lokalnie rekurencyjnej jest więc zagwarantowana, jeżeli każdy z neuronów dynamicznych jest stabilny asymptotycznie (Patan, 2008b). Stosując zapis w postaci transmitancji operatorowych oraz dokonując prostych przekształceń jednostki opisanej w rozdziale 4.2.1, można ją przedstawić jak na Rys. 4.6. Dla uproszczenie dalszych rozważań przyjęto, że $na > nb$ oraz $nd > ne$.



Rys. 4.6: Neuron dynamiczny jako system złożonych z podsystemów typu wejście-wyjście

Dalszą analizę stabilności przeprowadzono dla dwóch przypadków neuronu: z liniową funkcją wyjścia (4.6), oraz z nieliniową funkcją wyjścia (4.8). Pierwszy przypadek pozwala na bezpośrednie zastosowanie znanych kryteriów stabilności asymptotycznej dla liniowych systemów dynamicznych.

Bardziej złożonym przypadkiem jest jednostka z nieliniową funkcją wyjścia. Jeżeli funkcja ta jest ograniczona (jak w przypadku rozpatrywanej funkcji sigmoidalnej), to jednostka jest stabilna w sensie BIBO (ang. *Bounded Input Bounded Output*). Niemniej jednak efekt działania części dynamicznej neuronu może prowadzić do sytuacji, że będzie się on zachowywał jak element przełączający (Patan, 2008b). Będzie to powodować, że uzyskany model neuronowy będzie stabilny w sensie BIBO, przy czym wiele elementów przetwarzających będzie bezużytecznych. Drugi przypadek rozważano, stosując pierwszą metodę Lapunowa, badając lokalną stabilność asymptotyczną dla punktu równowagi w początku układu współrzędnych.

1. **Warunki globalnej stabilności asymptotycznej neuronu liniowego**

   Do zbadania stabilności asymptotycznej i-tego neuronu w sieci, który posiada liniową funkcję wyjścia, wystarczające jest rozpatrzenie położenia pierwiastków następujących wielomianów charakterystycznych:

$$
\begin{aligned}
A(z) &= z^{na} + a_1 z^{na-1} + \ldots + a_{na}\,, && (4.57)\\
D(z) &= z^{nd} + d_1 z^{nd-1} + \ldots + d_{nd}\,, && (4.58)\\
H(z) &= A(z)\,D(z) - w_{r_n+1} B(z)\,E(z) = && (4.59)\\
&= h_0 z^{nh} + h_1 z^{nh-1} + \ldots + h_{nh}\,,
\end{aligned}
$$

gdzie stopień wielomianu $H(z)$ wynosi $nh = \max\{\deg(A(z)\,D(z)),\ \deg(B(z)\,E(z))\}$, a jego współczynniki obliczamy według wzoru

$$h_j = \sum_{k}^{nh} a_k d_{j-k} - w_{r_n+1} \sum_{l}^{nh} b_l e_{j-l}, \tag{4.60}$$

przy czym $j = 0, 1, \ldots, nh$, $a_k = 0$ dla $k > na$, $d_{j-k} = 0$ dla $0 > j - k > nd$ oraz $b_l = 0$ dla $l > nb$, $e_{j-l} = 0$ dla $0 > j - l > ne$.

2. **Warunki lokalnej stabilności asymptotycznej neuronu nieliniowego**

   Dla nieliniowego modelu neuronu można sformułować warunki lokalnej stabilności asymptotycznej z zastosowaniem pierwszej metody Lapunowa. W tym celu należy zbadać zachowanie neuronu w pobliżu punktu równowagi będącego początkiem układu współrzędnych. Jeżeli rozwinie się funkcję wyjścia neuronu w postaci $f(x) = \tanh(x/\alpha_f)$ w punkcie $x = 0$ w szereg Taylora i pominie wyrazy wyższego rzędu, to otrzyma się jej liniowe przybliżenie $f(x) = x/\alpha_f$. Wtedy warunki dla wielomianów charakterystycznych (4.57,4.58) pozostają bez zmian, a wielomian (4.59) przyjmuje następującą formę:

$$H(z) = \alpha_f A(z)\,D(z) - w_{r_n+1} B(z)\,E(z)\,. \tag{4.61}$$

Biorąc pod uwagę przytoczone w obu punktach rozważania, model neuronowy zbudowany z liniowych i nieliniowych jednostek dynamicznych (zaproponowanych w niniejszej pracy) będzie stabilny asymptotycznie lub stabilny asymptotycznie lokalnie wtedy, jeżeli wszystkie pierwiastki wielomianów charakterystycznych (4.57-4.59) dynamicznych neuronów liniowych lub (4.57,4.58,4.61) nieliniowych neuronów dynamicznych sieci znajdować się będą wewnątrz koła o promieniu jednostkowym.

Do badania położenia pierwiastków wielomianów charakterystycznych na płaszczyźnie zmiennej zespolonej można zastosować kryterium Schura-Cohna (Kaczorek, 1999). Pozwala to zdecydować na podstawie znajomości wartości współczynników badanego wielomianu, czy pierwiastki tego wielomianu znajdują się w kole jednostkowym.

Twierdzenie (Kaczorek *i in.*, 2005)

Pierwiastki wielomianu charakterystycznego znajdują się wewnątrz koła o promieniu jednostkowym wtedy i tylko wtedy, gdy

1. $|a_{na}| < a_0\,,$

2. wielomian stopnia $na-1$

$$m(z) = \frac{1}{z}\left[a_0 A(z) - a_{na} A\left(z^{-1}\right) z^{na}\right] = a_0^* z^{na-1} + a_1^* z^{na-2} + \ldots + a_{na-1}^*, \quad (4.62)$$

ma również pierwiastki w kole jednostkowym. Współczynniki wielomianu $m(z)$ wyznacza się według następującego schematu:

$$a_k^* = a_0 a_k - a_{na} a_{na-k} \text{ dla } k = 0, 1, \ldots, na-1. \quad (4.63)$$

Powyższe twierdzenie można udowodnić indukcyjnie. Kryterium Schura-Cohna jest kryterium koniecznym i wystarczającym. Charakterystyczną cechą tego kryterium jest to, że stabilność wielomianu charakterystycznego bada się w sposób rekurencyjny.

4.7. Poszukiwanie modelu suboptymalnego

Dobór architektury sieci neuronowej jest jednym z czynników wpływających na zdolność uogólniania (generalizacji) tej sieci. Proponowana metoda selekcji struktury sieci jest połączeniem podejścia bazującego na izoliniach kryterialnych (Pokropińska *i in.*, 2006; Rutkowski, 2005) oraz podejścia umożliwiającego eliminację połączeń nieistotnych.

Pierwsza z wymienionych metod stosowana jest do wstępnego doboru struktury sieci, przy czym możliwe jest zarówno porównywanie topologii sieci o różnym typie, jak i modeli innej klasy (np. modeli parametrycznych lub modeli neuronowo-rozmytych). Jest to podejście, w którym wykreślana jest mapa izolinii dla określonych kryteriów informacyjnych (znanych z identyfikacji systemów). Izolinie reprezentują obszary, dla których wartości rozpatrywanego kryterium są stałe w funkcji złożoności modelu p oraz wartości oceny dokładności modelu Q_G. W badaniach autor stosował szereg kryteriów informacyjnych, z czego najważniejsze zostały omówione w poprzednim podrozdziale. Na tak wykreślanych mapach nanosi się wyniki testów dla wygenerowanych modeli. Liczbę rozpatrywanych modeli można oszacować, stosując heurystyki z rozdziału B.4. Zastosowanie wykresów izolinii kryterialnych oraz zasady oszczędności (lub ostrożności) pozwala na wskazanie struktury najbardziej obiecującej.

W kolejnym kroku, dla tak wybranego modelu, stosowana jest jedna ze znanych procedur przycinania sieci, której kluczowym elementem jest wyznaczenie miary istotności s_k dla każdego parametru swobodnego sieci. Poniżej opisano trzy metody, które autor stosował do optymalizacji struktury sieci lokalnie rekurencyjnej.

1. Iteracyjne usuwanie parametrów na podstawie wartości wskaźnika

$$s_k = E(\boldsymbol{\omega}^*) - E(\boldsymbol{\omega}_0^*), \quad (4.64)$$

gdzie $\boldsymbol{\omega}^*$ jest wektorem parametrów wyznaczonym w procesie uczenia, zaś $\boldsymbol{\omega}_0^* = \left[\omega_1^* \ \omega_2^* \ \ldots \omega_k^* = 0 \ \omega_{k+1}^* \ \ldots \ \omega_p^*\right]$ jest wektorem z wyzerowanym k-tym parametrem. Po każdorazowej eliminacji parametru, dla którego s_k ma najmniejszą war-

tość, przeprowadza się douczania sieci z zastosowaniem algorytmu lokalnego. Proces ten powtarza się do momentu, kiedy douczanie nie pozwala na osiągnięcie błędu testowania na poziomie osiąganym przed etapem przycinania.

2. Eliminacja parametrów, dla których

$$s_k = H\left(\epsilon_\omega - |\omega_k|\right) = 0, \tag{4.65}$$

gdzie ϵ_ω jest wartością progową określającą, które z parametrów należy usunąć, zaś H jest funkcją skokową Heaviside'a. Po usunięciu parametrów przeprowadza się proces douczania z zastosowaniem algorytmu lokalnego.

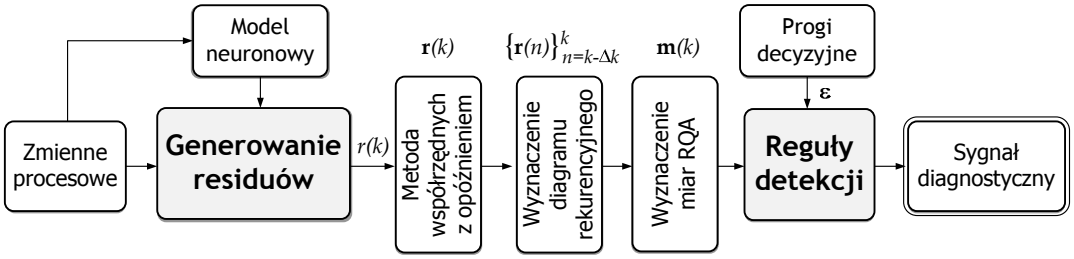
3. Eliminacja parametrów z zastosowaniem metody OBD (ang. *Optimal Brain Damage*), w której istotność k-tego parametru sieci wyznaczamy z zależności (Cun *i in.*, 1990):

$$s_k = \frac{1}{2}\frac{\partial^2 E\left(\boldsymbol{\omega}\right)}{\partial \omega_k^2}\omega_k^2 = \frac{1}{2}\left[\mathbf{H}\left(\boldsymbol{\omega}\right)\right]_{kk}\omega_k^2 = \frac{1}{2}\left[\boldsymbol{J}\left(\boldsymbol{\omega}\right)\boldsymbol{J}\left(\boldsymbol{\omega}\right)^T\right]_{kk}\omega_k^2, \tag{4.66}$$

gdzie h_{kk} jest k-tym elementem diagonalnym macierzy drugich pochodnych funkcji (4.25). Elementy te wyznaczane są bezpośrednio lub z użyciem aproksymacji za pomocą jakobianu $\boldsymbol{J}$. Parametry, dla których uzyskano najmniejsze wartości wskaźnika s_k, zostają usunięte, a model douczany jest z zastosowaniem algorytmu lokalnego.

4.8. Metoda odpornej detekcji uszkodzeń

Proponowana metoda detekcji uszkodzeń polega na generacji residuów z zastosowaniem opracowanych struktur neuronowych w sposób klasyczny. Nowym elementem w podejściu zaproponowanym przez autora jest zastosowanie analizy diagramów rekurencyjnych do oceny residuów w bloku decyzyjnym systemu detekcji. Analiza ilościowa diagramów rekurencyjnych (RQA) wyznaczonych dla residuów prowadzona jest w celu podniesienia odporności oceny tych sygnałów. Schemat blokowy metody (dla jednego residuum) pokazano na Rys. 4.7.



Rys. 4.7: Schemat blokowy metody detekcji uszkodzeń opartej na miarach RQA

Model neuronowy tworzony jest dla danych pozyskanych podczas pracy obiektu w stanie pełnej zdatności. Na podstawie utworzonego modelu generowane jest residuum. Poprawnie wyznaczony model zapewnia, że dla pracy nominalnej obiektu sygnał ten ma

charakter procesu stochastycznego o nieznanym rozkładzie. Uzyskane residuum przekształcane jest za pomocą metody współrzędnych z opóźnieniem (rozdział 4.4.1):

$$\mathbf{r}(k) = \{r(k), r(k+\tau), \ldots, r(k+(d-1)\tau)\}^T. \tag{4.67}$$

W ten sposób uzyskuje się trajektorię reprezentującą dynamikę zmian residuum. Jest to podstawa do wyznaczenia miary rekurencji opisanej zależnością:

$$[\mathbf{RR}]_{ij} = H(\epsilon - \|\mathbf{r}(i) - \mathbf{r}(j)\|), \tag{4.68}$$

gdzie $i, j = k-\Delta k, k-\Delta k+1, \ldots, k$, Δk to szerokość okna przesuwnego określającego długość realizacji residuum używanego do wyznaczenia diagramu rekurencyjnego. Dla tak wyliczonej cechy funkcyjnej możliwe jest wygenerowanie diagramu rekurencyjnego oraz miar RQA - $\mathbf{m}(k) = [\mathrm{rr}(k)\ \det(k)\ \ldots]$ charakteryzujących złożoność zmian residuum. Efektem pojawienia się uszkodzenia jest zaburzenie (np. pojawienie się dodatkowych linii diagonalnych, zmiana ich długości itp.) w diagramie rekurencyjnym, co bezpośrednio wpływa na wyznaczane ilościowe miary RQA. Część decyzyjna zaproponowanego systemu detekcji wykorzystuje zbiór prostych reguł:

$$\mathcal{R} = \left\{ \begin{array}{lllll} \textbf{if} & m_1^i(k) \geq \epsilon_1^i & \textbf{then} & \mathcal{F}_1^i = 1 & \textbf{else} \quad \mathcal{F}_1^i = 0; \\ \textbf{if} & m_2^i(k) \geq \epsilon_2^i & \textbf{then} & \mathcal{F}_2^i = 1 & \textbf{else} \quad \mathcal{F}_2^i = 0; \\ \ldots & & & & \end{array} \right\}_{i=1}^{n}, \tag{4.69}$$

których przesłanki oparte są na stałych progach detekcyjnych określonych dla danych miar RQA. Uszkodzenie wykrywane jest w sytuacji, gdy wszystkie flagi $\mathcal{F}_j^i$ wyznaczone dla i-tego residuum w chwili k przyjmują wartość 1. W opisywanej metodzie można wyróżnić dwie grupy parametrów niezbędnych do prawidłowej pracy systemu detekcji. Pierwsza grupa parametrów dotyczy etapu wyznaczenia diagramu rekurencyjnego (τ^i, d^i, ϵ^i). Pozostałe parametry (ϵ_j^i) stanowią podstawę zbioru reguł (4.69) systemu detekcji uszkodzeń. Możliwe są następujące sposoby ich określenia:

- na podstawie analizy diagramów rekurencyjnych i wiedzy ekspertów,
- z zastosowaniem algorytmu optymalizacji,
  - na podstawie danych reprezentujących obiekt w stanie pełnej zdatności oraz z uszkodzeniami,
  - wyłącznie na podstawie danych zgromadzonych podczas pracy nominalnej diagnozowanego obiektu.

W pierwszym z wymienionych podejść parametry dobiera się arbitralnie, stosując metody przytoczone w rozdziale 4.4.1 oraz wykonując proste analizy diagramów rekurencyjnych i uzyskiwanych dla nich miar RQA. Drugi sposób wymaga optymalizacji wielokryterialnej odpowiednio zdefiniowanej funkcji oceniającej poprawność działania projektowanego systemu detekcji. Ogólna postać optymalizowanej funkcji jest następująca:

$$\mathbf{\Phi}(\boldsymbol{\kappa}) = [\Phi_1\ \Phi_2\ \ldots\ \Phi_n]^T, \tag{4.70}$$

gdzie $\boldsymbol{\kappa} = [\tau \ d \ \ldots]$ jest wektorem parametrów analizy diagramów rekurencyjnych, Φ_1, Φ_2, ... - wskaźniki (tj. sprawność, błędy względne i in.) będące składowymi oceny poprawności systemu detekcji.

Jeżeli dysponuje się zbiorem danych reprezentującym obiekt w różnych stanach, zadanie optymalizacji nie wymaga dodatkowych działań. Natomiast gdy istnieje dostęp do danych zgromadzonych wyłącznie podczas nominalnej pracy obiektu, niezbędne jest działanie polegające na wygenerowaniu przykładów z fikcyjnie wprowadzanymi uszkodzeniami. Polega to na dodaniu wygenerowanego sygnału o odpowiedniej postaci do residuum uzyskanego dla danych nominalnych. Kształt generowanego sygnału może być dobierany na podstawie wiedzy o danym procesie. Jednym z możliwych rozwiązań jest określenie kształtu tego sygnału tak aby był on podobny do sygnału sterującego (wymuszenia).

4.9. Ocena sprawności systemu diagnostycznego

Formalnie ocena merytoryczno-techniczna poprawności systemu diagnostycznego może być zrealizowana poprzez określenie wartości estymaty sprawności części decyzyjnej tego systemu w odniesieniu do zbioru przykładów testowych. Zakłada się przy tym, *że rozkład przykładów jest losowy i reprezentatywny dla oceny systemu* (Moczulski, 2002). Miarę całkowitej sprawności systemu określa się zazwyczaj jako:

$$\eta_{ov} = 1 - \epsilon_{ov}, \tag{4.71}$$

gdzie ϵ_{ov} jest całkowitym błędem względnym wyrażanym jako:

$$\epsilon_{ov} = \frac{n_{err}}{\mathrm{card}\left(\mathcal{L}_G\right)}, \tag{4.72}$$

gdzie n_{err} oznacza liczbę przykładów błędnie sklasyfikowanych. Tak zdefiniowana miara określa zdolność systemu diagnostycznego do generalizacji zdobytej wiedzy w trakcie uczenia/formułowania części decyzyjnej systemu.

Oprócz łącznego błędu względnego istnieją inne efektywne miary błędu pozwalające na zidentyfikowanie przyczyny obniżenia sprawności ogólnej klasyfikatora (Moczulski, 2002). Pierwszy z nich to względny błąd pominięcia wyrażony jako:

$$\epsilon_{om} = \frac{1}{K} \sum_{k=1}^{K} \frac{\mathrm{card}\left(\mathcal{L}_{G_k}^{+}\right)}{\mathrm{card}\left(\mathcal{L}_{G_k}\right)}, \tag{4.73}$$

gdzie $\mathcal{L}_{G_k}^{+}$ oznacza podzbiór przykładów błędnie pominiętych, $\mathcal{L}_{G_k}$ jest podzbiorem przykładów klasy o indeksie k. Drugi z błędów cząstkowych, to względny błąd niesłusznego zaliczenia:

$$\epsilon_{cm} = \frac{1}{K} \sum_{k=1}^{K} \left(\frac{1}{\mathrm{card}\left(\mathcal{L}_G \setminus \mathcal{L}_{G_k}\right)} \sum_{\substack{j=1 \\ j \neq K}}^{K} \mathrm{card}\left(\mathcal{L}_{G_{kj}}^{-}\right) \right), \tag{4.74}$$

gdzie $\mathcal{L}^{-}_{G_{kj}}$ jest zbiorem przykładów klasy j niesłusznie zaklasyfikowanych do klasy k.

W. Moczulski (2002) wskazał na istotną wadę przytoczonych estymatorów, która wiąże się z niezrównoważeniem rozkładu zbioru przykładów w poszczególnych klasach. Sytuacja taka zazwyczaj obserwowana jest w zbiorach rzeczywistych danych i zaleca się wtedy stosować ważone błędy względne (Moczulski, 2002).

W niektórych sytuacjach definiuje się specyficzne miary (na podstawie wskaźników 4.71-4.74) dla danego problemu. Przykładem takich wskaźników mogą być miary zdefiniowane w ramach problemu testowego projektu DAMADICS (DAMADICS, 2002). W niniejszych badaniach autor stosował dwie zaproponowane w ramach projektu miary, które służą do oceny części decyzyjnej systemu detekcji uszkodzeń (Bartyś *i in.*, 2006). Pierwsza z nich to wskaźnik fałszywych alarmów (ang. *false detection rate*):

$$r_{fd} = \frac{\sum_i t^i_{fd}}{t_f - t_o}, \tag{4.75}$$

gdzie t^i_{fd} jest okresem czasu występowania i-tego fałszywego alarmu, t_f oznacza punkt czasu, w którym pojawiło się uszkodzenie, t_o oznacza punkt czasu, od którego rozpatrywany jest test. Drugi wskaźnik dotyczy poprawnie generowanych alarmów (ang. *true detection rate*) i jest zdefiniowany jako

$$r_{td} = \frac{\sum_i t^i_{td}}{t_h - t_f}, \tag{4.76}$$

gdzie t^i_{td} jest okresem czasu występowania i-tego poprawnie wskazanego alarmu, t_h jest punktem czasu oznaczającym koniec istotnego fragmentu strefy testu.

Do estymacji sprawności części decyzyjnej systemu diagnostycznego można zastosować metody wywodzące się z teorii estymacji takie jak:

- **Metoda resubstytucji**, która polega na testowaniu części decyzyjnej systemu wyłącznie za pomocą danych uczących.
- **Metoda holdout**, która polega na podziale zbioru dostępnych próbek na dwa wzajemnie wykluczające się podzbiory o podobnej liczebności.
- **Metoda rotacji (*leave-k-out*)**, w której dokonuje się $m = N/k$ podziałów poprzez przypisanie k wzorców do zbioru testujacego oraz pozostałych $(N - k)$ wzorców do zbioru trenującego. Uzyskane w ten sposób m wyników najczęściej uśredniania się.

Wymienione metody stosowane były między innymi w ocenie sprawności systemów wnioskujących (Korbicz *i in.*, 2004), w tym w ocenie sprawności baz wiedzy systemów ekspertowych (Moczulski, 2002).

4.10. Podsumowanie

W niniejszym rozdziale zaprezentowano metodykę modelowania neuronowego, w której uwzględniono wybrane elementy teorii chaosu deterministycznego. Metodyka ta przedstawia sposób budowy modelu neuronowego procesu oraz jego zastosowania do odpornej

detekcji uszkodzeń o różnym charakterze. Podstawę metodyki stanowi opis formalny lokalnie rekurencyjnej sieci neuronowej opartej na dynamicznym modelu neuronu (o złożonej dynamice), zaproponowanym przez autora rozprawy.

Dla opracowanej struktury lokalnie rekurencyjnej zaadaptowano szereg znanych algorytmów optymalizacji globalnej i lokalnej funkcji wielu zmiennych. W algorytmach globalnych zaproponowano nowe sposoby generowania elementów przestrzeni poszukiwań (swobodnych parametrów sieci) z zastosowaniem systemów chaotycznych, tj. odwzorowania Hénona, Ikedy oraz dyskretnego równania Mackeya-Glassa. Algorytmy te stanowią podstawę opracowanych schematów uczenia hybrydowego, które umożliwiają skuteczną identyfikację parametrów swobodnych modelu. Przedstawiono znane sposoby doboru wejść relewantnych modelu neuronowego. Biorąc pod uwagę niedoskonałości tych metod, zaproponowano modyfikację metody wskaźników pojemności informacyjnej o miary nieliniowe oraz miary niosące informację o zależnościach czasowych w badanych sygnałach. Dodatkowo rozwiązano problem doboru struktury budowanego modelu za pomocą podejścia bazującego na metodzie izolinii kryterialnych oraz znanych sposobach przycinania połączeń nieistotnych sieci. W ramach metodyki opracowano również kryteria badania stabilności asymptotycznej modeli neuronowych uzyskiwanych w wyniku identyfikacji.

Zaproponowana postać modelu neuronowego oraz sposób jego budowy nie eliminują całkowicie niepewności strukturalnej i parametrycznej. Dlatego opracowana metodyka uwzględnia metodę odpornej detekcji uszkodzeń opartą na analizie ilościowej diagramów rekurencyjnych wyznaczanych dla residuów.

Rozdział 5

Badania weryfikacyjne

W rozdziale przedstawiono wyniki badań weryfikacyjnych opracowanej metodyki modelowania neuronowego z uwzględnieniem elementów teorii chaosu deterministycznego. W badaniach wykorzystano dane zgromadzone w wyniku symulacji wybranych układów dynamicznych oraz dane pozyskane w trakcie realizacji procesów przemysłowych.

5.1. Plan weryfikacji

W celu udowodnienia postawionej w pracy tezy przeprowadzono badania weryfikacyjne w następujących etapach:

1. **Badania wstępne** mające na celu wykazanie przydatności opracowanej metodyki do modelowania procesów dynamicznych. Badania te obejmowały następujące najważniejsze zadania:

    - wybór modelowanych procesów oraz struktur wzorcowych,
    - porównanie klasycznych metod optymalizacji użytych do uczenia zaprojektowanych struktur neuronowych,
    - porównanie zaproponowanych schematów uczenia hybrydowego,
    - modelowanie procesów, których nośnikami są modele matematyczne,
    - modelowanie procesów rzeczywistych.

2. **Badania aplikacyjne** dotyczące zastosowania opracowanej metodyki w zadaniu odpornej detekcji uszkodzeń wybranego urządzenia pracującego w warunkach rzeczywistych.

5.2. Oprogramowanie

W celu przeprowadzenia badań weryfikacyjnych skorzystano z systemu obliczeń naukowych i inżynierskich MATLAB firmy MathWorks. Podczas badań korzystano z gotowych pakietów (Leontitsis, 2009; Marwan, 2009; The MathWorks, 2007), takich jak:

- Chaotic Systems Toolbox,
- Cross Recurrence Plot Toolbox,
- Genetic Algorithm and Direct Search Toolbox,
- Neural Networks Toolbox,
- Optimization Toolbox,
- System Identification Toolbox,

oraz opracowano własne biblioteki systemu umożliwiające:

- automatyczne generowanie danych używanych do celów weryfikacyjnych,
- tworzenie i trenowanie struktur lokalnie rekurencyjnych globalnie jednokierunkowych (zaproponowanych przez autora),
- tworzenie i trenowanie wielokontekstowych struktur Jordana i Elmana,
- wybór zmiennych niezależnych z zastosowaniem zmodyfikowanej metody wskaźników pojemności informacyjnej,
- dobór struktury modeli neuronowych z zastosowaniem izolinii kryterialnych oraz wybranych metod przycinania sieci,
- ocenę poprawności działania tworzonych modeli.

Opracowane przez autora biblioteki tworzą nowy pakiet systemu obliczeń inżynierskich MATLAB, umożliwiający budowę i trenowanie modeli neuronowych w postaci zaproponowanych struktur lokalnie rekurencyjnych oraz wielokontekstowych sieci Jordana i Elmana.

5.3. Weryfikacja wstępna

W podrozdziale opisano przebieg weryfikacji opracowanej metodyki w typowym zadaniu modelowania procesów technicznych. Głównym celem badań było określenie przydatności zaproponowanej struktury lokalnie rekurencyjnej do modelowania różnego rodzaju procesów oraz porównanie jej z innymi architekturami neuropodobnymi powszechnie stosowanymi. Niektóre fragmenty badań prezentowane w poniższym opisie omówione były częściowo w pracach (Przystałka i Moczulski, 2005; Przystałka, 2006; Przystałka, 2007a; Przystałka, 2007b; Przystałka, 2008; Przystałka, 2009).

5.3.1. Wybór modelowanych procesów i struktur wzorcowych

Obiektywne porównanie opracowanych struktur lokalnie rekurencyjnych z rozwiązaniami już istniejącymi w zadaniach modelowania procesów wymagało zgromadzenia odpowiedniego zbioru przykładów testowych. Ponadto należało wybrać odpowiednie rodzaje sieci neuronowych, które umożliwiałyby relatywną ocenę uzyskanych wyników modelowania. Przyjęto następujące kryteria doboru problemów testowych:

- możliwie szerokie spektrum klas procesów,
- dostępność informacji dotyczących procesów,
- możliwość obiektywnej oceny opracowanej metodyki modelowania,

oraz kryteria doboru wzorcowych struktur sieci:

- możliwość identyfikacji obiektów klasy MIMO,
- istnienie skutecznych algorytmów uczących.

Autor dokonał przeglądu wielu źródeł danych mogących stanowić podstawę relatywnej oceny opracowanego podejścia. Do najważniejszych źródeł danych, które autor brał pod uwagę, należy zaliczyć: bazę danych DaISy (ang. Database for the Identification of Systems) rozwijaną i udostępnianą przez (De Moor, 2009), bazę danych zgromadzonych przez E. Keogha i współpracowników (Keogh *i in.*, 2009), bazę danych archiwalnych zarejestrowanych podczas monitorowania złożonego obiektu przemysłowego (Szulim i Moczulski, 2005). Ponadto, aby uzupełnić spektrum rozpatrywanych problemów testowych o klasy procesów nieuwzględnione w przytoczonych bazach danych, autor opracował własne funkcje umożliwiające generowanie danych na podstawie równań różnicowych i różniczkowych opisujących zachowanie znanych układów dynamicznych (Li, 2006; Li *i in.*, 2006; Narendra i Parthasarathy, 1990).

Ze względu na fakt, że wiele problemów testowych ma podobny charakter, w dalszych rozważaniach dotyczących weryfikacji wstępnej rozpatrzono następujące źródła danych:

- system nieliniowy klasy MISO,
- nieliniowy system dynamiczny klasy MIMO,
- oscylator Duffinga z sygnałem wymuszenia,
- proces przemysłowy zwijania tkaniny,
- proces przemysłowy redukcji miedzi z żużla.

Początkowo badania porównawcze struktur lokalnie rekurencyjnych zaproponowanych przez autora prowadzone były w odniesieniu do wielowarstwowych struktur Elmana i Jordana oraz sieci z liniami opóźniającymi. W obliczeniach wykorzystano środowisko do obliczeń rozproszonych (Tomanek *i in.*, 2006; Tomanek *i in.*, 2007). Przeprowadzone doświadczenia wykazały przewagę zaprojektowanych przez autora struktur sieci lokalnie rekurencyjnych nad strukturami Jordana/Elmana w zadaniach modelowania procesów. Było to spowodowane faktem, że struktury wzorcowe (klasyczne sieci Elmana i Jordana oraz sieci z liniami opóźniającymi) mają ograniczenia wynikające z ich natury (np. pamiętanie stanu wyłącznie z poprzedniego kroku w przypadku pierwszej klasy sieci oraz szeregowo-równoległy sposób identyfikacji w przypadku drugiej). Dlatego też w bieżących badaniach jako wzorcowe struktury wykorzystano:

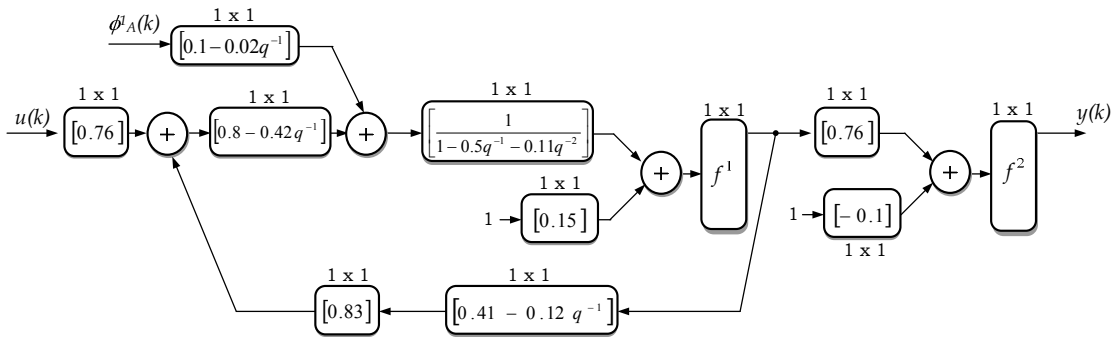
- wielokontekstowe sieci neuronowe Elmana i Jordana,
- sieci neuronowe typu NNARX.

Struktury tego typu pozbawione są niedoskonałości, o których była mowa powyżej. Wielokontekstowe sieci Elmana i Jordana wymagały zaadaptowania schematów uczenia hybrydowego, które zaproponowane zostały przez autora dla struktur lokalnie rekurencyjnych. Sieci typu NNARX uczone były z zastosowaniem funkcji dostępnych w przyborniku Neural Networks Toolbox oprogramowania MATLAB.

5.3.2. Porównanie hybrydowych algorytmów uczących

Aby pokazać zalety algorytmów hybrydowego uczenia opracowanych struktur lokalnie rekurencyjnych w porównaniu z algorytmami klasycznego strojenia parametrów, rozpatrzono następujący przykład modelowania procesu, którego nośnikiem był nieliniowy system dynamiczny klasy SISO.

System ten celowo zapisano w postaci dwuwarstwowej struktury sieci pokazanej na Rys. 5.1. Warstwa ukryta złożona jest z jednego neuronu z liniowymi systemami dynamicznymi w bloku aktywacyjnym i w bloku sprzężenia wyjściowego. Warstwa wyjściowa zawiera jeden neuron statyczny. Jako funkcje wyjścia neuronów przyjęto odpowiednio tangens hiperboliczny f^1 ($\alpha_f = 1$) oraz funkcję liniową f^2. Zakłócenie $\phi_A^1(k)$ oddziałujące na system miało charakter procesu stochastycznego o rozkładzie normalnym $\mathcal{N}(0, 0.1)$.



Rys. 5.1: Obiekt identyfikacji

Na wejście systemu zadano wymuszenie $\{u(k)\}_{k=1}^{3000}$ w postaci pseudolosowego ciągu binarnego (ang. *Pseudorandom Binary Sequence*), uzyskując odpowiedź $y(k)$ systemu. Przyjęto zerowe warunki początkowe oraz prawdopodobieństwo zmiany poziomu sygnału $\alpha_{\mathrm{PBS}} = 0.90$.

Zadanie polegało na zamodelowaniu procesu za pomocą struktury neuronowej zaproponowanej w rozdziale 4.2. Przyjęto ograniczenie, że struktura modelu neuronowego równoważna jest strukturze identyfikowanego systemu: $1 \to 1^{(2,2,2)}_{(0,2,0)} \to 1$. Wygenerowane dane podzielono na trzy zbiory: podzbiór danych trenujących $\mathcal{L}_T$, podzbiór danych weryfikacyjnych $\mathcal{L}_V$, oraz podzbiór danych testowych $\mathcal{L}_G$ w proporcjach 50%, 20%, 30%. W pierwszej fazie eksperymentu identyfikacyjnego dostrajano parametry modelu neuronowego minimalizując funkcję celu (4.25) w postaci sumy kwadratów błędu ($R = 2$,

$\lambda = 0$) z użyciem standardowych algorytmów opartych na metodach optymalizacji lokalnej (z aproksymacją gradientów i jakobianów funkcji celu). Dla każdego z algorytmów przyjęto wstępny dobór parametrów modelu zgodnie z metodą INIT0 (patrz rozdz. B.5). Proces trenowania powtarzano dziesięciokrotnie dla każdego algorytmu.

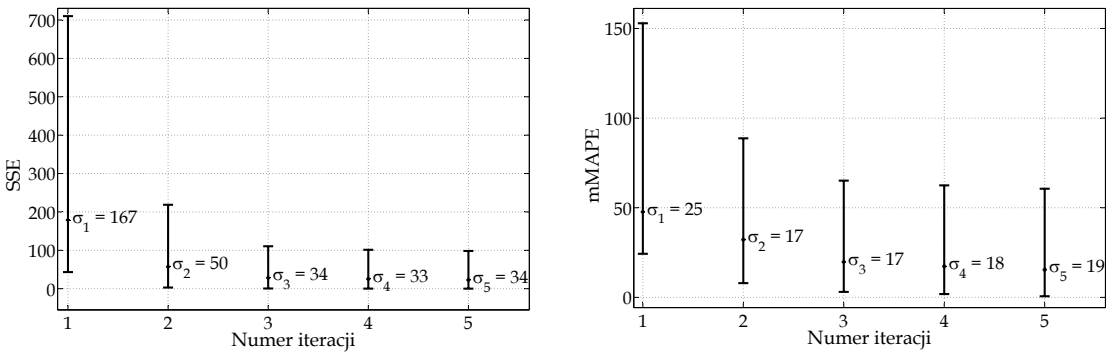
W Tab. 5.1 zawarto wartości średniego błędu testowania (mMAPE) oraz ich rozrzut (σ) dla danych ze zbioru $\mathcal{L}_G$. Na przykład dla algorytmu LM tylko w czterech próbach uzyskano błędy testowania $\mathrm{mMAPE} < 5\%$. Otrzymane wyniki potwierdzają dużą zależność wyniku modelowania od początkowych wartości parametrów swobodnych sieci neuronowej niezależnie od przyjętego algorytmu strojenia. Dodatkowym problemem jest tu brak procedury stabilizowania modelu w trakcie procesu uczenia.

Tab. 5.1: Porównanie wyników dla elementarnych algorytmów uczenia

Algorytm	**Aproksymacja gradientu**	**Parametry**	**Statystyki**		
			mMAPE	σ	E_c
LM	wg (A.29)	$\mathcal{I} = 5$	18	25	88
BFGS	wg (A.30)	$\mathcal{I} = 5,\ m = 4$	19	15	390
DFP	wg (A.29)	$\mathcal{I} = 5,\ m = 4$	25	17	210
GB	wg (A.31)	$\mathcal{I} = 15,\ k = 5,\ m = 4,\ c = 0.1,\ \gamma_c = 2$	27	12	270
BFGS	wg (A.29)	$\mathcal{I} = 5,\ m = 4$	27	14	210
DFP	wg (A.30)	$\mathcal{I} = 5,\ m = 4$	29	14	390
RS-D1	–	$\mathcal{I} = 15,\ k = 5,\ m = 7$	25	19	421

E_c - średnia liczba wyliczeń funkcji celu, $\alpha_1 = 0,\ \alpha_2 = 5,\ \alpha_3 = 10$

Na Rys. 5.2 pokazano uśrednione przebiegi błędu trenowania (SSE) uzyskanego dla danych ze zbioru $\mathcal{L}_T$ i błędu weryfikacyjnego (mMAPE) wyliczanego dla danych ze zbioru $\mathcal{L}_V$.



(a) (b)

Rys. 5.2: Uśrednione przebiegi błędu trenowania SSE i błędu weryfikacyjnego mMAPE oraz odpowiadające im wartości minimalne i maksymalne uzyskane dla algorytmu LM

Na podstawie wyników otrzymanych dla standardowych algorytmów uczenia oraz badań wstępnych autora omówionych w pracy (Przystałka, 2007b) postanowiono, że porównane zostaną wyłącznie trzy schematy uczenia hybrydowego (EA-LM, SA-LM oraz

DS-LM) pod kątem minimalizacji średniego błędu testowania mMAPE oraz jego rozrzutu σ. Dla każdego schematu w drugiej fazie uczenia stosowany był algorytm LM (z jakobianem wyliczanym za pomocą różnic w przód). Faza ta uruchamiana była podobnie jak poprzednio dla maksymalnej liczby iteracji $\mathcal{I} = 5$.

Schemat EA-LM

Ponieważ algorytm ewolucyjny służy do znalezienia początkowego rozwiązania, które powinno znajdować się w pobliżu minimum globalnego, dlatego badania prowadzono dla różnych operatorów mutacji przy jednoczesnym systematycznym przeszukiwaniu wartości prawdopodobieństwa krzyżowania. Przyjęto następujące wartości cech niezmiennych algorytmu ewolucyjnego:

- populacja początkowa uzyskiwana metodą INIT0,
- funkcja przystosowania w postaci (4.25) z $R = 4$, $\lambda = 0$,
- liczebność populacji równa liczbie parametrów swobodnych sieci $L_p = p = 13$,
- selekcja chromosomów metodą ruletki,
- krzyżowanie heurystyczne ($\lambda_h = 1.1$),
- sukcesja elitarna ($\delta_s = 2$).

Dla każdej kombinacji wartości cech algorytmu ewolucyjnego przeprowadzono dziesięć prób. W Tab. 5.2 dla różnych wartości parametrów operatorów reprodukcji zawarto wartości uśrednionego błędu testowania mMAPE oraz ich rozrzut (σ).

Najlepsze wyniki dla tego schematu otrzymano dla mutacji równomiernej z odwzorowaniem Hénona ($r_m = 0.01$) przy prawdopodobieństwie krzyżowania $p_k = 0.6$.

Schemat SA-LM

Dla schematu będącego połączeniem symulowanego wyżarzania i algorytmu Levenberga-Marquardta przyjęto następujące założenia:

- rozwiązanie początkowe uzyskiwane metodą INIT0,
- funkcja celu w postaci (4.25) z $R = 4$, $\lambda = 0$,
- prawdopodobieństwo pogorszenia rozwiązania wyliczane wg zależności (A.14),
- liczba iteracji algorytmu symulowanego wyżarzania równa 100.

Dla tego schematu badano wpływ różnych sposobów generowania nowych parametrów modelu, sposobów redukcji temperatury wyżarzania oraz jej wartości początkowej na średni błąd testowania oraz jego rozrzut (Tab. 5.3). Podobnie jak poprzednio przeprowadzono dziesięć prób dla każdej kombinacji wartości parametrów.

Szczególnie dobre wyniki dla tego algorytmu otrzymano dla funkcji generującej $g_N^2(\boldsymbol{\omega}_n)$ i funkcji redukcji temperatury $t_f(T_n)$ przy $T_0 = 1$. Można zauważyć dodatkowo, że algorytm symulowanego wyżarzania dla małej wartości temperatury początkowej działa podobnie jak algorytm ewolucyjny z mutacją nierównomierną, bez krzyżowania.

Dla temperatury w końcowym etapie strojenia parametrów algorytm symulowanego wyżarzania zachowuje się jak algorytm z losowym wyborem kierunku poszukiwań (z zerowym progiem).

Tab. 5.2: Wybrane wartości uśrednionego błędu testowania mMAPE oraz ich rozrzut (σ) otrzymane dla uczenia wg schematu EA-LM

Parametry mutacji	**Prawdopodobieństwo krzyżowania p_k** 0.00	0.20	0.40	0.60	0.80	1.00
○ **Mutacja nierównomierna z rozkładem Gaussa**						
$\gamma_1 = 1$, $\gamma_2 = 1$	2.03 (1.59)	9.78 (17.32)	1.67 (1.36)	2.61 (2.28)	1.61 (0.71)	2.00 (2.18)
1, 0.50	9.28 (13.68)	9.89 (17.17)	6.90 (13.45)	1.43 (0.82)	6.57 (13.09)	1.91 (1.50)
1, 0.10	75.69 (230.02)	1.58 (0.86)	6.11 (12.66)	5.95 (13.42)	1.44 (0.75)	2.09 (2.79)
1, 0.05	4.50 (9.90)	10.41 (17.17)	$\oplus$ 1.36 (0.75)	5.23 (13.04)	2.00 (1.80)	2.09 (1.62)
1, 0.01	9.96 (24.82)	28.03 (55.25)	5.83 (13.03)	1.55 (1.31)	2.57 (1.63)	2.15 (1.80)
○ **Mutacja nierównomierna z odwzorowaniem Hénona**						
..., ...	...	...	...	...	...	...
1, 0.001	6.32 (12.98)	5.79 (12.44)	1.26 (0.72)	1.57 (0.99)	2.70 (5.51)	2.26 (1.59)
..., ...	...	...	...	...	...	...
○ **Mutacja nierównomierna z odwzorowaniem Ikedy**						
..., ...	...	...	...	...	...	...
1, 0.1	7.35 (12.41)	6.40 (14.09)	1.50 (0.92)	$\odot$ 1.28 (0.90)	5.03 (12.35)	2.40 (1.83)
..., ...	...	...	...	...	...	...
○ **Mutacja nierównomierna z dyskretnym równaniem Mackeya-Glassa**						
..., ...	...	...	...	...	...	...
1, 0.05	3.89 (5.76)	12.20 (22.54)	5.67 (13.18)	1.40 (0.50)	2.15 (1.47)	3.46 (4.47)
..., ...	...	...	...	...	...	...
○ **Mutacja równomierna z rozkładem prostokątnym**						
...	...	...	...	...	...	...
$r_m = 0.1$	1.96 (1.37)	2.08 (1.83)	5.45 (13.65)	1.83 (1.47)	$\ominus$ 0.99 (1.14)	1.69 (1.32)
...	...	...	...	...	...	...
○ **Mutacja równomierna z odwzorowaniem Hénona**						
...	...	...	...	...	...	...
0.01	1.07 (0.82)	1.69 (1.90)	2.16 (1.95)	$\oslash$ 0.95 (0.46)	2.30 (1.24)	1.89 (1.55)
...	...	...	...	...	...	...
○ **Mutacja równomierna z odwzorowaniem Ikedy**						
...	...	...	...	...	...	...
0.01	1.03 (0.49)	14.00 (38.51)	4.66 (7.17)	2.65 (2.21)	2.23 (1.91)	1.44 (1.31)
...	...	...	...	...	...	...
○ **Mutacja równomierna z dyskretnym równaniem Mackeya-Glassa**						
...	...	...	...	...	...	...
0.1	1.82 (0.80)	5.10 (11.74)	1.06 (0.55)	2.21 (3.19)	2.82 (2.51)	6.20 (12.42)
...	...	...	...	...	...	...

Schemat DS-LM

Podobnie jak w poprzednich schematach uczenia i w tym przypadku niezbędne było przeprowadzenie szeregu prób, które umożliwiły uzyskanie satysfakcjonujących wyników. Przyjęto następujące założenia:

- rozwiązanie początkowe uzyskiwane metodą INIT0,
- funkcja celu w postaci (4.25) z $R = 4$, $\lambda = 0$,
- liczba iteracji algorytmu przeszukiwania bezpośredniego równa 10.

Tab. 5.3: Wybrane wartości uśrednionego błędu testowania mMAPE oraz ich rozrzut (σ) otrzymane dla uczenia wg schematu SA-LM

Funkcja generująca	**Redukcja temp.**	**Temperatura początkowa T_0** 1	2	3	4	5	...
...	...	...	...	...	...	...	...
g_N^1	t_e	1.90 (1.56)	5.82 (12.21)	24.25 (20.80)	93.06 (164.25)	35.04 (40.29)	...
g_N^1	t_f	$\boxed{\boxdot\ 1.50\ (0.80)}$	5.67 (12.52)	5.40 (13.41)	5.55 (12.88)	14.30 (19.67)	...
g_N^1	t_b	7.11 (13.69)	11.67 (21.40)	59.01 (136.44)	24.20 (20.51)	90.65 (190.62)	...
g_N^2	t_e	5.81 (11.32)	24.18 (24.87)	38.28 (76.23)	6.02 (12.85)	30.47 (19.85)	...
g_N^2	t_f	$\boxed{\boxplus\ 1.11\ (0.59)}$	5.89 (12.73)	21.58 (50.01)	5.89 (12.94)	14.02 (19.93)	...
g_N^2	t_b	$\boxed{\boxminus\ 1.51\ (0.85)}$	5.36 (12.73)	18.25 (20.92)	8.54 (13.51)	8.53 (15.10)	...
g_H^2	t_e	2.41 (1.82)	6.35 (12.48)	2.51 (1.10)	19.78 (19.17)	22.87 (21.68)	...
g_H^2	t_f	2.05 (1.23)	9.07 (15.38)	14.26 (19.75)	$\boxed{1.71\ (0.70)}$	18.15 (20.96)	...
...	...	...	...	...	...	...	...

Badania schematu prowadzono dla różnych kombinacji metod tworzenia oraz przeszukiwania lokalnego siatki przy jednoczesnych zmianach mnożnika kontrakcji i ekspansji siatki. W Tab. 5.4 pokazano wpływ tych parametrów na średni błąd testowania oraz jego rozrzut (wyniki uśrednione dla dziesięciu prób).

Tab. 5.4: Wybrane wartości uśrednionego błędu testowania mMAPE oraz ich rozrzut (σ) otrzymane dla uczenia wg schematu DS-LM

Faza pierwsza	**Faza druga**	**Mnożnik kontrakcji | ekspansji siatki** 0.5|2	0.16|6	0.1|10	0.07|14	0.05|18
...	...	...	...	...	...	...
GPS2P	MADSPp1	2.09 (1.20)	2.13 (1.26)	$\boxed{\ltimes\ 1.95\ (0.80)}$	6.26 (13.42)	2.00 (1.11)
–	MADSPp1	13.97 (19.61)	6.44 (12.99)	15.18 (15.18)	5.77 (12.98)	$\boxed{\rtimes\ 1.59\ (0.74)}$
MADS2P	MADSPp1	4.79 (9.79)	39.75 (83.46)	2.46 (1.74)	5.89 (13.86)	10.05 (17.60)
MADSPp1	MADS2P	1.90 (1.07)	6.13 (14.11)	13.00 (18.34)	6.71 (15.90)	$\boxed{\diamond\ 1.63\ (0.64)}$
–	MADS2P	1.91 (1.01)	2.57 (1.69)	14.37 (19.90)	5.88 (13.16)	50.09 (139.69)
...	...	...	...	...	...	...

Jak widać schemat ten pozwala na uzyskanie błędów testowania oraz ich rozrzut na podobnym poziomie jak w przypadku EA-LM i SA-LM. Ze względu na dużą prostotę algorytmu wydaje się być on ciekawą alternatywą dla poprzednich algorytmów.

Podsumowanie

W Tab. 5.5 syntetycznie przedstawiono najlepsze wyniki uzyskane dla wybranych wariantów schematów uczących. Dokonano porównania wartości średnich błędów testowania,

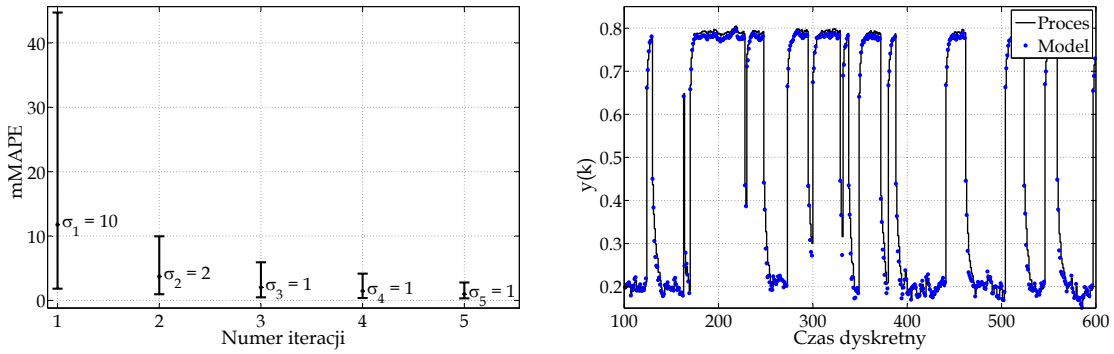
ich rozrzutu oraz średniej liczby wyliczeń funkcji celu $E_c()$ potrzebnych na każdym z etapów uczenia. Najniższy średnią wartość błądu testowania przy minimalnym rozrzucie otrzymano dla algorytmu hybrydowego zrealizowanego wg schematu EA-LM z mutacją chaotyczną (mutacja równomierna z odwzorowaniem Hénona).

Na Rys. 5.3 przedstawiono uśrednione przebiegi błędu weryfikacyjnego mMAPE oraz wartości maksymalne i minimalne jakie otrzymano dla tego wariantu (rysunek dotyczy drugiej fazy treningu). Wyraźnie widać poprawę w stosunku do uczenia realizowanego wyłącznie za pomocą algorytmu LM (Rys. 5.2).

Tab. 5.5: Zbiorcze porównanie wyników uczenia dla rozpatrywanych schematów

Schemat	**Oznaczenie**	**Statystyki**		
		mMAPE	σ	$E_c(\mathrm{I}) \mid E_c(\mathrm{II})$
EA-LM	$\oslash$	0.95	0.46	143 | 88
EA-LM	$\ominus$	0.99	1.14	143 | 88
EA-LM	$\odot$	1.28	0.90	143 | 88
EA-LM	$\oplus$	1.36	0.75	143 | 88
SA-LM	$\boxplus$	1.11	0.59	154 | 88
SA-LM	$\boxdot$	1.50	0.80	195 | 88
SA-LM	$\boxminus$	1.51	0.85	135 | 88
DS-LM	$\rtimes$	1.59	0.74	246 | 88
DS-LM	$\diamond$	1.63	0.64	381 | 88
DS-LM	$\ltimes$	1.95	0.80	80 | 88

Druga część Rys. 5.2 pokazuje etap testowania modelu, dla którego otrzymano wartość błędu $\mathrm{mMAPE} = 0.65$. Pozostałe schematy charakteryzowały się wartością średniego błędu i rozrzutem na podobnym poziomie, wymagały jednak zazwyczaj większej liczby wyliczeń funkcji błędu. Autor prowadził również badania dla innych kombinacji metod elementarnych (Przystałka, 2007b). Niemniej jednak najlepsze wyniki otrzymał dla schematów opisanych powyżej.



(a) (b)

Rys. 5.3: Uśrednione przebiegi błędu weryfikacyjnego mMAPE (a) oraz przykładowe wyniki testowania modelu (b) otrzymane dla schematu EA-LM

Prezentowane wyniki potwierdzają, że przeprowadzenie procesu uczenia według opracowanych schematów pozwala na minimalizację błędów testowania oraz zmniejszenie prawdopodobieństwa utknięcia algorytmu lokalnego w minimum lokalnym.

5.3.3. Identyfikacja wielowymiarowego systemu dynamicznego

Przydatność opracowanych struktur neuronowych w zadaniach modelowania wielowymiarowych obiektów dynamicznych sprawdzono na przykładzie identyfikacji nieliniowego systemu dynamicznego klasy MIMO zapisanego w następującej postaci:

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} = \begin{bmatrix} y_1(k-1) - 0.1y_1(k-2) \\ y_1(k-1) - 0.3y_1(k-1)\, y_1^2(k-2) \end{bmatrix} + \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}. \tag{5.1}$$

W celu przygotowania wzorców trenujących i testowych na wejście systemu zadano wymuszenie $\{\mathbf{u}(k)\}_{k=1}^{3000}$, którego elementy składowe stanowią sygnały niezależne w postaci pseudolosowych ciągów o wartościach $\{-1, 1\}$. Przyjęto prawdopodobieństwo zmiany poziomu sygnałów $\alpha_{\mathrm{PBS}} = 0.90$ oraz zerowe warunki początkowe.

Pierwszych 1500 próbek sygnałów wejściowych i wyjściowych stanowiło przykłady trenujące. Pozostałe próbki sygnałów tworzyły zbiór testowy. Dla rozpatrywanego problemu utworzono modele neuronowe typu MIMO oraz MISO, których struktury dobrano stosując kryterium AIC oraz zasadę oszczędności. Ograniczono się wyłącznie do struktur lokalnie rekurencyjnych z jedną warstwą ukrytą.

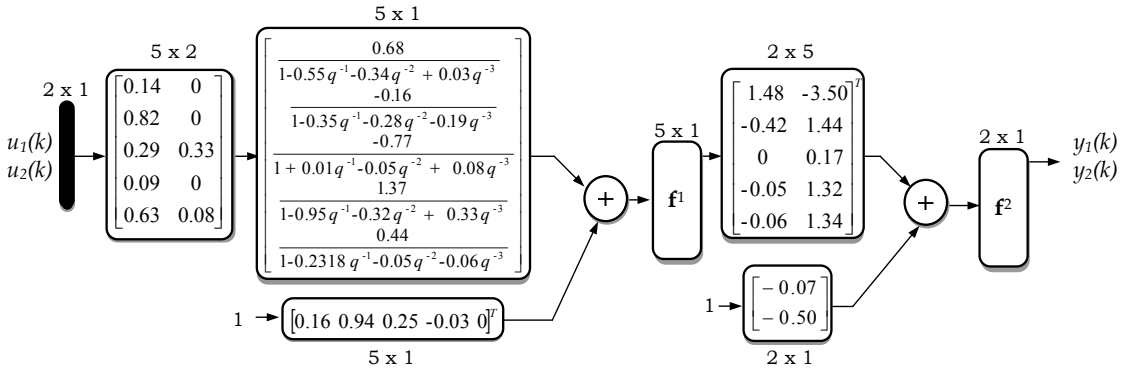
Każdy model neuronowy dostrajano z zastosowaniem algorytmu hybrydowego uczenia według schematu EA-LM. Przyjęto następujące wartości parametrów funkcji celu (4.25) dla algorytmu globalnego: $R = 2$, $\lambda = 0.05$, $\omega_0 = 1.5$ oraz dla algorytmu lokalnego: $R = 2$, $\lambda = 0$.

W pierwszej fazie treningu stosowano algorytm ewolucyjny z maksymalną liczbą epok równą 20, z krzyżowaniem heurystycznym ($\lambda_h = 1.1$, $p_k = 0.8$), z mutacją równomierną z odwzorowaniem Ikedy ($r_m = 0.1$), stałą liczebnością populacji równą 50 osobników (przy czym populacja bazowa uzyskiwana była metodą INIT1 - patrz rozdz. B.5), selekcją metodą ruletki oraz sukcesją elitarną ($\delta_s = 1$). W kolejnej fazie procesu dostrajania modeli stosowano algorytm LM (15 iteracji) z numerycznie wyznaczanym jakobianem. Wyniki modelowania zaprezentowano w Tab. 5.6.

Tab. 5.6: Wskaźniki jakości modeli neuronowych rozpatrywanego systemu

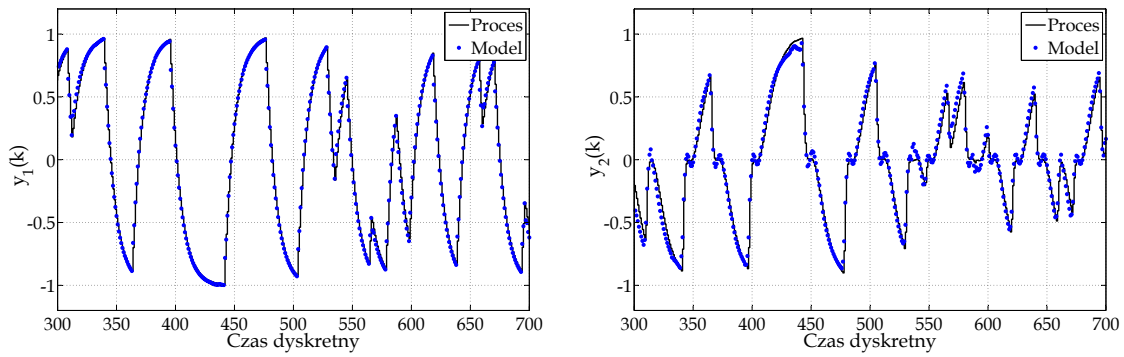
Nr	Model	Struktura	p	mMAPE	mMAPE$_1$	mMAPE$_2$	AIC
1	$(y_1(k), y_2(k)) = f(u_1(k), u_2(k))$	$2 \to 5_{(0)}^{(3,1,0)} \to 2$	47	1.630	1.040	2.221	827
2	$y_1(k) = f_1(u_1(k), u_2(k))$	$1 \to 5_{(0)}^{(3,1,0)} \to 1$	41	–	0.397	–	−1301
3	$y_2(k) = f_2(u_1(k), u_2(k))$	$1 \to 5_{(0)}^{(3,1,0)} \to 1$	41	–	–	6.941	2986

Jako wskaźniki jakości rozpatrywanych modeli przyjęto średni błąd mMAPE oraz błąd $\mathrm{mMAPE}_{1/2}$ wyznaczany dla wyjścia $1/2$. Podczas identyfikacji systemu najlepsze rezultaty uzyskano dla modelu lokalnie rekurencyjnego typu MIMO z neuronami warstwy ukrytej z filtrami o nieskończonej odpowiedzi impulsowej w bloku aktywacyjnym. Strukturę tego modelu oraz uzyskane parametry pokazano na Rys. 5.4 (f^1 - tangens hiperboliczny - $\alpha_f = 1$, f^2 - funkcja liniowa).



Rys. 5.4: Model neuronowy nr 1

Otrzymane wyniki pozwalają stwierdzić, że w przypadku identyfikacji modeli lokalnych rozpatrywanego systemu nie jest możliwe tak wierne odtworzenie dynamiki modelowanego obiektu, jak w przypadku modelu klasy MIMO. Jest to spowodowane tym, że wartość wyjścia nr 2 systemu zależy od aktualnej wartości sygnału sterowania, jak również od wartości sygnału wyjścia nr 1 systemu z poprzednich chwil czasu. Na Rys. 5.5 przedstawiono odpowiedzi systemu i modelu dla danych testowych.

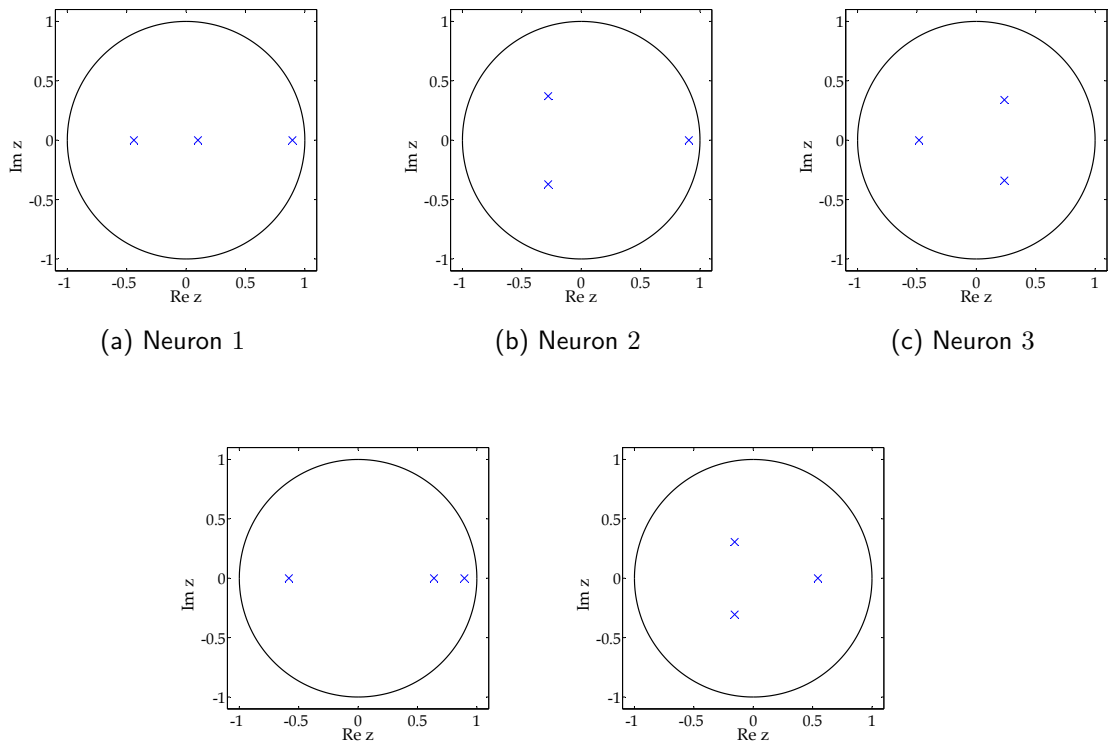


(a) (b)

Rys. 5.5: Wyjścia systemu (5.1) i modelu neuronowego nr 1 uzyskane dla danych testowych

Dla modelu z Rys. 5.4 przeprowadzono ponadto analizę stabilności, która polegała na zbadaniu położenia biegunów transmitancji dyskretnych części dynamicznej wszystkich neuronów warstwy ukrytej. Wyniki obliczeń pokazano na Rys. 5.6.

Jak widać, wszystkie bieguny transmitancji znajdują się wewnątrz koła jednostkowego, co zapewnia stabilność asymptotyczną uzyskanego modelu.



(a) Neuron 1 (b) Neuron 2 (c) Neuron 3

(d) Neuron 4 (e) Neuron 5

Rys. 5.6: Bieguny transmitancji dyskretnych poszczególnych neuronów warstwy ukrytej modelu nr 1

5.3.4. Identyfikacja układu chaotycznego

Bardzo często podczas budowy modelu procesu dostępna jest wyłącznie ograniczona liczba zmiennych procesowych. Dynamikę obiektu w takim przypadku można odtworzyć przez odpowiednie przekształcenie mierzonych zmiennych. Skuteczność metodyki w zastosowaniu do tego typu zadania sprawdzono na przykładzie danych opisujących zachowanie wybranego układu chaotycznego, którym był oscylator Duffinga z sygnałem wymuszenia (Li *i in.*, 2006; Li, 2006):

$$\mathcal{R}_D = \begin{cases} \text{Jeżeli } x_1(t) \text{ jest } M_1, \text{ to } \dot{\mathbf{x}}(t) = \mathbf{A}_1\mathbf{x}(t) + \mathbf{B}_1 u(t) + \mathbf{E}d(t), \\ \text{Jeżeli } x_1(t) \text{ jest } M_2, \text{ to } \dot{\mathbf{x}}(t) = \mathbf{A}_2\mathbf{x}(t) + \mathbf{B}_2 u(t) + \mathbf{E}d(t), \end{cases} \tag{5.2}$$

gdzie

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 \\ 0 & -0.1 \end{bmatrix}, \quad \mathbf{A}_2 = \begin{bmatrix} 0 & 1 \\ -v^2 & -0.1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{E} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

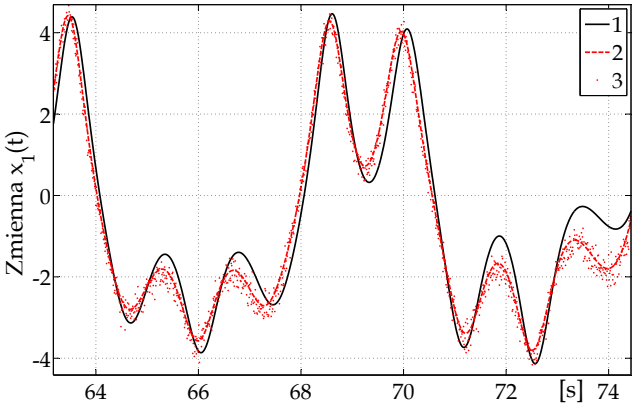
oraz

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) & x_2(t) \end{bmatrix}^T, \quad u(t) = 24\cos(t), \quad M_1(x_1) = 1 - \tfrac{x_1}{v^2}, \quad M_2(x_1) = \tfrac{x_1}{v^2},$$

przy czym $d(t)$ reprezentuje oddziaływanie niemierzalne zrealizowane w postaci szumu gausowskiego poddanego operacji wydłużenia okresu (Söderström i Stoica, 1997). Zadanie to sprowadza się do budowy predykcyjnego modelu neuronowego na podstawie danych pochodzących z rozwiązania układu zapisanego w formie systemu rozmytego (5.2). Układ wykazuje zachowanie nieregularne w czasie dla $v = 50$.

Szeregi czasowe opisujące zachowanie obiektu uzyskano, rozwiązując powyższe równania z zastosowaniem metody Runge-Kutty czwartego rzędu (stały odstęp całkowania równy 0.01, czas symulacji 100 s). Na Rys. 5.7 pokazano różne warianty rozwiązania układu równań.

Przebieg nr 1 przedstawia układ bez wpływu zakłóceń. Drugi przebieg reprezentuje układ, na który oddziaływuje zakłócenie $d(t)$. Przebieg trzeci jest wynikiem symulacji układu z zakłóceniami niemierzalnymi oraz zakłóceniem reprezentującym szumy pomiarowe (SNR=26dB). Poniższe rozważania dotyczą budowy modelu dla trzeciego wariantu rozwiązania.



Rys. 5.7: Przebiegi zmiennej $x_1(t)$ w zależności od różnego rodzaju zakłóceń

W celu przygotowania przykładów uczących zwiększono odstęp próbkowania, wybierając co czwartą próbkę szeregu czasowego (poprzez decymację). W ten sposób uzyskano przebiegi czasowe zmiennych stanu o liczbie próbek równej 2500. Przyjmując ograniczenie, że znana jest wyłącznie składowa stanu $x_1(t)$, dynamikę układu można odtworzyć, stosując metodę współrzędnych z opóźnieniem. Problem ten można sformułować następująco: znając wartości kilku kolejnych elementów szeregu czasowego do dyskretnej chwili k, należy sporządzić prognozę kolejnej wartości z horyzontem H:

$$\langle x_1\left(k-(d-1)\tau\right), \ldots, x_1\left(k-\tau\right), x_1\left(k\right)\rangle \rightarrow x_1\left(k+H\right),$$

przy czym wymiar zanurzenia $d = 3$ wyznaczony został metodą fałszywych sąsiadów, a opóźnienie $\tau = 10$ za pomocą informacji wzajemnej, dzięki czemu otrzymano:

$$\langle x_1\left(k-20\right), x_1\left(k-10\right), x_1\left(k\right)\rangle \rightarrow x_1\left(k+10\right).$$

Pierwszych 1500 próbek posłużyło do trenowania modeli neuronowych. Kolejnych 970 wartości szeregu czasowego zastosowano w procesie testowania modeli. Rozpa-

trzono następujące klasy modeli: struktury lokalnie rekurencyjne opracowane przez autora (dwuwarstwowe oznaczone w Tab. 5.7 przez ■, trójwarstwowe oznaczone przez •) oraz wielokontekstowe sieci Jordana (oznaczone przez ×) i Elmana (oznaczone przez ►). Ograniczono się jedynie do struktur złożonych z neuronów z funkcjami wyjścia typu tangens hiperboliczny ($\alpha_f = 1$) w warstwach ukrytych i typu funkcja liniowa w warstwie wyjściowej.

Podczas dostrajania modeli neuronowych stosowano algorytm hybrydowy według schematu EA-LM. Przyjęto następujące cechy schematu:

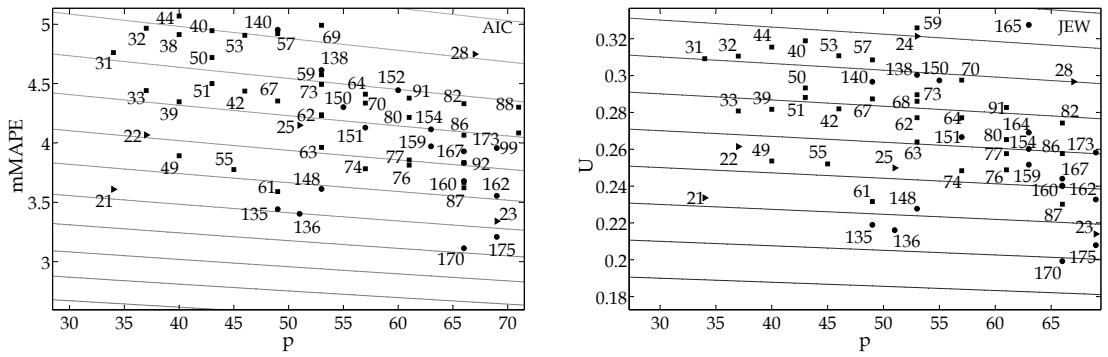
- populacja początkowa uzyskana metodą INIT0,
- funkcja przystosowania w postaci (4.25) z $R = 4$, $\lambda = 0$, $\omega_0 = 1.5$,
- liczebność populacji równa 50 osobników,
- selekcja chromosomów metodą ruletki,
- krzyżowanie heurystyczne ($\lambda_h = 1.1$, $p_k = 0.6$),
- mutacja równomierna z rozkładem prostokątnym ($r_m = 0.1$),
- sukcesja elitarna ($\delta_s = 1$),
- funkcja celu algorytmu LM w postaci sumy kwadratów różnic,
- maksymalna liczba epok algorytmu ewolucyjnego równa 10,
- maksymalna liczba iteracji algorytmu LM równa 10.

Uzyskane wartości błędu mMAPE oraz U (wyznaczane dla prognozy naiwnej) dla wybranych struktur neuropodobnych oraz odpowiadające im wartości funkcji kryterialnych (AIC i JEW) zamieszczono w Tab. 5.7. Dodatkowo na Rys. 5.8 przedstawiono punkty odpowiadające poszczególnym badanym modelom neuronowym.

Tab. 5.7: Oceny błędów wybranych modeli neuronowych zmiennej $x_1(t)$

$x_1(k+10) = f\left(x_1(k), x_1(k-10), x_1(k-20)\right)$							
Nr	**Oznaczenie**	**Struktura**	**p**	**mMAPE**	**U**	**AIC**	**JEW**
170	•	$3 \to 4 \to 3^{(3,3,0)}_{(0,2,0)} \to 1$	66	3.113	0.199	1233	3.362
175	•	$3 \to 4 \to 4^{(2,2,0)}_{(0,1,0)} \to 1$	69	3.209	0.207	1268	3.479
136	•	$3 \to 4 \to 2^{(2,3,1)}_{(0,2,0)} \to 1$	51	3.404	0.216	1290	3.604
87	■	$5 \to 5^{(2,3,1)}_{(0,1,0)} \to 1$	66	3.643	0.230	1386	3.931
61	■	$5 \to 4^{(2,3,0)}_{(0,1,0)} \to 1$	49	3.590	0.231	1337	3.796
21	►	$5 \to 3^{(1,2)} \to 1$	34	3.609	0.233	1434	3.749
10	×	$3 \to 5^{(1)} \to 1$	31	8.107	0.455	2091	8.393
...	...	...	...	...			

Takie podejście pozwoliło rozwiązać problem kompromisu pomiędzy błędem działania modelu a jego złożonością. Ponadto możliwe było porównanie modeli neuronowych różnych typów.

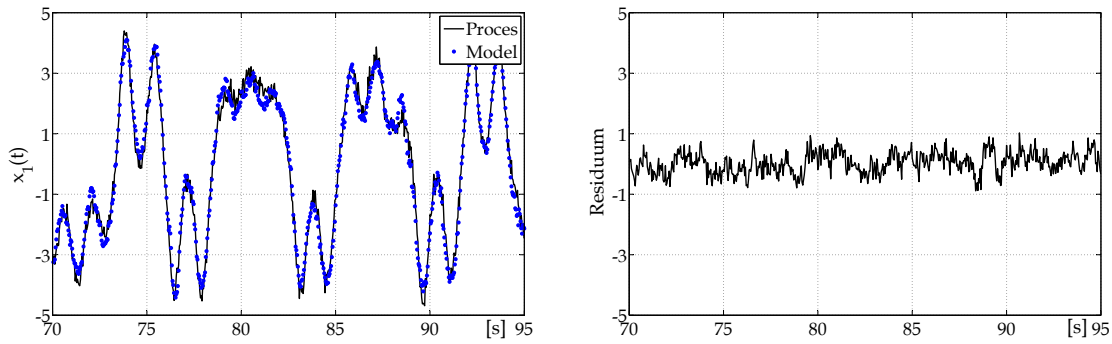


(a) (b)

Rys. 5.8: Linie izokryterialne: wyniki testowania uzyskane dla wybranych modeli neuronowych dla kryteriów: (a) AIC i (b) Jenkinsa-Wattsa

Z Tab. 5.7 i Rys.5.8 wynika, że wartości funkcji kryterialnych AIC oraz JEW są najmniejsze dla modelu nr 170. Jak można zauważyć, zadowalające wyniki otrzymano również dla sieci Elmana. Przewagę struktur lokalnie rekurencyjnych i sieci Elmana nad strukturami Jordana w tego typu zadaniu tłumaczy fakt, że struktury te posiadają lepsze możliwości pamiętania stanów wewnętrznych. Z kolei struktura Jordana lepiej zapamiętuje stany wyjściowe.

W celu zobrazowania jakości otrzymanego modelu, na Rys. 5.9(a) pokazano przebieg zmiennej $x_1(t)$ oscylatora Duffinga oraz wyjścia z modelu nr 170 odtwarzającego dynamikę układu. Rys. 5.9(b) przedstawia wartości residuum obliczone dla sygnałów uzyskanych odpowiednio z układu i modelu.



(a) (b)

Rys. 5.9: Faza testowania modelu neuronowego nr 170

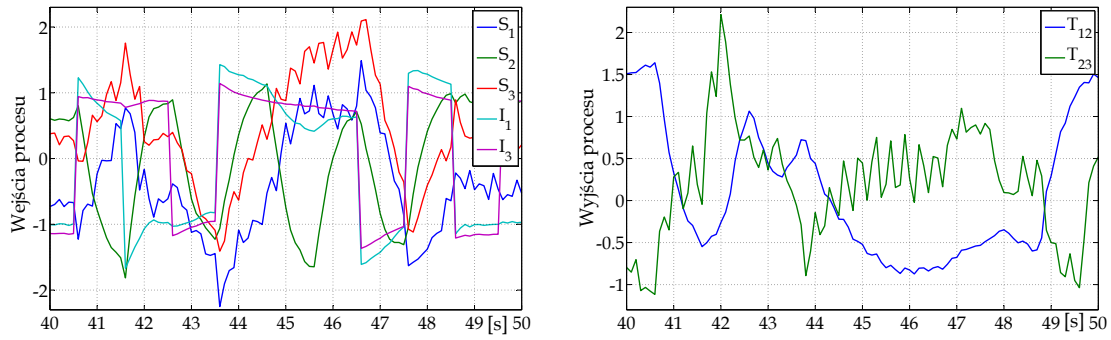
Dynamikę układu próbowano również odtworzyć z zastosowaniem wielowymiarowych modeli ARX oraz modeli neuronowych typu NNARX. Dla obu klas modeli dobór struktury realizowano metodą prób i błędów, stosując kryteria AIC oraz JEW. Dla modeli typu ARX nie udało się uzyskać wartości błędów mMAPE poniżej 10%. Modele NNARX pozwalały na uzyskanie wartości błędów mMAPE na poziomie 5%.

5.3.5. Modelowanie procesu rzeczywistego

W poprzednich przykładach pokazano przydatność metodyki w zastosowaniu do danych, które uzyskano w wyniku symulacji modeli matematycznych wybranych układów dynamicznych. Aby wykazać praktyczną użyteczność zaproponowanego podejścia, przeprowadzono testy z użyciem danych, które pozyskano na pilotażowym obiekcie realizującym proces zwijania tkaniny (De Moor, 2009; Noura i Bastogne, 1997). Stanowią one część obszernego zbioru danych zgromadzonych przez E. Keogha i współpracowników dla celów weryfikacji i badania różnych metod odkrywania wiedzy (Keogh *i in.*, 2009).

Obiekt identyfikacji złożony jest z trzech rolek (rozwijającej, pociągowej i nawijającej), które napędzane są przez odpowiednio sprzęgnięte silniki prądu stałego z przekładniami redukującymi. Wartościami mierzonymi są: prędkości obrotowe rolek - S_1, S_2, S_3 [rad/s], naciąg pomiędzy rolkami 1 i 2 oraz 2 i 3 - T_{12}, T_{23} [N], wartości zadane prądów silników 1 i 3 - I_1 i I_3 [A]. Każdy z silników sterowany jest w pętli sprzężenia zwrotnego z zastosowaniem regulatorów PI. Wartości zadane (prądu i prędkości obrotowej) generowane są przez odpowiedni sterownik PLC w celu utrzymania zadanej prędkości i naprężenia materiału.

Na Rys. 5.12 pokazano przebiegi rejestrowanych zmiennych procesowych. Odstęp pomiędzy kolejnymi próbkami wynosi 0.1 [s]. Dane obejmują około cztery minuty realizacji procesu (2500 próbek). Dostępne sygnały wejściowe i wyjściowe podzielono na zbiory danych trenujących oraz testowych w proporcjach odpowiednio 60% i 40%.



(a) (b)

Rys. 5.10: Znormalizowane przebiegi zmiennych procesowych

W pierwszej fazie eksperymentu identyfikacyjnego utworzono dwa wzorcowe modele procesu: model parametryczny ARX oraz model neuronowy typu NNARX. Strukturę pierwszego modelu dobierano metodą prób i błędów z zastosowaniem kryterium AIC oraz zasady oszczędności. Otrzymano model ARX (Rys. 5.11 - model nr 226) o strukturze NA $= [3\ 3;\ 3\ 3]$, NB $= [3\ 3\ 3\ 3\ 3;\ 3\ 3\ 3\ 3\ 3]$ oraz NK $= [0\ 0\ 0\ 0\ 0;\ 0\ 0\ 0\ 0\ 0]$, którego parametry dostrojono z zastosowaniem metody najmniejszych kwadratów. Dla tego modelu uzyskano wartość błędu testowania $\mathrm{mMAPE} \approx 4\%$.

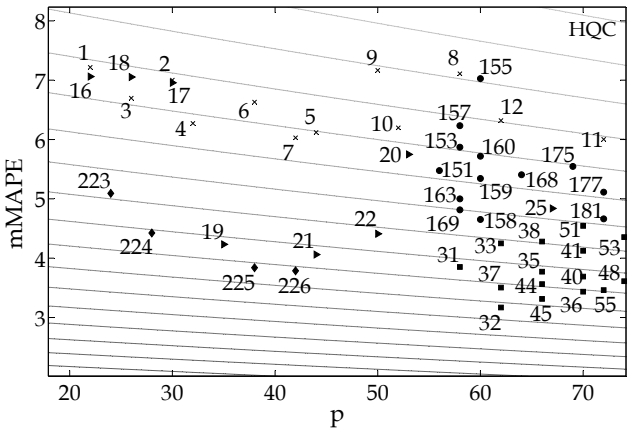
Strukturę drugiego modelu dobierano podobnie jak poprzednio, stosując metodę prób i błędów. Najlepsze wyniki otrzymano dla modelu $5 \rightarrow^{(0,1)}_{(1,2)} 6 \rightarrow 2$ z funkcjami tangen-

soidalnymi neuronów warstwy ukrytej i liniowymi warstwy wyjściowej. Parametry tego modelu dostrajano algorytmem LM z regularyzacją Bayesa, uzyskując wartość błędu testowania $\mathrm{mMAPE} \approx 3.7\%$.

Następnym krokiem było utworzenie zbioru modeli lokalnie rekurencyjnych (■,●), modeli wielokontekstowych Jordana (×) i Elmana (▸). Przyjęto ograniczenie, że w warstwach ukrytych neurony posiadają funkcje wyjścia w postaci $\tanh(x)$, a w warstwie wyjściowej funkcję wyjścia liniową. Podczas uczenia zastosowano algorytm hybrydowy według schematu EA-LM, przy czym przyjęto następujące wartości parametrów funkcji celu (4.25) dla algorytmu globalnego: $R = 4$, $\lambda = 0.5$, $\omega_0 = 1.5$ oraz dla algorytmu lokalnego: $R = 2$, $\lambda = 0$. W pierwszym etapie treningu stosowano algorytm ewolucyjny (10 epok) z kodowaniem rzeczywisto-liczbowym chromosomów, z krzyżowaniem heurystycznym ($\lambda_h = 1.1$, $p_k = 0.8$), z mutacją równomierną z odwzorowaniem Hénona ($r_m = 0.1$), stałą liczebnością populacji równą 50 osobników (populacja bazowa uzyskiwana metodą INIT1), selekcją proporcjonalną oraz sukcesją elitarną ($\delta_s = 1$). W kolejnym etapie procesu dostrajania modeli stosowano algorytm Levenberga-Marquardta (15 iteracji) z numerycznie wyznaczanym jakobianem. Uzyskane wyniki zaprezentowano w Tab. 5.8 i na Rys. 5.11.

Tab. 5.8: Oceny poprawności działania wybranych modeli neuronowych procesu

$(T_{12}, T_{23}) = f(S_1, S_2, S_3, I_1, I_3)$					
Nr	**Oznaczenie**	**Struktura**	**p**	**mMAPE**	**HQC**
32	■	$5 \to 4^{(2,2,0)}_{(0,2,0)} \to 2$	62	3.168	−5514
45	■	$5 \to 4^{(3,2,1)}_{(0,1,0)} \to 2$	66	3.313	−5454
21	▸	$5 \to 3^{(1,2)} \to 2$	44	4.060	−5336
4	×	$5 \to 3^{(1)} \to 2$	32	6.272	−4947
...	...	...	...	...	



Rys. 5.11: Linie izokryterialne: wyniki testowania osiągnięte przez wybrane modele neuronowe oraz modele ARX dla kryterium Hannana-Quinna

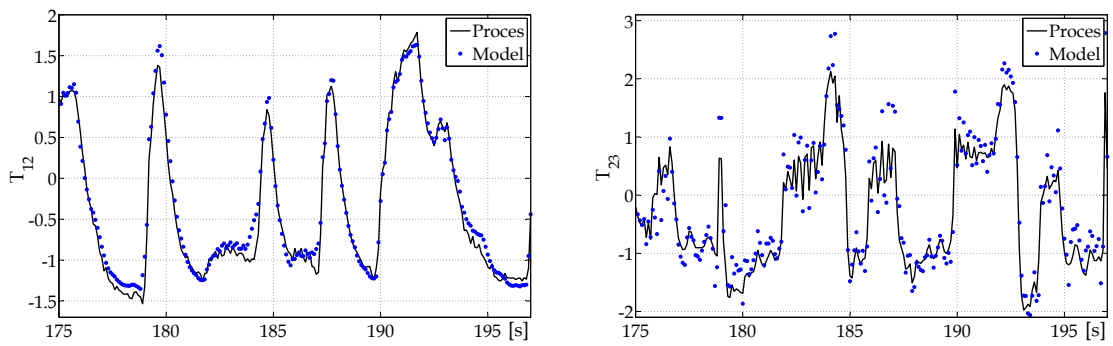
Na podstawie analizy Rys. 5.11 możliwe było wybranie modelu neuronowego najbardziej obiecującego. Dla tego przypadku wybrano model o numerze 32. Kolejnym etapem modelowania procesu była optymalizacja poszczególnych połączeń modelu z zastosowaniem metody OBD. Po usunięciu (wyzerowaniu) parametrów nieistotnych przeprowadzono proces douczania modelu wyłącznie z użyciem algorytmu Levenberga-Marquardta (10 iteracji).

Wyniki testowania modelu przed i po etapie przycinania sieci zawarto w Tab. 5.9. Wyznaczone wartości względnego błędu predykcji RPE (w odniesieniu do modeli wzorcowych ARX i NNARX) umożliwiają obiektywną ocenę jakości odwzorowania dynamiki obiektu. Dodatkowo miara p/p_w określa stosunek złożoności uzyskanego modelu do złożoności modelu wzorcowego (ARX lub NNARX).

Tab. 5.9: Oceny poprawności działania modelu neuronowego - przed i po procesie przycinania jego struktury

Model	**p**	**mMAPE**	**nRMSE**	**l^2**	**RPE(ARX)**	**p/p_w(ARX)**	**RPE(NNARX)**	**p/p_w(NNARX)**
				przed przycinaniem				
32	62	3.169	0.249	0.134	0.796	1.409	0.848	0.596
				po przycinaniu				
...	...	...	...	...	...	...	...	...
32_{58}	58	3.586	0.266	0.158	0.901	1.318	0.959	0.557
32_{57}	57	4.489	0.326	0.220	1.128	1.295	1.201	0.548

Z Rys. 5.12(a) i (b) oraz z Tab. 5.9 wynika, że zadowalające odwzorowywanie dynamiki obiektu możliwe było z użyciem struktury lokalnie rekurencyjnej o dużo mniejszej złożoności niż w przypadku sieci typu NNARX.



(a) (b)

Rys. 5.12: Wyjścia procesu i modelu neuronowego 32_{58}

5.3.6. Modelowanie złożonego procesu przemysłowego

Kolejny przykład dotyczy zastosowania opracowanej metodyki do modelowania zjawisk zachodzących podczas redukcji miedzi z żużla zawiesinowego. Proces ten realizowany jest przez piec elektryczny do redukcji miedzi z żużla. Piec ten jest istotnym obiektem w ciągu produkcyjnym miedzi (Szulim i Moczulski, 2004). System SCADA podczas monitorowania obiektu rejestruje szereg wielkości fizycznych takich jak: prądy i napięcia, ciśnienia, przepływy, temperatury. Dodatkowo zapisywane są różnego rodzaju informacje dotyczące analizy składu chemicznego żużla zawiesinowego, pozycji elektrod zanurzonych w żużlu, masy wsadu itp. Łącznie gromadzonych jest około 180 zmiennych procesowych. Jednym z ważniejszych parametrów diagnostycznych, które podlegają monitorowaniu, jest temperatura.

Zadanie, które postanowiono tu rozwiązać, dotyczyło budowy modelu neuronowego temperatury sklepienia pieca pomiędzy elektrodami E1 i E2. Pierwszym etapem budowy modelu było wybranie zmiennych procesowych mających największy wpływ na zmiany wartości tej temperatury. Wstępnie na podstawie dokumentacji technicznej dotyczącej procesu oraz po zasięgnięciu opinii personelu prowadzącego obiekt wyselekcjonowano pewien podzbiór zmiennych procesowych (Tab. 5.10), mogących mieć istotny wpływ na wartości modelowanej temperatury.

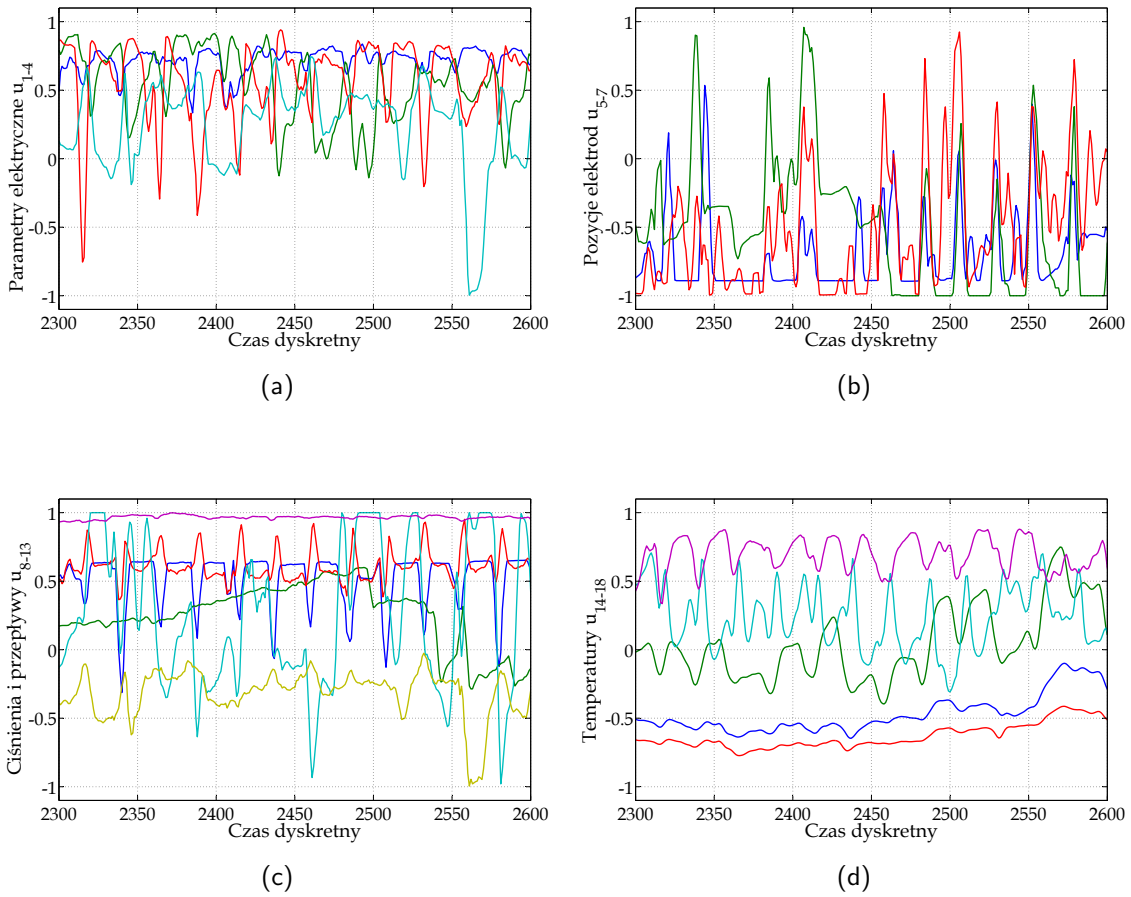
Tab. 5.10: Zestawienie wstępnie wyselekcjonowanych zmiennych procesowych

Zmienna	Jednostka	Opis zmiennej procesowej
y	°C	Temperatura sklepienia pieca pomiędzy elektrodami E1 i E2.
u_1	A	Prąd elektrody E1
u_2	A	Prąd elektrody E2
u_3	A	Prąd elektrody E3
u_4	A	Prąd wentylatora nadmuchu do komory dopalania
u_5	m	Pozycja elektrody E1
u_6	m	Pozycja elektrody E2
u_7	m	Pozycja elektrody E3
u_8	Pa	Ciśnienie w piecu elektrycznym
u_9	MPa	Ciśnienie wody na odpływie z płyt kontaktowych elektrod E2 i E1
u_{10}	Pa	Ciśnienie w komorze dopalania
u_{11}	Pa	Ciśnienie przed chłodnicami
u_{12}	m^3/h	Przepływ wody do kolektora nr 2, chłodzenie kesonów E1/E2
u_{13}	m^3/h	Ilość powietrza przed pierwszą sekcją komory dopalania
u_{14}	°C	Temperatura blachy trzonu elektrody E1
u_{15}	°C	Temperatura blachy trzonu elektrody E2
u_{16}	°C	Temperatura blachy trzonu elektrody E3
u_{17}	°C	Temperatura sklepienia pieca pomiędzy elektrodami E2 i E3
u_{18}	°C	Temperatura sklepienia pieca pomiędzy elektrodami E1 i E3

Dane trenujące i testowe utworzono w następujący sposób. Rozpatrywany zbiór danych obejmował około 36 godzin pracy obiektu. Zmienne procesowe rejestrowane były przez system SCADA z krokiem czasowym $\Delta t = 1$ min. Tak zgromadzone dane przetworzono z zastosowaniem filtru cyfrowego o skończonej odpowiedzi impulsowej 40. rzędu. Uzyskane przebiegi poddano kolejno operacji decymacji wybierając co 20. próbkę sygnału oraz operacji skalowania do przedziału $[-1, 1]$.

W wyniku tych działań otrzymano szeregi czasowe zmiennych procesowych o kroku

czasowym $\Delta t = 20$ min. i liczbie próbek równej 2736. Pierwszych 1500 próbek każdego sygnału tworzyło zbiór trenujący. Pozostałe próbki zastosowano na etapie testowania modeli. Przykładowe przebiegi zmiennych niezależnych u_{1-18} pokazano na Rys. 5.13 (a)-(d).



Rys. 5.13: Przebiegi zmiennych niezależnych

W kolejnym kroku przyjęto założenie, że modelowana temperatura w głównej mierze zależy od prądów elektrod u_{1-3} oraz temperatur u_{14-18}. Wpływ pozostałych zmiennych procesowych wyznaczono, stosując metodę wskaźników informacyjnych (MWI) oraz test Gamma (Γ). Wyniki obliczeń zapisane są w Tab. 5.11.

Biorąc pod uwagę wszystkie wyniki testów zamieszczone w Tab. 5.11 stwierdzono, że najczęściej wskazywanymi zmiennymi, które mają wpływ na temperaturę sklepienia pieca pomiędzy elektrodami E1 i E2 są: pozycje elektrod (u_5, u_6, u_7), ciśnienie w piecu elektrycznym (u_8) oraz ciśnienie wody na odpływie z płyt kontaktowych (u_9).

Dla wszystkich wariantów zmiennych niezależnych utworzono predykcyjne modele neuronowe o stałej strukturze wewnętrznej (dla horyzontów prognozy równych 20, 40 i 60 minut). Podczas dostrajania modeli neuronowych stosowano algorytm hybrydowy według schematu EA-LM.

Przyjęto następujące wartości parametrów funkcji celu (4.25) dla algorytmu globalnego: $R = 4$, $\lambda = 0.5, \omega_0 = 1.5$ oraz dla algorytmu lokalnego: $R = 2$, $\lambda = 0$.

Tab. 5.11: Wybór relewantnych wejść modelu neuronowego

Lp.	Metoda	Parametry	Zmienne niezależne									
			u_4	u_5	u_6	u_7	u_8	u_9	u_{10}	u_{11}	u_{12}	u_{13}
1	MWI	$v_{jk} = I_{jk},\ \nu_{ij} = r_{ij}$	0	1	0.9	0.4	0.4	0	0.1	0	0.4	0
2	MWI	$v_{jk} = nr_{jk},\ \nu_{ij} = r_{ij}$	0	1	0.6	0.6	0.7	0.3	0	0	1	0
3	MWI	$v_{jk} = nr_{jk},\ \nu_{ij} = I_{ij}$	0.4	1	1	1	1	1	0.4	0.5	1	0.2
4	MWI	$v_{jk} = I,\ \nu_{ij} = I_{ij}$	0.3	1	1	1	0.7	0.7	0.1	0.1	0.9	0
5	MWI	$v_{jk} = \rho_{jk},\ \nu_{ij} = r_{ij}$	0	0	0.3	0.2	1	1	0.2	0.8	0.4	0
6	MWI	$v_{jk} = \tau_{jk},\ \nu_{ij} = r_{ij}$	0	0	0.3	0.1	1	1	0.2	0.7	0.5	0
7	MWI	$v_{jk} = I_{jk},\ \nu_{ij} = \mathrm{det}_{ij}$	0	1	0.4	0.2	0.2	0.1	0.2	0.1	0	0
8	MWI	$v_{jk} = I_{jk},\ \nu_{ij} = \mathrm{ent}_{ij}$	0.1	0.9	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0
9	MWI	$v_{jk} = I_{jk},\ \nu_{ij} = \mathrm{lam}_{ij}$	0	1	0.5	0.2	0.2	0	0.2	0.2	0	0
10	Γ	$L_2,\ z = 5$	0.4	0.5	0.6	0.7	1	0.8	0.7	0.5	0.4	0.8
11	Γ	$L_1,\ z = 5$	0.6	0.6	0.7	0.7	0.6	0.6	0.8	0.4	0.7	0.6
12	Γ	$L_\infty,\ z = 5$	0.6	0.6	0.7	0.7	0.6	0.6	0.8	0.4	0.7	0.6

W pierwszym etapie treningu stosowano algorytm ewolucyjny (20 epok) z kodowaniem rzeczywisto-liczbowym chromosomów, z krzyżowaniem heurystycznym ($\lambda_h = 1.1$, $p_k = 0.8$), z mutacją równomierną z odwzorowaniem Hénona ($r_m = 0.1$), stałą liczebnością populacji równą 100 osobników (populacja bazowa uzyskiwana metodą INIT1), selekcją proporcjonalną oraz sukcesją elitarną ($\delta_s = 1$). W kolejnym etapie procesu dostrajania modeli stosowano algorytm Levenberga-Marquardta (10 iteracji) z numerycznie wyznaczanym jakobianem. Uzyskane wyniki modelowania zaprezentowano w Tab. 5.12.

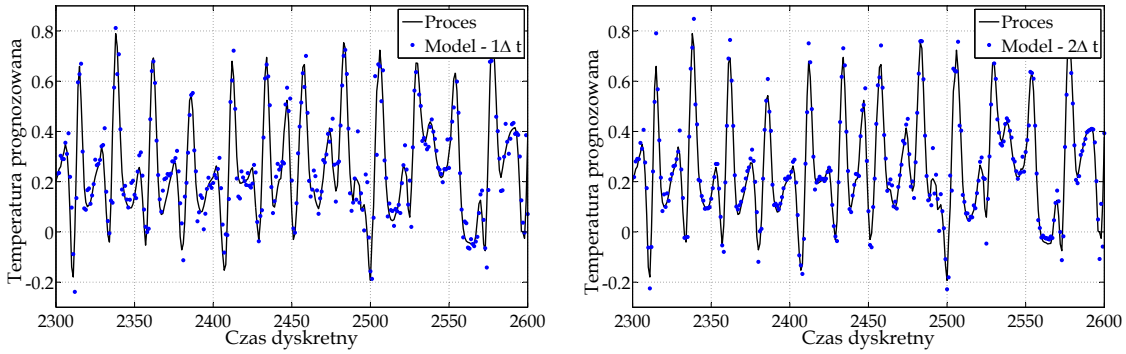
Tab. 5.12: Wyniki testowania modeli neuronowych dla różnych konfiguracji zmiennych niezależnych

Model predykcyjny	Struktura	p	Horyzont predykcji H=1Δt	H=2Δt	H=3Δt	
$y(k+H) = f(y, u_{1-3}, u_{5-9}, u_{14-18})$	$14 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	126	3.126 0.444	6.059 0.473	22.973 1.086	× •
$y(k+H) = f(y, u_{1-3}, u_{5-6}, u_{14-18})$	$11 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	111	3.085 0.429	10.944 0.558	9.493 0.605	× •
$y(k+H) = f(y, u_{1-3}, u_{5-8}, u_{12}, u_{14-18})$	$14 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	126	2.929 0.442	– –	25.094 0.844	× •
$y(k+H) = f(y, u_{1-3}, u_{5-9}, u_{12}, u_{14-18})$	$15 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	131	3.082 0.429	6.299 0.467	11.251 0.667	× •
$y(k+H) = f(y, u_{1-3}, u_{8-9}, u_{11}, u_{14-18})$	$13 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	116	3.054 0.471	6.209 0.470	9.304 0.611	× •
$y(k+H) = f(y, u_{1-3}, u_5, u_{14-18})$	$10 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	106	2.992 0.421	6.700 0.491	9.732 0.649	× •
$y(k+H) = f(y, u_{1-3}, u_{6-10}, u_{13-18})$	$15 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	131	2.824 0.429	7.875 0.567	24.287 0.841	× •
$y(k+H) = f(y, u_{1-3}, u_{4-10}, u_{12-18})$	$18 \to 4^{(2,2,2)}_{(0,2,0)} \to 1$	146	2.917 0.452	6.947 0.505	23.836 0.824	× •

× - błąd mMAPE, • - błąd U wyznaczany w odniesieniu do prognozy naiwnej

Największą selektywność istotnych zmiennych procesowych uzyskano dla metody wskaźników informacyjnych, dla której współczynnik określający zależność pomiędzy i-tą

i j-tą potencjalną zmienną wejściową modelu wyznaczano za pomocą miar bazujących na analizie diagramów rekurencyjnych. Podobne wyniki otrzymano, gdy jako miary zależności υ_{jk} i ν_{ij} stosowano odpowiednio informację wzajemną oraz współczynnik korelacji Pearsona. Najmniejszą selektywność uzyskiwano dla metody opartej na teście Γ. Na Rys. 5.12 przedstawiono dokładność uzyskiwanych prognoz dla modelu o najmniejszej liczbie wejść.



(a) (b)

Rys. 5.14: Prognozowane wartości temperatury sklepienia pieca pomiędzy elektrodami E1 i E2 uzyskane dla modelu neuronowego o najmniejszej liczbie wejść: (a) H $= 20$ [min.], (b) H $= 40$ [min.] ($\Delta t = 20$ [min.])

5.4. Odporna detekcja uszkodzeń wybranego elementu wykonawczego

Autor postanowił ponadto zweryfikować przydatność opracowanej metodyki do projektowania algorytmów odpornej detekcji uszkodzeń procesów rzeczywistych. Badania przeprowadzono dla wybranego elementu wykonawczego zainstalowanego w stacji wyparnej Cukrowni Lublin S.A. (Kościelny, 2001). Dane zgromadzone podczas monitorowania procesu zagęszczania soku buraczanego zostały udostępnione w ramach europejskiego projektu RTN-DAMADICS (ang. *Research Training Network - Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems*) (Bartyś, 2001; DAMADICS, 2002).

Ponadto opracowany został symulator testowy urządzenia wykonawczego (uwzględniający uszkodzenia), który umożliwia generowanie danych reprezentujących obiekt w różnych stanach (Bartyś *i in.*, 2006). Udostępnione dane historyczne i symulator obiektu były stosowane przez wielu autorów do relatywnej oceny algorytmów detekcji i lokalizacji uszkodzeń projektowanych projektowanych przez tych autorów (Calado *i in.*, 2006; Korbicz *i in.*, 2004; Korbicz, 2006; Korbicz i Kowal, 2007; Korbicz i Mrugalski, 2008; Kościelny *i in.*, 2006; Mrugalski *i in.*, 2008; Patan, 2008b; Patan *i in.*, 2008; Puig *i in.*, 2007; Witczak *i in.*, 2006).

Zgodnie z zaleceniami twórców projektu DAMADICS badania opisane w dalszej części tego podrozdziału prowadzono w dwóch etapach. Pierwszy etap polegał na budowie i ocenie systemu detekcji uszkodzeń, działającego na podstawie proponowanego modelu neuronowego wyłącznie na danych pozyskanych z symulatora numerycznego. W końcowym stadium weryfikacji pokazano przydatność metodyki na danych historycznych zgromadzonych podczas działania obiektu w warunkach rzeczywistych.

5.4.1. Obiekt diagnozowania

Proces technologiczny realizowany w stacji wyparnej cukrowni polega na zagęszczaniu soku buraczanego poprzez odparowanie z niego wody. Stacja ta składa się z siedmiu aparatów wyparnych, które połączone są szeregowo. Sok buraczany, przepływając przez kolejne aparaty, jest stopniowo zagęszczany. Czynnikiem grzewczym w aparatach wyparnych jest para świeża (z komory grzewczej) lub para powstała wtórnie z wody odparowanej z soku. W trakcie przepływu czynnika pomiędzy kolejnymi aparatami/sekcjami następuje stopniowy spadek jego temperatury i ciśnienia. W ostatnim aparacie utrzymywane jest podciśnienie oparów.

Przebieg procesu w stacji wyparnej kontrolowany jest przez wiele układów automatycznej regulacji, do których zaliczamy: układy regulacji temperatury soku, układy regulacji poziomu soku w aparatach wyparnych, układy regulacji przepływu soku pobieranego ze stacji wyparnej, układy regulacji ciśnienia i podciśnienia. Typowym urządzeniem wykonawczym pracującym w stacji wyparnej jest zespół składający się z zaworu regulacyjnego, siłownika pneumatycznego oraz ustawnika pozycyjnego.

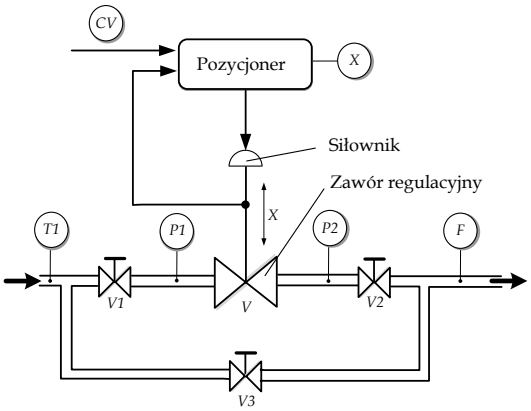
Na Rys. 5.15 pokazano fotografię tego typu urządzenia oraz schemat ideowy zasady jego działania w typowych warunkach eksploatacyjnych. Przyjęto następujące oznaczenia: V - zawór regulacyjny, $V1 - V3$ - zawory odcinające ręczne, CV - sygnał wartości zadanej [$\%$], X - sygnał wyjściowy przemieszczenia trzpienia siłownika [$\%$], $P1$, $P2$ - sygnały ciśnienia medium na wejściu i wyjściu zaworu [kPa], $T1$ - temperatura medium na wejściu zaworu [°C], F - sygnał strumienia przepływu [m^3/h]. Zakłada się, że do celów diagnostycznych dostępne pomiarowo są zmienne procesowe: X, F, CV, $P1$, $P2$ oraz $T1$.

Na podstawie analizy strukturalnej oraz dokumentacji dotyczącej obiektu (Kościelny, 2001; Bartyś *i in.*, 2006) możliwe było zdefiniowanie dwóch ogólnych zależności pomiędzy zmiennymi procesowymi:

- $F = g_F\,(X, P1, P2, T1)$,
- $X = g_X\,(CV, P1, P2, T1)$.

Wymienione zależności mogą służyć do projektowania systemów detekcji, lokalizacji i identyfikacji uszkodzeń rozpatrywanego zespołu wykonawczego. W ramach projektu DAMADICS przyjęto dwa podstawowe typy uszkodzeń, które mogą wpływać na pracę obiektu: uszkodzenia nagłe (ang. *abrupt*) o małym, średnim lub dużym rozmiarze oraz uszkodzenia wolno narastające (ang. *incipient*) o krótkim, średnim i długim horyzoncie

(a) Przykładowy obiekt diagnozowania



(b) Schemat blokowy obiektu

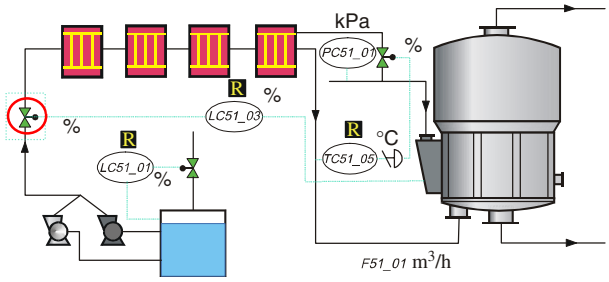
Rys. 5.15: Urządzenie wykonawcze składające się z zaworu regulacyjnego, siłownika pneumatycznego oraz ustawnika pozycyjnego (Bartyś *i in.*, 2006)

rozwoju. W Tab. 5.13 wyszczególniono 44 scenariusze z różnymi przypadkami uszkodzeń (uzasadnionych fizycznie). Dodatkowo przyjęto ograniczenie, że rozpatrywane są wyłącznie stany elementarne.

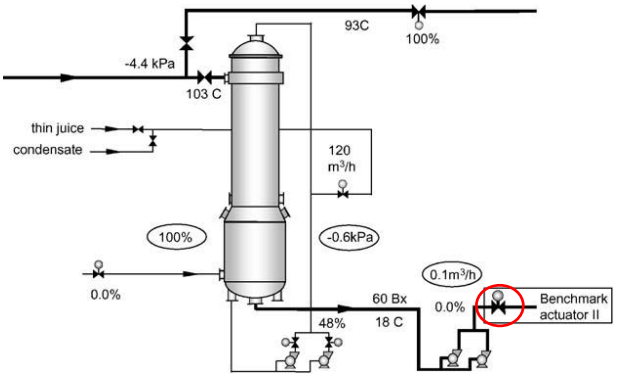
Tab. 5.13: Rozpatrywane uszkodzenia obiektu badań

Id.	**Opis uszkodzenia**	**Nagłe**			**Narastające**		
		Małe	Średnie	Duże	Krótko	Średnio	Długo
Uszkodzenia zaworu regulacyjnego							
f_1	Zablokowanie zaworu przez ciało obce	X	X	X	-	-	-
f_2	Zmiana charakterystyki przepływowej (sedymentacja)	-	-	X	-	-	X
f_3	Zużycie erozyjne zaworu	-	-	-	-	-	X
f_4	Wzrost siły tarcia w dławnicy lub w gnieździe	-	-	-	-	X	-
f_5	Przeciek zewnętrzny medium	-	-	-	-	-	X
f_6	Przeciek wewnętrzny medium	-	-	-	-	-	X
f_7	Zjawisko kawitacji	X	X	X	-	-	-
Uszkodzenia siłownika pneumatycznego							
f_8	Skrzywienie trzpienia	X	X	X	-	-	-
f_9	Nieszczelność komory lub przyłączy	-	-	-	-	-	X
f_{10}	Przebicie membrany siłownika	X	X	X	-	-	-
f_{11}	Uszkodzenie sprężyny siłownika	-	-	X	-	-	X
Uszkodzenia elementów pozycjonera							
f_{12}	Uszkodzenie przetwornika elektropneumatycznego	X	X	X	-	-	-
f_{13}	Uszkodzenie toru pomiarowego położenia	X	X	X	X	-	-
f_{14}	Uszkodzenie czujnika ciśnienia	X	X	X	-	-	-
f_{15}	Uszkodzenie mechaniczne toru sprzężenia zwrotnego	-	-	X	-	-	-
Uszkodzenia zewnętrzne							
f_{16}	Spadek ciśnienia zasilania pozycjonera	X	X	X	-	-	-
f_{17}	Zmiana ciśnienia przed/lub za zaworem	-	-	X	-	X	-
f_{18}	Otwarty lub nieszczelny tor obejścia	X	X	X	-	-	X
f_{19}	Uszkodzenie toru pomiarowego przepływu	X	X	X	-	-	-

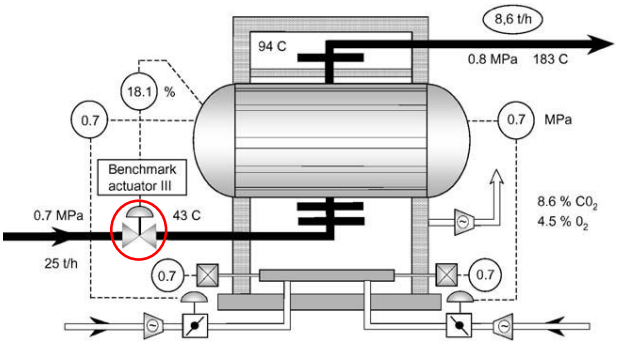
Dane rzeczywiste zgromadzone w stacji wyparnej dotyczą urządzeń wykonawczych zainstalowanych w różnych miejscach obiektu. Na Rys. 5.16 pokazano trzy punkty, w których prowadzono eksperymenty diagnostyczne w celu pozyskania danych. Eksperymenty te polegały na wprowadzaniu fikcyjnych uszkodzeń. Łącznie zgromadzono dane dotyczące pracy obiektu w okresie około jednego miesiąca (odstęp próbkowania $\Delta t = 1$ s), przy czym w warunkach rzeczywistych wprowadzano jedynie uszkodzenia $f_{16}, f_{17}, f_{18}, f_{19}$.



(a) Aparat wyparny starszego typu



(b) Aparat wyparny nowszego typu



(c) Komora grzewcza

Rys. 5.16: Rozmieszczenie rozpatrywanych urządzeń wykonawczych (Z1, Z2, Z3) w stacji wyparnej cukrowni (Bartyś *i in.*, 2006)

W praktyce nie jest możliwe uzyskanie danych reprezentujących wszystkie stany obiektu prezentowane w Tab. 5.13. Jest to powodem, dla którego autorzy projektu opracowali symulator urządzenia w środowisku MATLAB Simulink, który umożliwia generowanie adekwatnych sygnałów odpowiadających różnym stanom obiektu.

Szczegółowy opis procesu wraz z danymi pochodzącymi ze stacji wyparnej cukrowni oraz opracowany na ich podstawie symulator urządzenia wykonawczego dostępne są na stronie internetowej projektu (DAMADICS, 2002).

5.4.2. Weryfikacja - Etap I

Skuteczność zaproponowanej metodyki sprawdzono dla problemu testowego z wykorzystaniem danych uzyskanych za pomocą symulatora obiektu. Proponowaną metodę detekcji uszkodzeń opisano w rozdziale 4.8.

Pierwszym krokiem w konstruowaniu tak pomyślanego systemu detekcji uszkodzeń było wytrenowanie modeli neuronowych na danych, które zgromadzone zostały w warunkach pracy nominalnej obiektu. Biorąc pod uwagę definicje i opisy zawarte w dokumentacji udostępnionej w ramach problemu testowego oraz analizy prowadzonej przez autora z użyciem metody wskaźników informacyjnych, wyznaczono dwie relacje: $X = g_X\,(CV, P1, P2, T1)$, $F = g_F\,(X, P1, P2, T1)$. Zależności te identyfikowano, stosując dwuwarstwowe i trójwarstwowe struktury lokalnie rekurencyjne z tangensoidalnymi funkcjami wyjścia neuronów warstwy ukrytej w postaci (4.8, $\alpha_f = 0.5$) i liniowymi funkcjami wyjścia neuronów warstwy wyjściowej. Użyta postać funkcji tangensoidalnych umożliwia wystąpienia zjawiska chaosu w pojedynczych elementach przetwarzających. Do trenowania modeli neuronowych użyto 2000 przykładów trenujących. Na etapie testowania modeli zastosowano 900 przykładów, które były różne od użytych na etapie trenowania. Dane generowano zgodnie z zaleceniami autorów projektu (DAMADICS, 2002).

Proces strojenia każdego modelu neuronowego realizowano przez minimalizację funkcji celu (4.25) za pomocą schematu hybrydowego będącego połączeniem algorytmu ewolucyjnego i algorytmu LM przy założeniu $R = 4$, $\lambda = 0.5$ i $\omega_0 = 1.5$ dla uczenia wstępnego i $R = 2$, $\lambda = 0$ dla douczania. Przyjęto następujące wartości głównych cech algorytmu ewolucyjnego: wielkość populacji równa 50 osobników (kodowanie rzeczywisto-liczbowe), mutacja równomierna z odwzorowaniem Ikedy ($r_m = 0.1$), krzyżowanie heurystyczne ($p_k = 0.6$, $\lambda_h = 1.1$). Algorytm globalny uruchamiany był dla liczby epok równej 30. W kolejnym kroku wykonywano 15 iteracji algorytmu lokalnego w celu dostrojenia parametrów modeli. Biorąc pod uwagę wyniki zamieszczone w Tab. 5.14 i 5.15 oraz zaprezentowane na Rys. 5.17(a) i (b), postanowiono, że zależność g_X może być reprezentowana za pomocą modelu neuronowego nr 32, natomiast do odwzorowania zależności g_F może być użyty model nr 7.

Następnie struktury wybranych modeli poddano procesowi przycinania połączeń nieistotnych z zastosowaniem metody OBD. Procedura ta pozwoliła na redukcję 25% parametrów modelu nr 32 ($p = 30$) oraz redukcję około 17% parametrów modelu nr 7 ($p = 26$). Po etapie przycinania niezbędne było ponowne dotrenowanie modeli za pomocą algorytmu LM (10 iteracji) w celu uzyskania błędów uogólniania na poziomie wartości uzyskanych przed procesem przycinania.

Kolejnym krokiem podczas budowy systemu detekcji uszkodzeń było opracowanie

Tab. 5.14: Oceny błędów wybranych modeli neuronowych przemieszczenia trzpienia

$X = g_X\left(CV, P1, P2, T1\right)$					
Nr	**Oznaczenie**	**Struktura**	**p**	**mMAPE**	**HQC**
...	...	...	...	...	...
32	■	$4-3^{(2,2,0)}_{(0,2,0)}-1$	40	1.582	−5548
34	■	$4-3^{(2,2,1)}_{(0,2,0)}-1$	43	1.646	−5501
38	■	$4-4^{(0,1,0)}_{(0,2,0)}-1$	41	1.663	−5499
...	...	...	...	...	...
91	•	$4-3-2^{(0,2,2)}_{(0,2,0)}-1$	43	2.667	−5067
95	•	$4-3-2^{(1,1,1)}_{(0,2,0)}-1$	41	2.457	−5148
102	•	$4-3-2^{(1,2,2)}_{(0,1,0)}-1$	43	2.303	−5199
...	...	...	...	...	...

Tab. 5.15: Oceny błędów wybranych modeli neuronowych przepływu medium

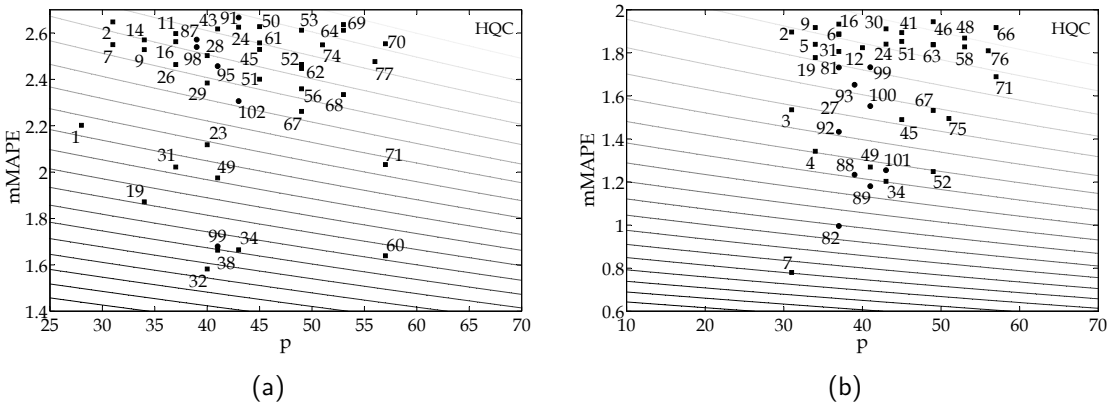
$F = g_F\left(X, P1, P2, T1\right)$					
Nr	**Oznaczenie**	**Struktura**	**p**	**mMAPE**	**HQC**
...	...	...	...	...	...
7	■	$4-3^{(0,2,0)}_{(0,1,0)}-1$	31	0.779	-6218
34	■	$4-3^{(2,2,1)}_{(0,2,0)}-1$	43	1.203	-5782
52	■	$4-4^{(1,1,1)}_{(0,2,0)}-1$	49	1.259	-5718
...	...	...	...	...	...
82	•	$4-3-2^{(0,1,1)}_{(0,1,0)}-1$	37	0.995	-5976
88	•	$4-3-2^{(0,2,1)}_{(0,1,0)}-1$	39	1.234	-5775
89	•	$4-3-2^{(0,2,1)}_{(0,2,0)}-1$	41	1.185	-5804
...	...	...	...	...	...

odpornego sposobu oceny residuów otrzymywanych za pomocą modeli neuronowych. Przyjęto ograniczenie, że na etapie projektowania systemu detekcji nie ma dostępu do danych reprezentujących stan obiektu z uszkodzeniami. Blok decyzyjny zrealizowano za pomocą zbioru reguł ostrych zapisanych w postaci kanonicznej:

$$\mathcal{R} = \left\{ \begin{array}{lllll} \textbf{if} & \mathrm{rr}^{(i)}\left(k\right) \geq \epsilon^{(i)}_{\mathrm{rr}} & \textbf{then} & \mathcal{F}^{(i)}_1 = 1 & \textbf{else} \quad \mathcal{F}^{(i)}_1 = 0; \\ \textbf{if} & \det{}^{(i)}\left(k\right) \geq \epsilon^{(i)}_{\det} & \textbf{then} & \mathcal{F}^{(i)}_2 = 1 & \textbf{else} \quad \mathcal{F}^{(i)}_2 = 0; \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \textbf{if} & \sum\limits_{j=1}^{5} \mathcal{F}^{(i)}_j = 5 & \textbf{then} & \mathcal{F}^{(i)} = 1 & \textbf{else} \quad \mathcal{F}^{(i)} = 0; \end{array} \right\}_{i=1}^{2}, \tag{5.3}$$

gdzie $k = \Delta k, \Delta k + 1, \ldots, N - \left(d^{(i)} - 1\right)\tau^{(i)}$, N - liczba próbek residuum, $\epsilon^{(i)}_{\mathrm{rr}}$, $\epsilon^{(i)}_{\det}$,

$\epsilon_{\text{ent}}^{(i)}$, ... reprezentują wartości graniczne dla pięciu miar RQA, które wyliczano dla każdego z residuów ($r^{(1)} = r_{g_X}$ i $r^{(2)} = r_{g_F}$). Miary te wyznaczano dla diagramów rekurencyjnych generowanych dla określonej wartości progu $\epsilon^{(i)}$, wybranej metryki oraz okna o wymiarze Δk, które przesuwano wzdłuż wykresu każdego z residuów. Występowanie uszkodzenia ($\mathcal{F}^{(i)} = 1$) w chwili k sygnalizowane było przez system detekcji, jeżeli wszystkie pięć flag $\mathcal{F}_j^{(i)}$ wyznaczanych dla i-tego residuum przyjmowało wartość logiczną 1.



(a) (b)

Rys. 5.17: Linie izokryterialne: Wyniki testowania modeli neuronowych przemieszczenia trzpienia siłownika (a) i przepływu strumienia medium (b)

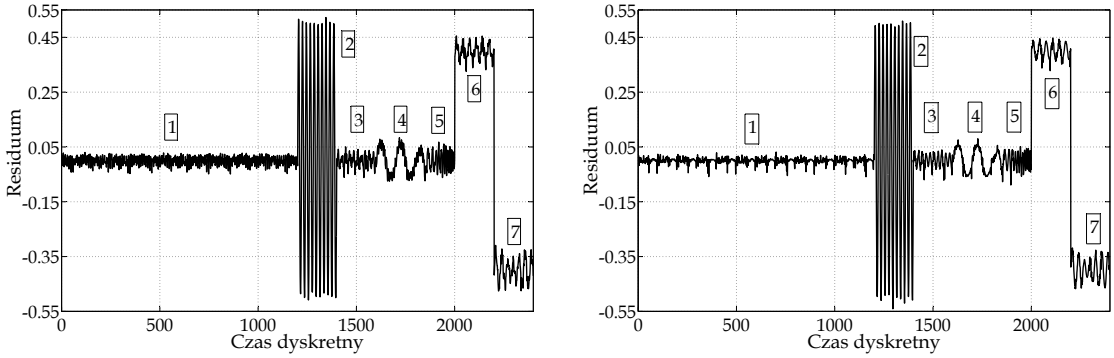
Głównym problemem w tym kroku było ustalenie wartości parametrów $d^{(i)}$, $\tau^{(i)}$ służących do rekonstrukcji obrazu atraktora i-tego residuum w przestrzeni fazowej oraz rodzaju metryki i wartości progu $\epsilon^{(i)}$ służących do uzyskania diagramu rekurencyjnego (dla i-tego residuum). Ponadto niezbędne było ustalenie wartości granicznych progów $\epsilon_{(\cdot)}^{(i)}$ dla miar RQA. W tym celu zdecydowano się na użycie algorytmu ewolucyjnego. W eksperymencie ewolucyjnym minimalizowano wartość następującej funkcji przystosowania:

$$\Phi^{(i)}\left(d^{(i)}, \tau^{(i)}, \boldsymbol{\epsilon}^{(i)}\right) = (1 - w_{td}) r_{fd}^{(i)} + w_{td}(1 - r_{td}^{(i)}), \tag{5.4}$$

gdzie $r_{fd}^{(i)}$, $r_{td}^{(i)}$ reprezentują wskaźniki fałszywych i prawdziwych alarmów (Bartyś *i in.*, 2006) uzyskiwanych dla odpowiednich residuów, za pomocą reguł bloku decyzyjnego, w_{td} określa wpływ poszczególnych wskaźników na wartość funkcji przystosowania.

Ze względu na założenie dotyczące braku przykładów reprezentujących pracę obiektu z uszkodzeniami, niezbędne było przygotowanie odpowiednich residuów (z fikcyjnie wprowadzanym uszkodzeniem), dla których można było wyznaczyć wartość funkcji przystosowania (patrz rozdział 4.8). Zaproponowano przebiegi residuów o postaciach pokazanych na Rys. 5.18(a) i (b). Fragmenty residuów oznaczone numerem 1 uzyskano dla nominalnej pracy obiektu (dla tych fragmentów wskaźnik $r_{fd}^{(i)}$ powinien przyjmować wartość 0). Fragmenty (2-5) stanowią sumę przebiegów residuów uzyskanych dla pracy nominalnej obiektu oraz wygenerowanych składowych harmonicznych/poliharmonicznych. Sygnały 6-7 złożone są ze składowej stałej, składowej harmonicznej oraz tak jak poprzednio z re-

siduum uzyskanego dla warunków nominalnych. Dla fragmentów residuów oznaczonych numerami 2-7, które reprezentują obiekt w stanie z fikcyjnym uszkodzeniem, wskaźnik $r_{td}^{(i)}$ powinien przyjmować wartość 1.



(a) Residuum $r^{(1)}$ (b) Residuum $r^{(2)}$

Rys. 5.18: Przebiegi residuów ze sztucznie wprowadzanymi uszkodzeniami

Przyjęto kodowanie rzeczywiste chromosomów, w których geny reprezentują wartości poszukiwanych parametrów. Algorytm ewolucyjny uruchamiany był dla obu residuów niezależnie. Główne parametry algorytmu były następujące: wielkość populacji równa 20 osobników, mutacja adaptacyjna ($r_m = 0.01$), krzyżowanie heurystyczne ($p_k = 0.8$, $\lambda_h = 1.1$), liczba epok równa 20. Rodzaj metryki i długość okna Δk dobrano tak, aby możliwe było osiągnięcie minimum funkcji przystosowania przy niewielkiej liczbie epok. Przyjęto równą wagę dla obu wskaźników ($w_{td} = 0.5$). W Tab. 5.16 zamieszczono wartości parametrów wyznaczone za pomocą algorytmu ewolucyjnego, których następnie użyto w bloku decyzyjnym.

Tab. 5.16: Parametry bloku decyzyjnego

$r^{(i)}$	Δk	Metryka	d	τ	$\epsilon^{(i)}$	$\epsilon_{\mathbf{rr}}^{(i)}$	$\epsilon_{\mathbf{det}}^{(i)}$	$\epsilon_{\mathbf{ent}}^{(i)}$	$\epsilon_{\mathbf{lam}}^{(i)}$	$\epsilon_{\mathbf{tt}}^{(i)}$
$r^{(1)}$	200	maksimum	[3.163]	[1.645]	0.681	0.920	0.192	0.442	0.417	0.477
$r^{(2)}$	200	maksimum	[2.147]	[1.845]	0.919	0.155	0.665	0.440	0.898	0.563

$[\cdot]$ - oznacza część całkowitą

W celu przetestowania opracowanego systemu detekcji uszkodzeń ponownie wygenerowano dane reprezentujące obiekt w stanach z uszkodzeniami. Przykłady testowe wygenerowano, używając symulatora obiektu i były one kompletnie różne od użytych na etapach trenowania modeli i optymalizacji części decyzyjnej systemu. Zastosowanie reguł bloku decyzyjnego do oceny residuów ($r^{(1)}$, $r^{(2)}$) umożliwiło detekcję uszkodzeń dla prawie wszystkich scenariuszy. Otrzymane wyniki zaprezentowano w Tab. 5.17. Przyjęto następującą notację:

- pozioma kreska wskazuje sytuacje, które nie były rozpatrywane,

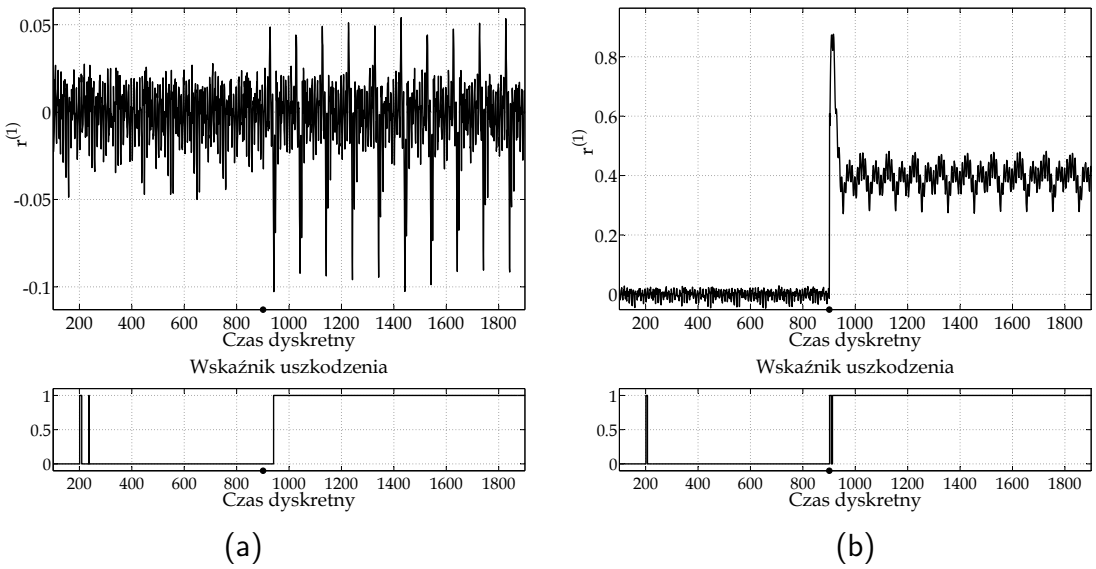
- znak $\times$ określa uszkodzenia niewykrywalne za pomocą wygenerowanych residuów,
- $r^{(1)}$ oznacza, że uszkodzenie może być wykryte z użyciem modelu g_X, $r^{(2)}$ oznacza, że uszkodzenie jest wykrywalne na podstawie modelu g_F,
- jeżeli uszkodzenie wykrywane jest z zastosowaniem obu modeli, to wskaźniki błędnych i prawidłowych decyzji dotyczą korzystniejszych wyników.

Tab. 5.17: Rezultaty detekcji uszkodzeń rozpatrywanego urządzenia wykonawczego

Id.	**Nagłe**						**Narastające**					
	Małe		Średnie		Duże		Krótki		Średni		Długi	
	r_{fd}	r_{td}	r_{fd}	r_{td}	r_{fd}	r_{td}	r_{fd}	r_{td}	r_{fd}	r_{td}	r_{fd}	r_{td}
f_1	0.01	0.96	0.00	0.99	0.00	1.00	–		–		–	
	$r^{(1)}$		$r^{(1)}$		$r^{(\underline{1},2)}$							
f_2	–		–		0.00	1.00	–		–		0.00	0.96
					$r^{(1,\underline{2})}$						$r^{(1,\underline{2})}$	
f_3	–		–		–		–		–		0.00	0.93
											$r^{(1,\underline{2})}$	
f_4	–		–		–		–		0.01	0.55	–	
									$r^{(1)}$			
f_5	–		–		–		–		–		0.00	0.60
											$r^{(1)}$	
f_6	–		–		–		–		–		0.00	0.96
											$r^{(1,\underline{2})}$	
f_7	0.01	0.99	0.01	0.99	0.01	0.99	–		–		–	
	$r^{(\underline{1},2)}$		$r^{(\underline{1},2)}$		$r^{(\underline{1},2)}$							
f_8	$\times$		$\times$		$\times$		–		–		–	
f_9	–		–		–		–		–		0.00	0.19
											$r^{(1)}$	
f_{10}	0.00	0.95	0.01	1.00	0.01	0.99	–		–		–	
	$r^{(1)}$		$r^{(1)}$		$r^{(1)}$							
f_{11}	–		–		0.00	0.99	–		–		0.00	0.80
					$r^{(1)}$						$r^{(1)}$	
f_{12}	$\times$		0.00	0.91	0.00	0.95	–		–		–	
			$r^{(\underline{1},2)}$		$r^{(\underline{1},2)}$							
f_{13}	0.00	1.00	0.00	1.00	0.00	1.00	0.01	0.96	–		–	
	$r^{(2)}$		$r^{(1,\underline{2})}$		$r^{(1,\underline{2})}$		$r^{(2)}$					
f_{14}	$\times$		$\times$		$\times$		–		–		–	
f_{15}	–		–		0.00	0.99	–		–		–	
					$r^{(\underline{1},2)}$							
f_{16}	$\times$		0.00	0.21	0.00	0.99	–		–		–	
			$r^{(1)}$		$r^{(1)}$							
f_{17}	–		–		0.00	0.99	–		0.00	0.97	–	
					$r^{(\underline{1},2)}$				$r^{(\underline{1},2)}$			
f_{18}	0.00	1.00	0.00	1.00	0.00	1.00	–		–		0.00	0.97
	$r^{(1,\underline{2})}$		$r^{(1,\underline{2})}$		$r^{(1,\underline{2})}$						$r^{(1,\underline{2})}$	
f_{19}	0.00	1.00	0.00	1.00	0.00	1.00	–		–		–	
	$r^{(2)}$		$r^{(2)}$		$r^{(2)}$							

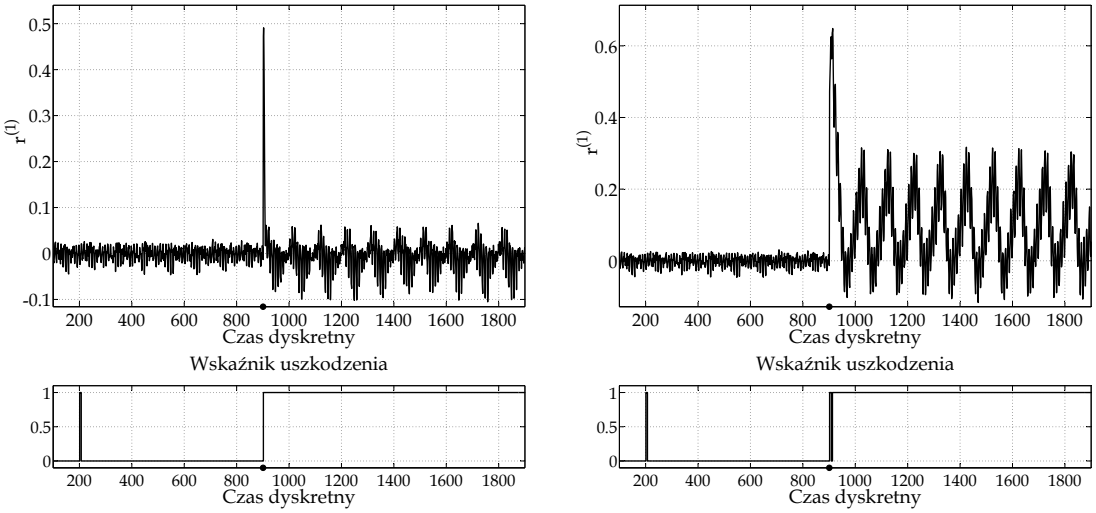
Jak można zauważyć, autor uzyskał satysfakcjonujące rezultaty detekcji obu typów uszkodzeń dla rozpatrywanego urządzenia wykonawczego. W dalszej części podrozdziału, na Rys. 5.19-5.24, pokazano residua oraz sygnały diagnostyczne uzyskane dla wybranych z Tab. 5.17 scenariuszy uszkodzeń nagłych i narastających. Moment

wystąpienia uszkodzenia oznaczony jest przez czarny punkt umieszczony na osi czasu.



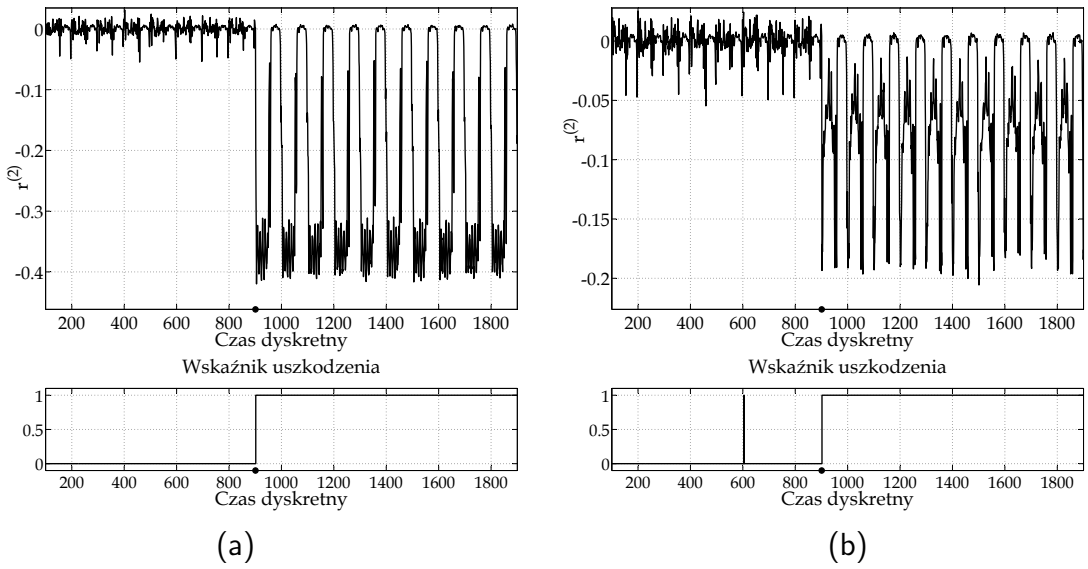
(a) (b)

Rys. 5.19: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residuum $r^{(1)}$ dla scenariuszy z uszkodzeniami nagłymi: (a) f_1(Małe) i (b) f_7(Średnie)

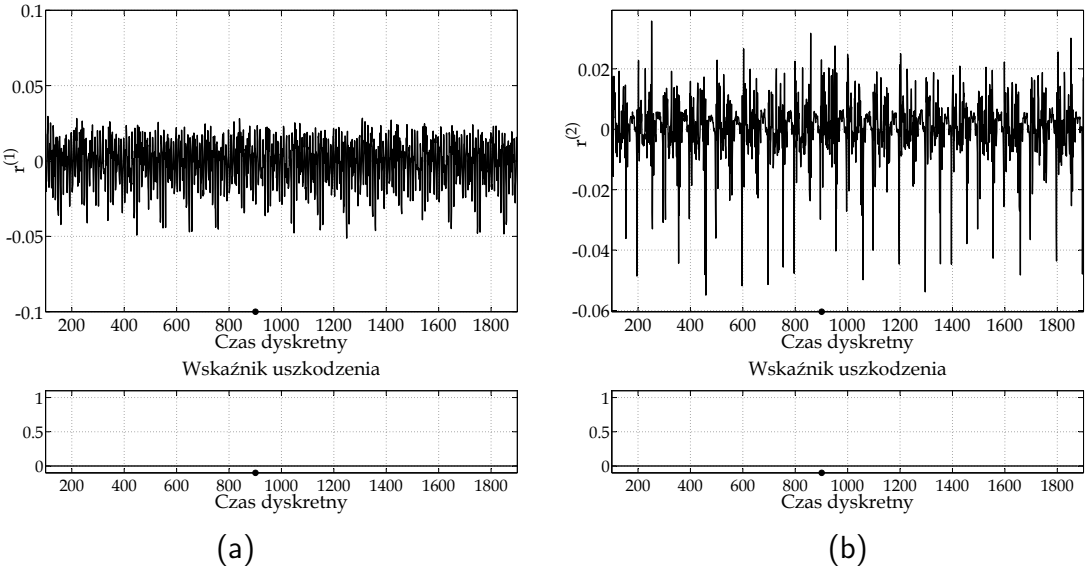


(a) (b)

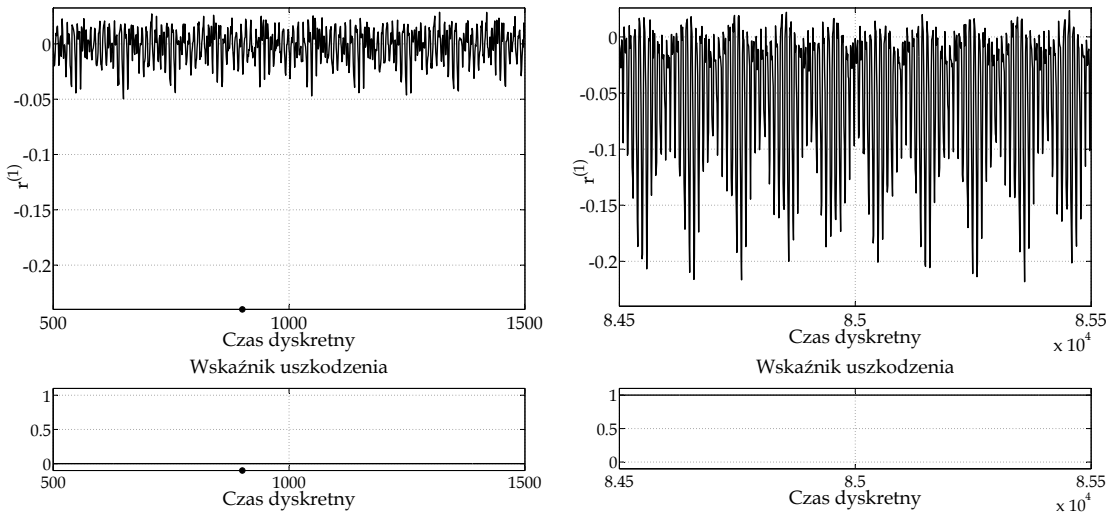
Rys. 5.20: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residuum $r^{(1)}$ dla scenariuszy z uszkodzeniami nagłymi: (a) f_{10}(Średnie) i (b) f_7(Duże)



Rys. 5.21: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residuum $r^{(2)}$ dla scenariuszy z uszkodzeniami nagłymi: (a) f_{18}(Małe) i (b) f_{19}(Małe)

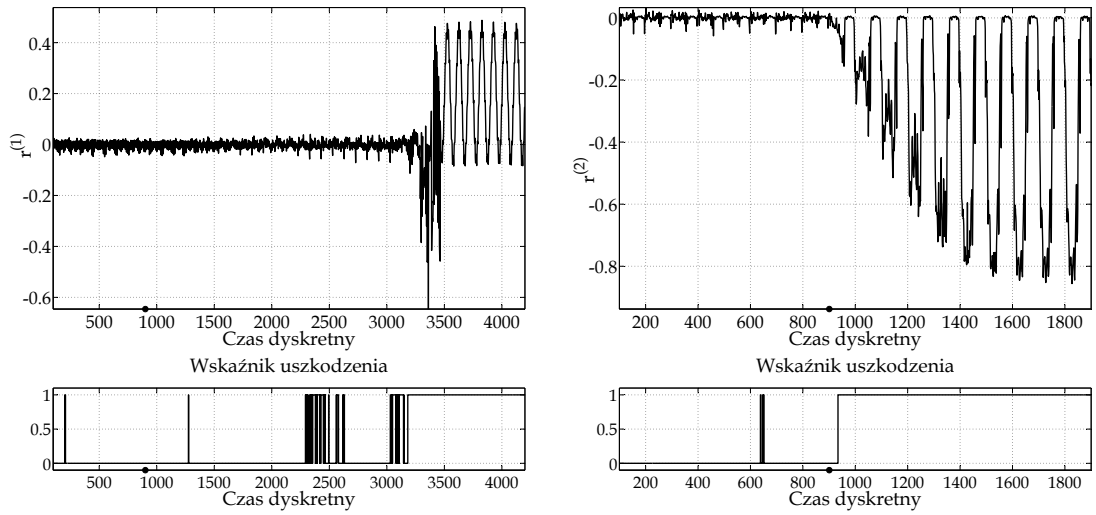


Rys. 5.22: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residuów $r^{(1)}$ i $r^{(2)}$ dla uszkodzenia f_8(Średnie)



(a) (b)

Rys. 5.23: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residuum $r^{(1)}$ dla scenariuszy z uszkodzeniem wolno narastającym f_{11}(Długi): (a) początkowa faza rozwoju, (b) końcowa faza rozwoju



(a) (b)

Rys. 5.24: Binarne sygnały diagnostyczne uzyskane w wyniku oceny residów $r^{(1)}$ i $r^{(2)}$ dla scenariuszy z uszkodzeniem średnio i szybko narastającym: (a) f_4(Średni) i (b) f_{13}(Krótki)

Niskie wartości wskaźników r_{fd} dla wszystkich scenariuszy (dla uszkodzeń niewykrywalnych również uzyskiwano wartości $r_{fd} \leq 0.01$) potwierdzają, że zbudowany system detekcji generuje bardzo mało fałszywych alarmów. W zdecydowanej większości przypadków otrzymano wartości $r_{td} \geq 0.95$.

Na podstawie wyników zamieszczonych w Tab. 5.17 oraz pokazanych na rysunkach widać, że nie było możliwe wykrycie następujących uszkodzeń: f_8 - skrzywienie trzpienia siłownika, f_{12}(Małe) - uszkodzenie przetwornika elektropneumatycznego, f_{14} - uszkodzenie czujnika ciśnienia oraz f_{16}(Małe) - spadek ciśnienia zasilania pozycjonera. Przykład uszkodzenia f_8(Średnie) niewykrytego przy zastosowaniu zaproponowanej metody (dla obu residuów) pokazano na Rys. 5.22. Ciekawe przykłady wykrywania uszkodzeń średnio i wolno narastających przedstawiono na Rys. 5.23 i 5.24. Dla tych scenariuszy można łatwo zauważyć, że zmiany sygnału residuum w początkowej fazie rozwoju uszkodzenia są bardzo małe (Rys. 5.24a) lub niezauważalne (rys. 5.23a). W tego typu sytuacjach nie zawsze możliwa jest więc szybka detekcja uszkodzenia. Z drugiej strony, biorąc pod uwagę wyniki z Tab. 5.17, możliwe było szybkie wykrycie uszkodzeń tego typu, np. w przypadku scenariuszy z uszkodzeniami: f_2(Długi) - zmiana charakterystyki przepływowej w wyniku sedymentacji, f_3(Długi) - zużycie erozyjne zaworu, f_6(Długi) - przeciek wewnętrzny medium, f_{17}(Średni) - zmiana ciśnienia przed/lub za zaworem oraz f_{18}(Długi) - otwarty lub nieszczelny tor obejścia.

Brak możliwości wykrycia uszkodzenia oraz bardzo niskie wartości wskaźników prawidłowych alarmów otrzymano dla scenariuszy z niektórymi uszkodzeniami, np. f_9(Długi) i f_{16}(Średnie). Jest to spowodowane następującymi faktami:

- zmiany wartości sygnału wyjściowego są poniżej poziomu szumów,
- urządzenie działa w pętli sprzężenia zwrotnego, co zapewnia pasywną adaptację układu sterowania do występującego uszkodzenia (w niewielkich zakresach),
- w tym teście rozpatrywany obiekt badań działa cyklicznie (zadana wartość przemieszczenia jest funkcją okresową), co prowadzi do sytuacji, że dla pewnych zakresów działania urządzenia uszkodzenia uwidaczniają się, a dla innych są ukryte.

Ostatni punkt można wyjaśnić na przykładzie prostego siłownika cylindrycznego złożonego z cylindra i tłoczyska. Uszkodzenie końcowego fragmentu cylindra będzie objawiało się, jeżeli zakres ruchu tłoczyska będzie obejmował uszkodzony fragment cylindra. Jeżeli urządzenie będzie pracować w zakresie, w którym uszkodzenie nie występuje, to będzie ono niewykrywalne. Dlatego też należy bardzo ostrożnie wyciągać wnioski co do sprawności zaprojektowanego systemu detekcji wyłącznie na podstawie analizy wartości wskaźnika poprawnie wskazanego uszkodzenia r_{td}.

5.4.3. Weryfikacja - Etap II

Ostatnim etapem badań była weryfikacja metodyki na danych zgromadzonych podczas pracy wcześniej opisanego typu urządzenia wykonawczego w warunkach rzeczywistych.

Dane użyte do identyfikacji i testowania modeli pochodziły z kilku wybranych dni i dotyczyły trzech urządzeń (Tab. 5.18). Szczegółowy opis danych rzeczywistych zawarty jest na stronie projektu (DAMADICS, 2002; Bartyś i Syfert, 2002).

Pozyskane dane wstępnie przetworzono poprzez filtrację dolnoprzepustową (z zastosowaniem dolnopasmowego filtru o skończonej odpowiedzi impulsowej $30.$ rzędu) oraz decymację ze współczynnikiem decymacji $M_d = 30$. Następnie uzyskane sygnały poddano procedurze normalizacji do przedziału $[-1, 1]$. W wyniku tych działań otrzymano zbiory danych trenujących ($\mathcal{L}_T$ - dla pracy obiektu w warunkach nominalnych) oraz testowych ($\mathcal{L}_G$ - dla pracy obiektu w warunkach nominalnych oraz $\mathcal{L}_F$ - dla pracy obiektu z występującym uszkodzeniem).

Tab. 5.18: Dane trenujące i testowe

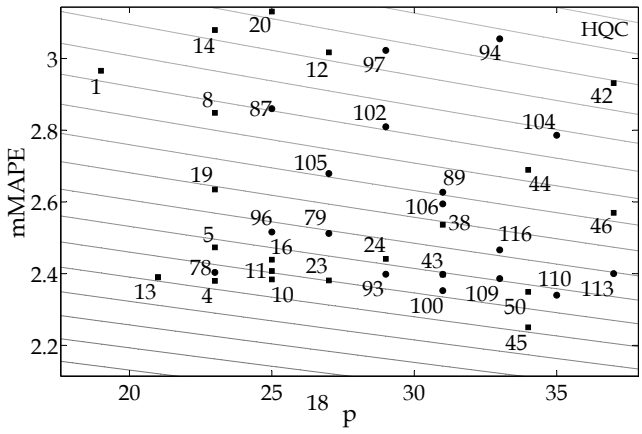
Urządzenie wykonawcze	F	Podzbiory danych $\mathcal{L}_T$	$\mathcal{L}_G$	$\mathcal{L}_F$	Czas występowania uszkodzenia
Z1	f_{18}	29.10.01 $1-2880$	30.10.01 $1-1000$	30.10.01 $1500-2500$	$1963-1996$
Z1	f_{18}	16.11.01 $1-2880$	17.11.01 $1-1000$	17.11.01 $1500-2500$	$1820-1823$
Z1	f_{16}	16.11.01 $1-2880$	17.11.01 $1-1000$	17.11.01 $1500-2500$	$1889-1892$
Z2	f_{17}	19.11.01 $1-2880$	20.11.01 $1-1000$	20.11.01 $1300-2500$	$1418-2500$
Z3	f_{19}	16.11.01 $1-2880$	17.11.01 $1-1000$	20.11.01 $1500-2500$	$1916-1958$

W dalszej części opisany zostanie kompletny przykład budowy modelu neuronowego z zastosowaniem opracowanej metodyki wyłącznie dla pierwszego przypadku z Tab. 5.18. Pozostałe modele tworzono w ten sam sposób.

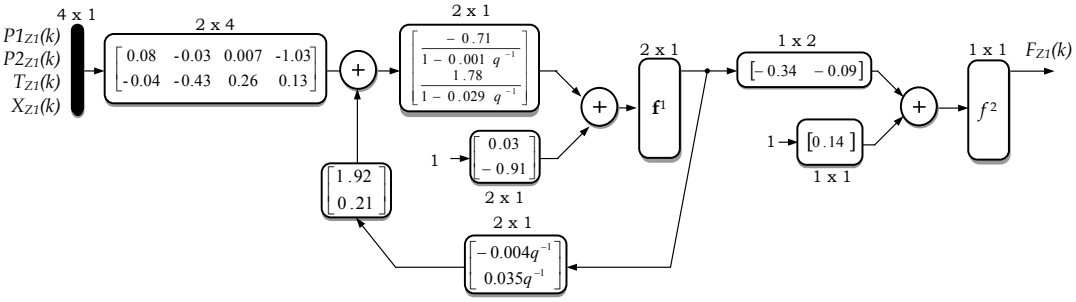
Ze względu na małą liczbę zmiennych procesowych, procedurę wyboru wejść relewantnych prowadzono wyłącznie w celu weryfikacji zgromadzonej wiedzy ekspertów udostępnionej w ramach opisu problemu testowego. Uzyskano w ten sposób ogólną zależność o postaci $F_{Z1} = g_1\left(P1_{Z1}, P2_{Z1}, T_{Z1}, X_{Z1}\right)$, którą identyfikowano za pomocą dwuwarstwowych i trójwarstwowych struktur lokalnie rekurencyjnych z tangensoidalnymi funkcjami wyjścia neuronów warstwy ukrytej w postaci (4.8, $\alpha_f = 0.5$) i liniowymi funkcjami wyjścia neuronów warstwy wyjściowej. Użyta postać funkcji tangensoidalnych neuronów warstw ukrytych zapewnia możliwość wystąpienia zjawiska chaosu w pojedynczych elementach przetwarzających.

Do strojenia modeli zastosowano algorytm hybrydowy według schematu EA-LM, przy czym przyjęto następujące wartości parametrów funkcji celu (4.25) dla algorytmu globalnego: $R = 2$, $\lambda = 0.5, \omega_0 = 1.5$ oraz dla algorytmu lokalnego: $R = 2$, $\lambda = 0$. W pierwszym etapie treningu stosowano algorytm ewolucyjny z maksymalną liczbą epok równą 30, z krzyżowaniem heurystycznym ($\lambda_h = 1.1$, $p_k = 0.6$), z mutacją równomierną z odwzorowaniem Hénona ($r_m = 0.1$), stałą liczebnością populacji równą 50 osobników (populacja bazowa uzyskiwana była metodą INIT0), z selekcją metodą ruletki oraz sukcesją elitarną ($\delta_s = 1$). Proces dostrajania modeli realizowano, stosując algorytm LM (15 iteracji) z numerycznie wyznaczanym jakobianem.

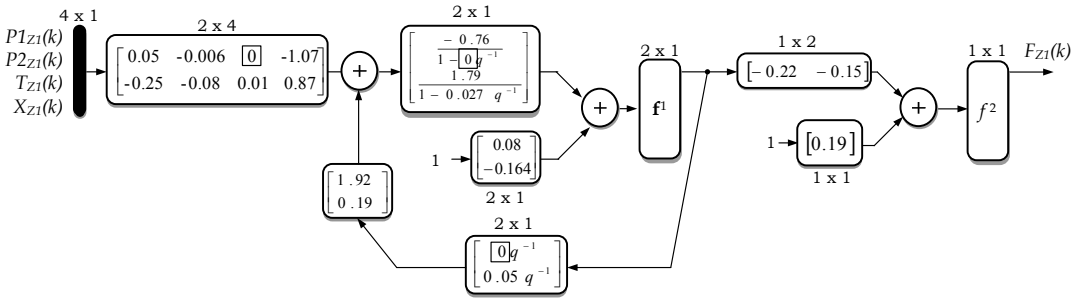
Wyniki testowania modeli zamieszczono na Rys. 5.25. Stosując zasadę oszczędności spośród dwóch modeli znajdujących się w pobliżu izolinii, dla której wartość kryterium HQC wynosi około -4500, wybrano model o numerze 13, którego strukturę pokazano na Rys. 5.26 (a).



Rys. 5.25: Izolinie kryterialne: wyniki testowania uzyskane dla poszczególnych modeli neuronowych dla kryterium Hannana-Quinna (identyfikacja modelu g_1)



(a) Przed przycinaniem



(b) Po przycięciu

Rys. 5.26: Model neuronowy nr 13 zależności $F_{Z1} = g_1\left(P1_{Z1}, P2_{Z1}, T_{Z1}, X_{Z1}\right)$ do detekcji uszkodzenia f_{18} urządzenia wykonawczego Z1

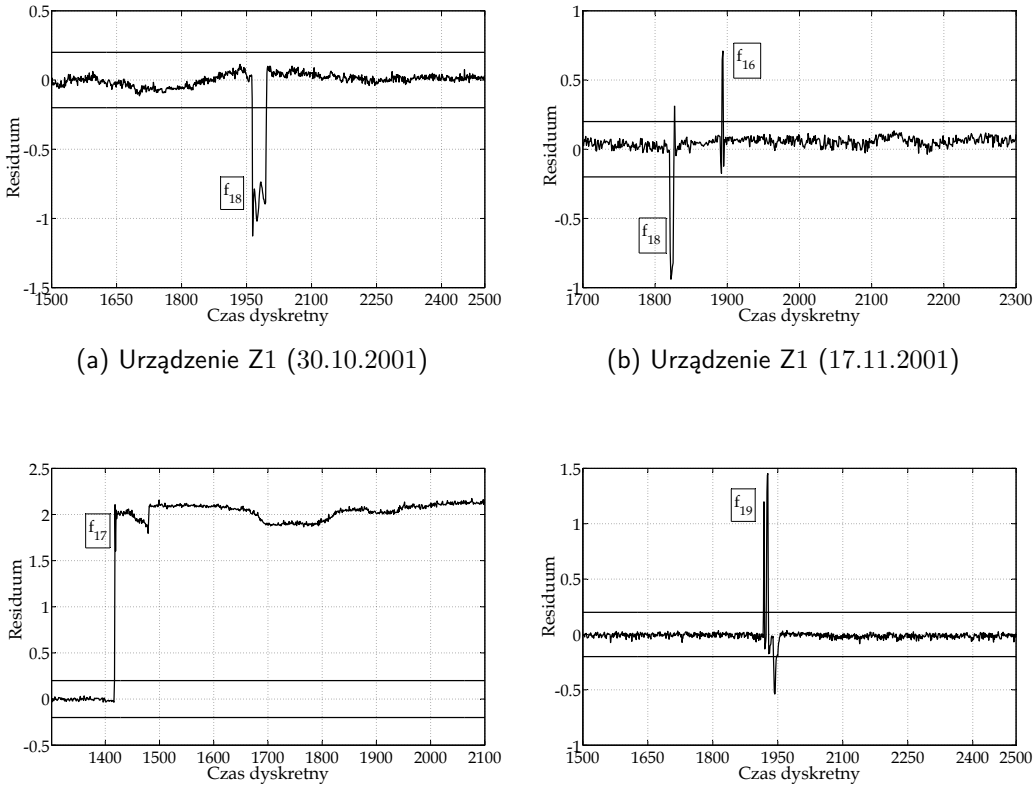
Kolejnym etapem budowy modelu była optymalizacja poszczególnych połączeń jego struktury z zastosowaniem metody OBD. Po usunięciu (wyzerowaniu) parametrów nieistotnych przeprowadzono proces douczania modelu wyłącznie z użyciem algorytmu LM (10 iteracji). W wyniku zastosowania tej procedury otrzymano model o ostatecznej

strukturze pokazanej na Rys. 5.26 (b), dla którego otrzymano wartość błędu testowania mMAPE na poziomie 2.7%.

Tak mała liczba parametrów nieistotnych potwierdza, że możliwe jest znalezienie struktury modelu reprezentatywnej dla danego problemu wyłącznie na podstawie analizy wykresów izolini kryterialnych. Ponadto można zauważyć, że metoda OBD umożliwiła wskazanie parametrów o najmniejszych wartościach.

Tab. 5.19: Wskaźniki jakości otrzymane dla poszczególnych modeli neuronowych

Urządzenie	F	Dane testowe	Model	Struktura	mMAPE
Z1	f_{18}	30.10.2001 $1-1000$	$F_{Z1} = g_1\left(P1_{Z1}, P2_{Z1}, T_{Z1}, X_{Z1}\right)$	$4 \to 2^{(1,1,0)}_{(0,1,0)} \to 1$	2.674
Z1	f_{18} f_{16}	17.11.2001 $1-1000$	$F_{Z1} = g_2\left(P1_{Z1}, P2_{Z1}, T_{Z1}, X_{Z1}\right)$	$4 \to 3^{(1,2,2)}_{(0,2,0)} \to 1$	3.216
Z2	f_{17}	20.11.2001 $1-1000$	$X_{Z2} = g_3\left(P1_{Z2}, P2_{Z2}, T_{Z2}, CV_{Z2}\right)$	$4 \to 3^{(0,2,1)}_{(0,1,0)} \to 1$	1.199
Z3	f_{19}	17.11.2001 $1-1000$	$X_{Z3} = g_4\left(P1_{Z3}, P2_{Z3}, T_{Z3}, CV_{Z3}\right)$	$4 \to 3^{(0,1,0)}_{(0,1,0)} \to 1$	4.890



(a) Urządzenie Z1 (30.10.2001)

(b) Urządzenie Z1 (17.11.2001)

(c) Urządzenie Z2 (20.11.2001)

(d) Urządzenie Z3 (17.11.2001)

Rys. 5.27: Residua uzyskane dla poszczególnych modeli neuronowych: (a) - model g_1, (b) - model g_2, (c) - model g_3, (d) - model g_4

Zbiorcze wyniki modelowania zamieszczono w Tab. 5.19. Model przepływu g_1 posłużył do detekcji uszkodzenia f_{18} elementu Z1 w dniu $30.10.2001$. Model g_2 zastosowano do wykrywania uszkodzenia f_{18} i f_{16} tego samego urządzenia w dniu $17.11.2001$. Z ko-

lei, residuum dla uszkodzenia f_{17} urządzenia Z2 wygenerowano z zastosowaniem modelu przemieszczenia trzpienia siłownika g_3, natomiast detekcję uszkodzenia f_{19} elementu Z3 uzyskano, stosując model g_4.

Na Rys.5.27 (a), (d), (c) i (d) pokazano przebiegi residuów uzyskane dla opracowanych modeli. Przyjęto arbitralnie progi detekcyjne na poziomie ± 0.2. Jak można zauważyć, w rozpatrywanym przykładzie, pomimo przyjęcia stałych progów detekcyjnych, możliwe było generowanie residuów, dla których nie uzyskiwano fałszywych alarmów.

Jak widać, wartości residuów w czasie występowania uszkodzeń f_{16} - spadek ciśnienia zasilania pozycjonera, f_{17} - zmiana ciśnienia przed lub za zaworem, f_{18} - otwarty lub nieszczelny tor obejścia oraz f_{19} - uszkodzenie toru pomiarowego przepływu, znacząco różnią się od zera, co potwierdza przydatność praktyczną opracowanych struktur neuronowych do detekcji uszkodzeń odpornej na szumy pomiarowe, zakłócenia niemierzalne oraz na błędy strukturalne.

5.5. Podsumowanie badań weryfikacyjnych

W niniejszym rozdziale zaprezentowano efektywność metodyki modelowania neuronowego z uwzględnieniem elementów teorii chaosu deterministycznego w diagnostyce procesów. Pierwszą część rozdziału stanowią badania wstępne, których celem było wykazanie przydatności opracowanej metodyki do modelowania procesów o różnym charakterze. Przeprowadzone w tym zakresie badania wykazały, że stosowanie zaproponowanego podejścia prowadzi do tworzenia modeli procesów o dużej dokładności. Uzyskiwane podczas eksperymentów identyfikacyjnych wartości błędów modelowania sugerują, że modele oparte na strukturze lokalnie rekurencyjnej opracowanej przez autora mogą znaleźć zastosowanie nie tylko w systemach diagnostycznych, ale również w systemach sterowania.

Druga część rozdziału zawiera wyniki badań aplikacyjnych, które wykazują przydatność zaproponowanego podejścia w zastosowaniach praktycznych. Badania aplikacyjne przeprowadzono na przykładzie diagnostyki uszkodzeń rzeczywistego obiektu przemysłowego. W tym celu wykorzystano zadanie testowe zdefiniowane w ramach projektu DAMADICS. Badania docelowe przeprowadzone na danych rzeczywistych oraz danych generowanych z użyciem symulatora obiektu wykazały, że zastosowanie opracowanej metodyki modelowania neuronowego, której metody i środki wykorzystują wybrane elementy teorii chaosu deterministycznego, umożliwia budowę odpornych systemów detekcji uszkodzeń.

Rozdział 6

Podsumowanie i wnioski

Przedstawiona praca jest wynikiem badań autora związanych z opracowaniem i weryfikacją metodyki modelowania neuronowego w diagnostyce procesów z uwzględnieniem wybranych elementów teorii chaosu deterministycznego. Przeprowadzone liczne badania pozwoliły na opracowanie uogólnionego modelu neuronu dynamicznego oraz struktury sieci neuropodobnej złożonej z tego typu jednostek, która umożliwia modelowanie szerokiej gamy procesów dynamicznych. Dla tak zdefiniowanej sieci lokalnie rekurencyjnej opracowano wiele schematów hybrydowego uczenia. W schematach tych zintegrowano znane metody optymalizacji globalnej i lokalnej funkcji wielu zmiennych. Metody te dodatkowo wzbogacono o nowe cechy charakterystyczne dla teorii chaosu deterministycznego. W celu weryfikacji i wykazania przydatności praktycznej opracowanej metodyki w diagnostyce procesów autor przeprowadził liczne testy dla danych symulacyjnych i pochodzących z obiektów rzeczywistych.

Efektem końcowym przeprowadzonych badań jest zapis formalny metodyki modelowania neuronowego, której zastosowanie podczas tworzenia systemu detekcji uszkodzeń wybranego obiektu rozważań może przyczynić się do poprawy sprawności tego systemu. Metodyka zawiera elementy teorii chaosu deterministycznego, które w głównej mierze stosowane są podczas budowy modelu neuronowego procesu oraz podczas budowy systemu diagnostycznego.

Podstawowymi elementami wchodzącymi w skład opisu formalnego metodyki są:

- uogólniony model neuronu dynamicznego,
- struktura sieci lokalnie rekurencyjnej globalnie jednokierunkowej,
- wiele metod dostrajania parametrów swobodnych opracowanej sieci neuronowej,
- wiele metod doboru struktury opracowanej sieci neuronowej,
- metoda badania stabilności modelu neuronowego,
- metoda odpornej detekcji uszkodzeń z zastosowaniem modelu neuronowego procesu oraz analizy diagramów rekurencyjnych.

Opis ten jest podstawą opracowanego przez autora oprogramowania w formie nowego pakietu dla systemu obliczeń inżynierskich MATLAB. Oprogramowanie to umożliwia tworzenie modeli neuronowych procesów dla potrzeb ich diagnostyki.

6.1. Wnioski ogólne

Wstępne badania weryfikacyjne potwierdziły, że zaproponowany model neuronu dynamicznego oraz struktura sieci neuropodobnej złożona z tego typu jednostek umożliwią modelowanie szerokiej gamy procesów dynamicznych. Wyniki, jakie otrzymano podczas badań, potwierdzają, że zaproponowana przez autora metodyka modelowania umożliwia skuteczne rozwiązanie wielu typowych problemów związanych z zastosowaniem obliczeń neuronowych w zadaniach identyfikacji systemów oraz modelowaniem procesów.

Badania zasadnicze przeprowadzone dla danych dotyczących wybranego elementu wykonawczego pracującego w warunkach rzeczywistych potwierdziły, że zastosowanie opracowanej metodyki podczas tworzenia systemów detekcji uszkodzeń może prowadzić do zwiększenia ich sprawności. Odporność detekcji uszkodzeń na zakłócenia, szumy pomiarowe oraz błędy modelowania uzyskano w wyniku dwóch działań: a) wprowadzenia nowej jednostki przetwarzającej, za pomocą której można tworzyć modele neuronowe dla szerokiej klasy procesów; b) opracowania metody oceny residuów z zastosowaniem analizy ilościowej diagramów rekurencyjnych (wyznaczanych dla tych residuów).

6.2. Wnioski szczegółowe

Biorąc pod uwagę wyniki uzyskane podczas badań wstępnych i zasadniczych, sformułowano następujące wnioski szczegółowe:

1. Wprowadzenie do prostego modelu neuronu systemów dynamicznych typu ARMAX w blokach aktywacyjnym i wyjściowego sprzężenia zwrotnego umożliwiło uzyskanie jednostki przetwarzającej, którą można zastosować podczas modelowania zjawisk zarówno o charakterze zdeterminowanym jak i losowym.
2. Analiza największego wykładnika Lapunowa wykazała złożony charakter zachowania pojedynczego neuronu uogólnionego o założonej strukturze.
3. Zaproponowany model neuronu umożliwia budowę modeli neuronowych dla szerokiej klasy układów dynamicznych, co przyczynia się do częściowego ograniczenia błędów strukturalnych.
4. Uczenie hybrydowe umożliwia częściowe ograniczenie błędów parametrycznych.
5. Zastosowanie schematu uczenia będącego połączeniem algorytmu ewolucyjnego i algorytmu Levenberga-Marquardta prowadzi do uzyskania najlepszych wyników dla sieci lokalnie rekurencyjnych o małej i średniej liczbie parametrów swobodnych.
6. Proste schematy bazujące na metodach przeszukiwania ślepego i estymacji kierunku gradientu mogą być stosowane w zadaniach modelowania procesów wymagających struktur o niewielkiej złożoności.
7. Zastosowanie mutacji chaotycznej umożliwia znalezienie rozwiązania początkowego, które zwiększa prawdopodobieństwo osiągnięcia rozwiązania bliskiego optymalnemu.

8. W przypadku procesów o wartościach sygnałów trudno przewidywalnych do selekcji wejść modelu neuronowego możliwe jest zastosowanie metody wskaźników pojemności informacyjnej rozwiniętej o miary zależności pomiędzy badanymi zmiennymi obliczane na podstawie analizy diagramów rekurencyjnych.
9. Możliwa jest znacząca redukcja wejść modelu neuronowego procesu z zastosowaniem rozwiniętej metody wskaźników pojemności informacyjnej.
10. Struktury lokalnie rekurencyjne zaproponowane przez autora umożliwiają osiągnięcie niższych błędów uogólnienia, niż w przypadku struktur wielokontekstowych sieci Jordana i Elmana oraz sieci globalnie rekurencyjnych typu NNARX.
11. Zastosowanie analizy ilościowej diagramów rekurencyjnych do oceny residuów pozwala na odporną detekcję uszkodzeń.

6.3. Kierunki dalszych badań

Podjęta przez autora próba opracowania metodyki modelowania neuronowego w diagnostyce procesów z uwzględnieniem elementów teorii chaosu pozwoliła zidentyfikować wiele ciekawych obszarów, w których celowe jest przeprowadzenie dodatkowych badań. Najważniejszymi zagadnieniami, które zdaniem autora należy rozważyć, są:

1. Sprawdzenie przydatności opracowanej metodyki w zadaniach lokalizacji i identyfikacji uszkodzeń (ze szczególnym naciskiem na zastosowanie elementów teorii chaosu deterministycznego).
2. Wyznaczenie analityczne gradientu oraz jakobianu funkcji błędu.
3. Opracowanie metody stabilizacji sieci z zastosowaniem bezpośredniej metody Lapunowa.
4. Opracowanie metod uczenia bazującego na metodach optymalizacji globalnej i lokalnej z ograniczeniami (wynikającymi z potrzeby stabilizacji modeli neuronowych).
5. Przeprowadzenie badań stabilności procesu uczenia z zastosowaniem drugiej metody Lapunowa.
6. Opracowanie metod automatycznego doboru struktury modelu neuronowego podczas strojenia jego parametrów swobodnych.
7. Przeprowadzenie wielu badań w zakresie rozbudowy jednostki przetwarzającej dotyczących np. zastosowania niestandardowych funkcji wyjściowych tj. funkcje specjalne (np. wielomiany Legendre'a), zastosowania wag będących funkcjami elementarnymi.
8. Opracowanie metody umożliwiającej interpretację modelu neuronowego procesu (np. poprzez transformację modelu neuronowego do postaci automatu skończonego).

Dodatek A

Globalne i lokalne algorytmy uczenia

A.1. Algorytm ewolucyjny

Optymalizację parametrów swobodnych sieci lokalnie rekurencyjnej realizowano według podstawowego schematu algorytmu ewolucyjnego zaproponowanego w (The MathWorks, 2007). Aby zastosować algorytm ewolucyjny w procesie uczenia sieci neuronowej niezbędne było określenie: sposobu reprezentacji przetwarzanej populacji, sposobu selekcji i sukcesji rozwiązań rokujących, sposobu prowadzenia operacji reprodukcji oraz warunku zatrzymania.

A.1.1. Populacja oraz kodowanie chromosomów

Przyjęto, że populacja składa się z L_p osobników z kodowaniem rzeczywisto-liczbowym chromosomów, w których geny przyjmują wartości odpowiednich parametrów sieci lokalnie rekurencyjnej (wagi połączeń synaptycznych, parametry funkcji wyjścia, parametry systemów dynamicznych kolejnych warstw sieci). Długość chromosomu zależy od złożoności struktury sieci i równa jest wymiarowi wektora $\boldsymbol{\omega}$:

$$\mathbf{chr} = \boldsymbol{\omega}^T = [\omega_1 \ \omega_2 \ \ldots \ \omega_j \ \ldots \ \omega_p] \,. \tag{A.1}$$

Populację początkową generowano z zastosowaniem wybranej metody INIT omówionej w rozdziale B.5. Przyjęty sposób kodowania osobników pozwala na bardziej zróżnicowane zastosowanie metod selekcji i sukcesji oraz operatorów mutacji i krzyżowania niż w przypadku prostego algorytmu genetycznego.

A.1.2. Funkcja przystosowania

Przystosowanie poszczególnych osobników w populacji określane jest za pomocą wartości funkcji celu danej wzorem (4.25).

A.1.3. Selekcja i sukcesja

Przyjęto, że stosowana będzie selekcja proporcjonalna (metodą ruletki), natomiast do utworzenia nowej populacji bazowej stosowano sukcesję elitarną z liczbą najlepszych osobników równą δ_s. W zadaniu strojenia parametrów modelu neuronowego (minimalizacji funkcji przystosowania), aby zastosować selekcję proporcjonalną, niezbędne jest przeprowadzenie odpowiedniego skalowania wartości funkcji przystosowania. Modyfikacja ta polega na symetrycznym odbiciu funkcji przystosowania względem wartości średniej przystosowania osobników populacji:

$$\tilde{E}\left(\boldsymbol{\omega}^i\right) = \frac{2}{L_p}\sum_{j=1}^{L_p} E\left(\boldsymbol{\omega}^j\right) - E\left(\boldsymbol{\omega}^i\right), \tag{A.2}$$

przy czym, jeżeli

$$\Phi = \min\left(\left\{\tilde{E}\left(\boldsymbol{\omega}^i\right)\right\}_{i=1}^{L_p}\right) < 0, \tag{A.3}$$

to, aby zapewnić dodatnie wartości funkcji przystosowania dla wszystkich osobników, wylicza się dodatkowo

$$\tilde{\tilde{E}}\left(\boldsymbol{\omega}^i\right) = \tilde{E}\left(\boldsymbol{\omega}^i\right) - \Phi. \tag{A.4}$$

Ostatecznie, na podstawie przystosowania i-tego osobnika przyporządkowuje się mu prawdopodobieństwo wylosowania zgodnie ze znanym wzorem (Rutkowski, 2005):

$$p\left(\boldsymbol{\omega}^i\right) = \frac{\tilde{\tilde{E}}\left(\boldsymbol{\omega}^i\right)}{\sum\limits_{j=1}^{L_p}\tilde{\tilde{E}}\left(\boldsymbol{\omega}^j\right)}, \tag{A.5}$$

Jeżeli warunek (A.3) nie jest spełniony, to wyliczenie prawdopodobieństwa nie wymaga dodatkowej operacji.

A.1.4. Operatory ewolucyjne

Na podstawie przeprowadzonych badań wstępnych (Przystałka, 2007b) oraz ze względu na fakt, że algorytm globalny używany jest do znalezienia początkowych wartości parametrów swobodnych sieci, zdecydowano się na użycie prostego operatora krzyżowania heurystycznego. Zasadnicze badania skuteczności algorytmu prowadzono dla różnych operatorów mutacji.

Krzyżowanie heurystyczne jest operatorem wykorzystującym informację o wartościach funkcji przystosowania krzyżowanych osobników. Na podstawie dwóch chromosomów $\boldsymbol{\omega}^1$ i $\boldsymbol{\omega}^2$ w przypadku, gdy $E\left(\boldsymbol{\omega}^1\right) < E\left(\boldsymbol{\omega}^2\right)$, tworzony jest nowy chromosom zgodnie z wzorem:

$$\boldsymbol{\omega}^3 = \boldsymbol{\omega}^2 + \lambda_h\left(\boldsymbol{\omega}^1 - \boldsymbol{\omega}^2\right), \tag{A.6}$$

gdzie λ_h jest współczynnikiem określającym wpływ lepiej przystosowanego osobnika na wynik krzyżowania.

Mutacja równomierna zmienia wybrany chromosom w następujący sposób:

$$[\omega_1\ \omega_2\ \ldots\ \omega_j\ \ldots\ \omega_p] \longrightarrow \left[\omega_1\ \omega_2\ \ldots\ \omega_j^*\ \ldots\ \omega_p\right], \tag{A.7}$$

gdzie wartość genu ω_j^* losowana jest z założonym rozkładem z przedziału określonego ograniczeniami nałożonymi na daną zmienną. Zmiana wartości genu uzależniona jest od założonego prawdopodobieństwa r_m.

Mutacja nierównomierna zmienia wybrany chromosom, przy czym realizuje to w następujący sposób:

$$\boldsymbol{\omega}^i = \boldsymbol{\omega}^i + \lambda_n \boldsymbol{\rho}^i, \tag{A.8}$$

gdzie $\boldsymbol{\rho}^i$ jest wektorem o rozmiarze $p \times 1$, którego wartości losowane są z określonym rozkładem, λ_n jest wartością zmieniającą się podczas trwania procesu ewolucyjnego zgodnie z regułą $\lambda_n = \lambda_{n-1}\left(1 - \gamma_2 n/m\right)$, przy czym $\lambda_0 = \gamma_1$ oznacza początkową wartość wariancji rozkładu, natomiast γ_2 określa szybkość jej redukcji, m oznacza maksymalną liczbę pokoleń, n oznacza numer aktualnej epoki algorytmu.

Dla potrzeb mutacji równomiernej i nierównomiernej do generowania wartości wybranego genu ω_j^* i wektora $\boldsymbol{\rho}$ stosowano następujące rozkłady zmiennej losowej:

- rozkład równomierny: $\mathcal{U}(-1, 1)$,
- rozkład normalny: $\mathcal{N}(0, 1)$,

oraz układy chaotyczne:

- odwzorowanie Hénona (Awrejcewicz i Mosdorf, 2003):

$$\boldsymbol{\rho}^i = \left[\left\{ \begin{array}{ll} \tilde{\rho}_j^i = & \rho_{j-1}^i + 1 - 1.4(\tilde{\rho}_{j-1}^i)^2 \\ \rho_j^i = & 0.3\tilde{\rho}_{j-1}^i \end{array} \right\}_{j=1}^{p}\right], \tag{A.9}$$

- odwzorowanie Ikedy (Awrejcewicz i Mosdorf, 2003):

$$\boldsymbol{\rho}^i = \left[\left\{ \begin{array}{ll} r_j = & 0.4 - 6/\left(1 + (\tilde{\rho}_{j-1}^i)^2 + (\rho_{j-1}^i)^2\right) \\ \tilde{\rho}_j^i = & 1 + 0.9\left(\tilde{\rho}_{j-1}^i \cos(r_j) - \rho_{j-1}^i \sin(r_j)\right) \\ \rho_j^i = & 0.9\left(\tilde{\rho}_{j-1}^i \sin(r_j) - \rho_{j-1}^i \cos(r_j)\right) \end{array} \right\}_{j=1}^{p}\right], \tag{A.10}$$

- dyskretne równanie Mackeya-Glassa (Awrejcewicz i Mosdorf, 2003):

$$\boldsymbol{\rho}^i = \left[\left\{ \rho_j^i = \frac{\rho_{j-1}^i + 0.2\rho_{j-17}^i}{(1+\rho_{j-17}^i)^{10} - 0.1\rho_{j-1}^i} \right\}_{j=1}^{p}\right], \tag{A.11}$$

przy czym wartości początkowe układów generowano losowo z rozkładem równomiernym $\mathcal{U}(0,1)$. W przypadku chaotycznej mutacji równomiernej generowano wektor $\boldsymbol{\rho}^i$ i wybierano j-ty gen, który następnie podmieniano w chromosomie $\boldsymbol{\omega}^i$. Warunkiem zatrzymania algorytmu ewolucyjnego jest przekroczenie maksymalnej liczby pokoleń.

A.2. Algorytm symulowanego wyżarzania

Kolejnym algorytmem optymalizacji globalnej stosowanym przez autora w badaniach jest algorytm symulowanego wyżarzania. Strojenie parametrów swobodnych sieci lokalnie rekurencyjnej realizowano według schematu algorytmu zaproponowanego w pracy (Korbicz *i in.*, 1994). Aby zastosować ten algorytm w procesie uczenia sieci neuronowej, niezbędne było określenie: sposobu generowania nowych rozwiązań, sposobu redukcji temperatury wyżarzania oraz sposobu akceptacji nowego rozwiązania.

A.2.1. Generowanie nowych rozwiązań

W pracy rozpatrzono dwa sposoby generowania nowego rozwiązania proponowane w (The MathWorks, 2007) różniące się wielkością aktualnego sąsiedztwa:

1. Przestrzeń poszukiwań proporcjonalna do wartości temperatury:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n + T_n \frac{\boldsymbol{\rho}_n}{\|\boldsymbol{\rho}_n\|}; \tag{A.12}$$

2. Przestrzeń poszukiwań proporcjonalna do wartości pierwiastka z temperatury:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n + \sqrt{T_n} \frac{\boldsymbol{\rho}_n}{\|\boldsymbol{\rho}_n\|}. \tag{A.13}$$

Wektor $\boldsymbol{\rho}_n$ generowano identycznie jak dla algorytmu ewolucyjnego. W ten sposób uzyskano następujące funkcje generujące nowe rozwiązania:

- $g_N^{1/2}(\boldsymbol{\omega}_n) : \boldsymbol{\omega}_n \to \boldsymbol{\omega}_{n+1}$ - funkcja generująca z rozkładem normalnym,
- $g_H^{1/2}(\boldsymbol{\omega}_n) : \boldsymbol{\omega}_n \to \boldsymbol{\omega}_{n+1}$ - funkcja generująca z odwzorowaniem Hénona,
- $g_I^{1/2}(\boldsymbol{\omega}_n) : \boldsymbol{\omega}_n \to \boldsymbol{\omega}_{n+1}$ - funkcja generująca z odwzorowaniem Ikedy,
- $g_M^{1/2}(\boldsymbol{\omega}_n) : \boldsymbol{\omega}_n \to \boldsymbol{\omega}_{n+1}$ - funkcja generująca z równaniem Mackeya-Glassa,

A.2.2. Redukcja temperatury

Stosowano trzy funkcje redukcji temperatury, które zaimplementowane są domyślnie w oprogramowaniu MATLAB (The MathWorks, 2007):

- $t_e(T_n) : T_n = T_0(0.95)^n$ - funkcja wykładnicza,
- $t_f(T_n) : T_n = T_0/n$ - funkcja potęgowa,
- $t_b(T_n) : T_n = T_0/\log(n)$ - funkcja logarytmiczna,

A.2.3. Akceptacja nowego rozwiązania

Zastosowano standardową metodę akceptacji nowego rozwiązania, przy czym wartość prawdopodobieństwa wyznaczano z równania:

$$p(\boldsymbol{\omega}_{n+1}) = \frac{1}{1 + \exp(\Delta E_n / T_n)}, \tag{A.14}$$

gdzie przyrost energii funkcji celu definiujemy jako $\Delta E_n = E(\boldsymbol{\omega}_{n+1}) - E(\boldsymbol{\omega}_n)$. Podobnie jak poprzednio, warunkiem zatrzymania algorytmu jest przekroczenie maksymalnej liczby iteracji.

A.3. Algorytm przeszukiwania bezpośredniego

Ostatnim algorytmem optymalizacji globalnej, używanym przez autora do strojenia parametrów modeli neuronowych, jest algorytm, który bazuje na metodzie przeszukiwania bezpośredniego (ang. *direct search method*) (Lewis *i in.*, 2000). W niniejszej pracy zastosowano dwa podejścia zaliczane do algorytmów poszukiwania wzorca (ang. *pattern search*): metodę GPS (ang. *generalized pattern search*) oraz jej modyfikację metodę MADS (ang. *mesh adaptive direct search*) (Audet *i in.*, 2000; Audet i J. Dennis Jr., 2006). W każdym kroku tego algorytmu realizowane są dwie procedury:

1. Przeszukiwanie wstępne (opcjonalnie).
2. Przeszukiwanie lokalne.

Generowanie siatki w pierwszej z wymienionych faz realizowane może być z zastosowaniem różnych strategii, takich jak na przykład: metoda według schematu GPS, metoda według schematu MADS, algorytm ewolucyjny, metoda bazująca na LHS (ang. *Latin Hypercube Sampling*). W drugiej fazie, przeszukiwanie realizowane jest w otoczeniu bieżącego rozwiązania za pomocą siatki skonstruowanej z zastosowaniem metody GPS lub MADS. Do generowania siatki stosowano następujące metody:

- metoda GPS2P: $P_n = \{\boldsymbol{\omega}_n \pm \Delta_n \mathbf{e}_i\}_{i=1}^{p}$,
- metoda GPSPp1: $P_n = \{\boldsymbol{\omega}_n + \Delta_n \mathbf{e}_i\}_{i=1}^{p} \cup \{\boldsymbol{\omega}_n - \Delta_n \mathbf{1}\}$,
- metody MADS2P, MADSPp1: siatka generowana jest podobnie jak poprzednio, przy czym wzorce użyte do generowania siatki tworzone są za pomocą kolejnych kolumn dolnej macierzy trójkątnej $(p \times p)$ permutacji losowych,

gdzie $\mathbf{e}_i$ - wektor zawierający jedynkę na i-tej pozycji, $\Delta_{n+1} = \tau_s \Delta_n$, τ_s oznacza mnożnik kontrakcji/ekspansji siatki.

Zastosowano algorytm zaimplementowany w oprogramowaniu MATLAB (The MathWorks, 2007). Przyjęto następujące ograniczenia: przeszukiwanie kompletne siatki w drugiej fazie oraz sposób losowy przeszukiwania siatki. Algorytm kończy działanie po przekroczeniu maksymalnej liczby iteracji.

A.4. Algorytm największego spadku

Algorytm największego spadku jest oparty na liniowym przybliżeniu funkcji celu (Nocedal i Wright, 2006; Osowski, 2006b). W algorytmie tym nowe rozwiązanie $\boldsymbol{\omega}_{n+1}$ wyliczane jest w każdym kroku zgodnie z regułą:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \alpha_n \mathbf{d}_n = \boldsymbol{\omega}_n - \alpha_n \nabla E\left(\boldsymbol{\omega}_n\right), \tag{A.15}$$

przy czym wartość α_n dobierana jest tak, aby osiągnąć punkt minimalizujący (w przybliżeniu) funkcję $\hat{E}\left(\alpha\right) = E\left(\boldsymbol{\omega}_n + \alpha \mathbf{d}_n\right)$ ze względu na $\alpha \geq 0$. W tym celu używano bezgradientowej metody aproksymacji kwadratowej, w której początkowo wyznaczane są wartości funkcji $\hat{E}\left(\alpha\right)$ w trzech punktach takich, że $\alpha_1 < \alpha_2 < \alpha_3$. Wtedy kolejny punkt α_m wyznaczamy zgodnie z równaniem:

$$\alpha_m = \frac{1}{2} \frac{\left(\alpha_2^2 - \alpha_3^2\right) \hat{E}\left(\alpha_1\right) + \left(\alpha_3^2 - \alpha_1^2\right) \hat{E}\left(\alpha_2\right) + \left(\alpha_1^2 - \alpha_2^2\right) \hat{E}\left(\alpha_3\right)}{\left(\alpha_2 - \alpha_3\right) \hat{E}\left(\alpha_1\right) + \left(\alpha_3 - \alpha_1\right) \hat{E}\left(\alpha_2\right) + \left(\alpha_1 - \alpha_2\right) \hat{E}\left(\alpha_3\right)}, \tag{A.16}$$

przy czym początkowy przedział $\alpha_1 < \alpha_2 < \alpha_3$ wyznaczano empirycznie, $m = \{4, 5, \ldots, [p/2]\}$ oraz stosowany jest dwuskośny test Goldsteina jako test stopu (Stachurski i Wierzbicki, 2001).

A.5. Algorytm zmiennej metryki

Algorytm ten zalicza się do tzw. metod quasi-newtonowskich i jest jedną z pierwszych realizacji praktycznych metody Newtona, która opiera się na kwadratowym przybliżeniu funkcji celu (Nocedal i Wright, 2006; Osowski, 2006b). W algorytmie tym nowe rozwiązanie osiągane jest zgodnie z regułą:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \alpha_n \left[\mathbf{G}\left(\boldsymbol{\omega}_n\right)\right]^{-1} \nabla E\left(\boldsymbol{\omega}_n\right), \tag{A.17}$$

gdzie $\mathbf{G}\left(\boldsymbol{\omega}_n\right)$ jest aproksymacją macierzy hesjanu $\mathbf{H}\left(\boldsymbol{\omega}_n\right)$ funkcji celu, zaś α_n wyznaczane jest w ten sam sposób jak poprzednio.

W pracy stosowano dwie najbardziej popularne metody aproksymacji odwrotnej macierzy hesjanu uwzględniające poprawki drugiego rzędu zgodne z:

- formułą DFP (Davidon−Fletcher−Powell)

$$\mathbf{V}_n = \left[\mathbf{G}\left(\boldsymbol{\omega}_n\right)\right]^{-1} = \mathbf{V}_{n-1} + \frac{\mathbf{s}_n (\mathbf{s}_n)^T}{(\mathbf{r}_n)^T \mathbf{s}_n} - \frac{\mathbf{V}_{n-1} \mathbf{r}_n (\mathbf{r}_n)^T \mathbf{V}_{n-1}}{(\mathbf{r}_n)^T \mathbf{V}_k \mathbf{r}_n}, \tag{A.18}$$

- formułą BFGS (Broyden−Fletcher−Goldfarb−Shanno)

$$\begin{aligned} \mathbf{V}_n = \left[\mathbf{G}\left(\boldsymbol{\omega}_n\right)\right]^{-1} &= \mathbf{V}_{n-1} + \left(1 + \frac{(\mathbf{r}_n)^T \mathbf{V}_{n-1} \mathbf{r}_n}{(\mathbf{r}_n)^T \mathbf{s}_n}\right) \frac{\mathbf{s}_n (\mathbf{s}_n)^T}{(\mathbf{r}_n)^T \mathbf{s}_n} + \\ &- \frac{\mathbf{s}_n (\mathbf{r}_n)^T \mathbf{V}_n + \mathbf{V}_n \mathbf{r}_n (\mathbf{s}_n)^T}{(\mathbf{r}_n)^T \mathbf{s}_n}, \end{aligned} \tag{A.19}$$

gdzie $\mathbf{s}_n = \boldsymbol{\omega}_n - \boldsymbol{\omega}_{n-1}$ oraz $\mathbf{r}_n = \nabla E(\boldsymbol{\omega}_n) - \nabla E(\boldsymbol{\omega}_{n-1})$. Pierwsza iteracja przeprowadzana jest zgodnie z algorytmem największego spadku. Wartość startową oszacowania hesjanu przyjmowano jako $\mathbf{V}_0 = \mathbf{I}$.

A.6. Algorytm Levenberga-Marquardta

Niniejszy algorytm bazuje na metodzie regularyzacyjnej, której idea zaproponowana została przez K. Levenberga i D. Marquardta (Levenberg, 1944; Marquardt, 1963). W metodzie tej wartość dokładną hesjanu (używaną w metodzie Newtona) przybliża się za pomocą macierzy jakobianu, przy czym wymagane jest odpowiednie zapisanie funkcji celu (4.25) w postaci wektorowej:

$$\mathbf{E}(\boldsymbol{\omega}_n) = [E_q(\boldsymbol{\omega}_n)\, E_{q+1}(\boldsymbol{\omega}_n) \ldots E_Q(\boldsymbol{\omega}_n)]^T, \tag{A.20}$$

Proces uczenia realizowany jest wg reguły:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \left[\mathbf{J}^T(\boldsymbol{\omega}_n)\,\mathbf{J}(\boldsymbol{\omega}_n) + \lambda_n \operatorname{diag}\left(\mathbf{J}^T(\boldsymbol{\omega}_n)\,\mathbf{J}(\boldsymbol{\omega}_n)\right)\right]^{-1} \mathbf{J}^T(\boldsymbol{\omega}_n)\,\mathbf{E}(\boldsymbol{\omega}_n), \tag{A.21}$$

gdzie λ_n jest parametrem skalarnym określającym moc czynnika regularyzacyjnego dobieranym w każdym kroku uczenia. Podczas badań struktur lokalnie rekurencyjnych zmiany parametru λ_n prowadzono według klasycznego schematu (Osowski, 2006b):

- Jeżeli $E(\boldsymbol{\omega}_{n+1}, \lambda_p) \leq E(\boldsymbol{\omega}_{n+1}, \lambda_n)$, to przyjmuje się $\lambda_{n+1} = \lambda_p = 0.1\lambda_n$,
- Jeżeli $E(\boldsymbol{\omega}_{n+1}, \lambda_p) > E(\boldsymbol{\omega}_{n+1}, \lambda_n)$ oraz $E(\boldsymbol{\omega}_{n+1}, \lambda_n) < E(\boldsymbol{\omega}_n, \lambda_n)$, to przyjmuje się $\lambda_{n+1} = \lambda_n$,
- Jeżeli $E(\boldsymbol{\omega}_{n+1}, \lambda_p) > E(\boldsymbol{\omega}_{n+1}, \lambda_n)$ oraz $E(\boldsymbol{\omega}_{n+1}, \lambda_n) > E(\boldsymbol{\omega}_n, \lambda_n)$, to wylicza się m razy $\lambda_p = 10m\lambda_n$, aż $E(\boldsymbol{\omega}_{n+1}, \lambda_p) \leq E(\boldsymbol{\omega}_n, \lambda_n)$ co pozwoli na przyjęcie $\lambda_{n+1} = 10m\lambda_n$.

przy czym $\lambda_0 = 0.01$.

A.7. Algorytmy z losowaniem kierunku

Idea działania algorytmów opisywanych w tym punkcie polega na tym, że wybiera się losowy kierunek poszukiwań $\mathbf{d}_n$ kolejnego przybliżenia, a następnie w tak wybranym kierunku wykonuje się krok o odpowiedniej długości (wyliczanej tak jak w przypadku metody najszybszego spadku). Reguła aktualizacji wartości swobodnych parametrów sieci jest następująca:

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \alpha_n \mathbf{d}_n, \tag{A.22}$$

przy czym nowe rozwiązanie $\boldsymbol{\omega}_{n+1}$ jest akceptowane, jeżeli $E(\boldsymbol{\omega}_{n+1}) \leq E(\boldsymbol{\omega}_n) - \delta_E$. Przyjęcie ujemnej wartości progu δ_E umożliwia pogorszenie rozwiązania, co może prowadzić do omijania minimów lokalnych.

Do generowania nowych przybliżeń rozwiązania zastosowano wybrane algorytmy omówione w pracy (Zieliński i Neumann, 1986):

- **Algorytm A1** – jednakowo prawdopodobny wybór kierunku jednej z osi układu współrzędnych lub kierunku przeciwnego do jednej z tych osi:

$$\mathcal{U}\left(\left\{e_i^+\right\}_{i=1}^p \cup \left\{e_i^-\right\}_{i=1}^p\right). \tag{A.23}$$

- **Algorytm B1** – jednakowo prawdopodobny wybór jednego z wierzchołków kostki o krawędziach równoległych do odpowiednich osi układu współrzędnych, o środku w początku układu współrzędnych:

$$\mathcal{U}\left(\{(d_1, d_2, \ldots, d_p) : d_j = \pm 1, j = 1, 2, \ldots, p\}\right). \tag{A.24}$$

- **Algorytm C1** – jednakowo prawdopodobny wybór dowolnego kierunku

$$\mathcal{U}\left(\mathcal{Y}(0,1) = \{\boldsymbol{\omega} : \|\boldsymbol{\omega}\| = 1\}\right). \tag{A.25}$$

- **Algorytm D1** – biorąc za punkt wyjścia k niezależnych kierunków losowych $\mathbf{d}_{n_j}$, $j = 1, 2, \ldots, k$ (jednakowo prawdopodobny wybór kierunku zgodny z jednym z wybranych algorytmów A1, B1, C1), za $\mathbf{d}_n$ wybiera się ten z nich, który realizuje równość:

$$E(\boldsymbol{\omega}_n + \alpha_n \mathbf{d}_n) = \min_{1 \leq j \leq k} E\left(\boldsymbol{\omega}_n + \alpha_{n_j} \mathbf{d}_{n_j}\right). \tag{A.26}$$

Metody bazujące na losowym wyborze kierunku nie wymagają przyjmowania żadnych ograniczeń na funkcję celu (np. o istnieniu jej pochodnej).

A.8. Gradient i jakobian funkcji celu

Algorytmy najszybszego spadku, zmiennej metryki i Levenberga-Marquardta wymagają wyznaczania gradientu:

$$\nabla E(\boldsymbol{\omega}_n) = \left[\frac{\partial E(\boldsymbol{\omega}_n)}{\partial \omega_1} \; \frac{\partial\, E(\boldsymbol{\omega}_n)}{\partial \omega_2} \cdots \frac{\partial E(\boldsymbol{\omega}_n)}{\partial \omega_p}\right]^T, \tag{A.27}$$

lub jakobianu:

$$\mathbf{J}(\boldsymbol{\omega}_n) = \begin{bmatrix} \frac{\partial E_q(\boldsymbol{\omega}_n)}{\partial \omega_1} & \frac{\partial E_q(\boldsymbol{\omega}_n)}{\partial \omega_2} & \cdots & \frac{\partial E_q(\boldsymbol{\omega}_n)}{\partial \omega_p} \\ \frac{\partial E_{q+1}(\boldsymbol{\omega}_n)}{\partial \omega_1} & \frac{\partial E_{q+1}(\boldsymbol{\omega}_n)}{\partial \omega_2} & \cdots & \frac{\partial E_{q+1}(\boldsymbol{\omega}_n)}{\partial \omega_p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial E_Q(\boldsymbol{\omega}_n)}{\partial \omega_1} & \frac{\partial E_Q(\boldsymbol{\omega}_n)}{\partial \omega_2} & \cdots & \frac{\partial E_Q(\boldsymbol{\omega}_n)}{\partial \omega_p} \end{bmatrix}, \tag{A.28}$$

funkcji celu w każdym kroku. Do wyznaczenia elementów gradientu i jakobianu w niniejszej pracy używano dwóch znanych podejść: aproksymacji numerycznej oraz aproksymacji stochastycznej.

A.8.1. Aproksymacja numeryczna

Aproksymacja tego typu opiera się bezpośrednio na określeniu cząstkowych pochodnych (Nocedal i Wright, 2006):

- za pomocą różnic skończonych:

$$\left.\frac{\partial E(\boldsymbol{\omega})}{\partial \omega_i}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}_n} = \frac{1}{h}\left[E(\boldsymbol{\omega}_n + h\boldsymbol{e}_i) - E(\boldsymbol{\omega}_n)\right], \tag{A.29}$$

- lub za pomocą centralnych różnic skończonych:

$$\left.\frac{\partial E(\boldsymbol{\omega})}{\partial \omega_i}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}_n} = \frac{1}{2h}\left[E(\boldsymbol{\omega}_n + h\boldsymbol{e}_i) - E(\boldsymbol{\omega}_n - h\boldsymbol{e}_i)\right], \tag{A.30}$$

gdzie $\mathbf{e}_i$ oznacza i-ty wersor, h określa dokładność aproksymacji.

A.8.2. Aproksymacja stochastyczna

Oszacowanie gradientu funkcji celu zdefiniowano na podstawie reguły zaproponowanej w pracy (Spall, 2003) oraz dobrze znanej heurystyki w statystyce matematycznej, że średnia na ogół lepiej przybliża estymowaną wielkość niż pojedyncza obserwacja:

$$\left.\frac{\partial \hat{E}(\boldsymbol{\omega})}{\partial \omega_i}\right|_{\boldsymbol{\omega}=\boldsymbol{\omega}_n} = \frac{1}{kc_n}\sum_{j=1}^{k}\rho_{ni}^{j}\left[E\left(\boldsymbol{\omega}_n + c_n\boldsymbol{\rho}_n^j\right) - E(\boldsymbol{\omega}_n)\right], \tag{A.31}$$

gdzie $\boldsymbol{\rho}_n^j = \left[\rho_{n1}^j\ \rho_{n2}^j\ \ldots\ \rho_{np}^j\right]^T$ jest wektorem wygenerowanych wartości zmiennej losowej o zadanym rozkładzie beta ($\alpha = \beta = 0.1$), $c_n = c/(n+1)^{\gamma_c}$ jest czynnikiem określającym moc zakłócenia, oraz zaleca się aby $k < [p/2]$.

Ostatecznie estymator kierunku gradientu funkcji celu w n-tym kroku wylicza się na podstawie równania:

$$\nabla E(\boldsymbol{\omega}_n) \cong \frac{\nabla \hat{E}(\boldsymbol{\omega}_n)}{\left\|\nabla \hat{E}(\boldsymbol{\omega}_n)\right\|}, \tag{A.32}$$

gdzie $\|\cdot\|$ oznacza metrykę maksimum.

Dodatek B

Wybrane zagadnienia modelowania neuronowego

B.1. Reprezentacja danych

W zagadnieniach praktycznych dotyczących diagnostyki procesów wspartej modelem obiektu niezbędne jest przyjęcie odpowiedniego sposobu reprezentacji danych (zmiennych procesowych, sygnałów diagnostycznych itp.), które posłużą do budowy i weryfikacji projektowanego systemu diagnostycznego (Moczulski, 2002). Dane pierwotne, które pozyskiwane są w ramach eksperymentu diagnostycznego na obiekcie rzeczywistym lub symulatorze, zazwyczaj gromadzone są w postaci (Duch *i in.*, 2000; Korbicz *i in.*, 2004):

- szeregów czasowych zależnych od położenia sensorów w przestrzeni,
- zbiorów opisów i obserwacji,
- sekwencji symboli danego alfabetu.

Pozyskane dane są wstępnie przetwarzane i zapisywane w formie wyekstrahowanych cech. Do reprezentacji danych stosuje się tu zwykle model atrybutowy, którego formalnym zapisem jest system informacyjny (Pawlak, 1983). Zbiory danych zorganizowane w tabelaryczne systemy informacyjne zawierają zestawy wartości atrybutów wejściowych U (dla uczenia bez nadzoru) oraz ewentualnie wartości atrybutów wyjściowych Y (dla uczenia z nadzorem). Należy nadmienić w tym miejscu, że dla atrybutów, których wartości zmieniają się w czasie, stosuje się pojęcie temporalnego systemu informacyjnego (Bjorvand, 1997). Umożliwia to uporządkowanie danych względem wybranego atrybutu porządkującego. Ogólną postać temporalnego systemu informacyjnego przedstawia Tab. B.1. Możliwe jest przekształcenie temporalnego systemu informacyjnego w system informacyjny, np. w celu poprawy dokładności uzyskiwanych rozwiązań, zmniejszenia stopnia ich złożoności oraz zwiększenia czytelności (Bjorvand, 1997; Wachla, 2006). W rozprawie przyjmuje się, że do reprezentacji danych w celu oceny sprawności budowanego systemu detekcji uszkodzeń stosowany będzie system informacyjny, który określony jest za pomocą następującej dwójki:

$$\mathrm{SI} = (X, Z)\,, \tag{B.1}$$

gdzie X jest niepustym i skończonym zbiorem obiektów zwanym uniwersum, Z jest niepustym i skończonym zbiorem atrybutów wejściowych i wyjściowych. Natomiast w zagadnieniach dotyczących tworzenia neuronowych modeli procesów do reprezentacji danych używany będzie temporalny system informacyjny, którego formalny opis jest następujący:

$$\mathrm{TSI} = (X, Z \cup \{\mathcal{K}\}, \prec, \Delta k), \tag{B.2}$$

gdzie $\mathcal{K}$ jest atrybutem określającym kolejność zdarzeń, Δk oznacza odległość pomiędzy kolejnymi elementami (zdarzeniami) w $\mathcal{K}$, $\prec$ jest relacją porządkującą dla atrybutu $\mathcal{K}$.

Tab. B.1: Temporalny system informacyjny

X	$\mathcal{K}$	Z							
		U				Y			
		U_1	U_2	$\cdots$	U_L	Y_1	Y_2	$\cdots$	Y_M
x_0	k_0	u_{01}	u_{02}	$\cdots$	u_{0L}	y_{01}	y_{02}	$\cdots$	y_{0M}
x_1	k_1	u_{11}	u_{12}	$\cdots$	u_{1L}	y_{11}	y_{12}	$\cdots$	y_{1M}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_N	k_N	u_{N1}	u_{N2}	$\cdots$	u_{NL}	y_{N1}	y_{N2}	$\cdots$	y_{NM}

B.2. Wybrane metody wstępnego przetwarzanie danych

Głównym celem wstępnego przetwarzania danych (ang. *preprocessing*) jest usprawnienie procesu budowy modelu wybraną techniką modelowania. Działania wykonane podczas tego typu operacji powodują takie przekształcenie danych, które nie zawsze wpływa na lepsze zrozumienie badanego zjawiska. Dobierając metody przetwarzania wstępnego danych, należy uwzględnić sposób ich pozyskania. Dane pierwotne rejestrowane jako wyniki pomiarów obarczone są błędami pomiarowymi, szumami, zakłóceniami itp. W literaturze zalecanymi i najczęściej stosowanymi przekształceniami podczas budowy modeli procesów są (Duch *i in.*, 2000; Korbicz *i in.*, 2004):

- **Czyszczenie danych** polegające na zastąpieniu nietypowych/brakujących elementów szeregów, które zostały wskazane na podstawie obróbki statystycznej wyników pomiarów, wartościami, które są najbardziej prawdopodobne. W tym celu stosuje się różnego rodzaju metody aproksymacji, które polegają na zbudowaniu odpowiedniej funkcji aproksymującej. Najczęściej jest to tzw. wielomian uogólniony w postaci:

$$Q_n(X) = \sum_{k=0}^{n} a_k q_k(X), \tag{B.3}$$

gdzie jako funkcje bazowe $q_k(X)$ stosowane są: jednomiany, funkcje trygonometryczne lub wielomiany ortogonalne, n oznacza stopień wielomianu. Aby uzupełnić brakujące dane należy wyznaczyć wartości współczynników a_k dla poprawnych danych (x_i, z_{ij}), a następnie użyć funkcji aproksymującej do odtworzenia brakujących próbek zmiennej Z_j.

- **Filtracja cyfrowa**, której zadaniem jest eliminacja składowych takich jak: trendy, składowe oscylacyjne, składowe szybkozmienne o charakterze szumu. W tym celu stosowane są filtry nieoptymalne (o skończonej lub nieskończonej odpowiedzi impulsowej) oraz filtry optymalne (np. Wienera, Kalmana).
- **Wygładzanie** w odróżnieniu od filtracji dopuszcza wykorzystanie pomiarów przeszłych jak również przyszłych w stosunku do wygładzanego punktu. W tym celu bardzo często stosuje się metodę wygładzania wielomianami aproksymującymi np. w postaci (B.3). Optymalny wielomian powinien posiadać dwie własności: mieć stopień na tyle wysoki, by dobrze przybliżać wartości zmiennej wygładzanej oraz nie powinien odzwierciedlać zakłóceń lub niedokładności pomiarów.
- **Decymacja** o czynnik M_d polega na M_d-krotnym zmniejszeniu częstotliwości próbkowania sygnału. W pierwszym etapie decymacji przeprowadzana jest filtracja dolnoprzepustowa zapobiegająca aliasingowi, a następnie pobierana jest wyłącznie co M_d-ta próbka przefiltrowanego sygnału.
- **Standaryzacja i normalizacja danych**, której celem jest ujednolicenie zmienności wszystkich składowych. Przekształcenie polegające na standaryzacji zmiennej Z_j realizowane jest według zależności:

$$\{\tilde{z}_{ij}\}_{i=0}^{N} = \left\{ \frac{z_{ij} - \bar{Z}_j}{\sigma_{z_j}} \right\}_{i=0}^{N}, \tag{B.4}$$

gdzie Z_j to zmienna podlegająca standaryzacji, $\tilde{z}_{ij}$ - wartość zmiennej po standaryzacji, $\bar{Z}_j$ - wartość średnia, σ_{z_j} - odchylenie standardowe. Możliwe są również inne formy standaryzacji, np. względem odchylenia od wartości średniej, względem odchylenia od wartości minimalnej. Z kolei przekształcenie polegające na normalizacji zmiennej Z_j wykonywane jest zgodnie z wzorem:

$$\{\tilde{z}_{ij}\}_{i=0}^{N} = \left\{ \frac{z_{ij}}{q} \right\}_{i=0}^{N}, \tag{B.5}$$

gdzie q - współczynnik, który może przyjmować następujące wartości: średnia arytmetyczna, odchylenie standardowe, wartość maksymalna, wartość minimalna, rozstęp, wartość maksymalna bezwzględna.

- **Skalowanie danych**, którego celem jest takie przekształcenie szeregu, aby wszystkie jego elementy mieściły się w zadanym przedziale. Skalowanie do dowolnego przedziału $[a, b]$ realizowane jest w następujący sposób:

$$\{\tilde{z}_{ij}\}_{i=0}^{N} = \{\alpha z_{ij} + \beta\}_{i=0}^{N}, \tag{B.6}$$

przy czym

$$\alpha = \frac{b-a}{c-d}, \quad \text{oraz} \quad \beta = \frac{2ad - bd - ac}{c-d},$$

gdzie $c = \max(Z_j)$, $d = \min(Z_j)$. Skalowanie danych często stosowane jest w celu przystosowania wartości szeregów do przedziału zmienności wartości dopuszczalnych dla funkcji wyjścia neuronów.

B.3. Podział zgromadzonych danych

Podział danych $\mathcal{L}$ jest realizowany na różne sposoby i często zależy od typu problemu jaki rozwiązujemy. Ze względu na fakt, że metodyka dotyczy budowy modelu neuronowego, którego zadaniem jest odtworzenie dynamiki procesu (aproksymacja zależności dynamicznych), w trakcie podziału danych należy uwzględnić atrybut porządkujący (patrz podrozdział B.1). W pracy przyjęto podział danych uczących $\mathcal{L}$ na trzy podzbiory:

- zbiór danych trenujących $\mathcal{L}_T$ stosowanych w trakcie strojenia modeli,
- zbiór danych testowych $\mathcal{L}_G$ używanych do oceny jakości uogólnienia sieci, przy czym zachodzi warunek $\mathcal{L}_G \cap \mathcal{L}_T = \emptyset$,
- zbiór danych weryfikujących $\mathcal{L}_V \subset \mathcal{L}_G$ służących do sprawdzania zdolności generalizacyjnych w trakcie uczenia. Podzbiór ten wyodrębnia się ze zbioru $\mathcal{L}_G$ i używa się go do wczesnego zakończenia procesu uczenia, jeżeli błąd wyliczany na tym podzbiorze zaczyna rosnąć.

W literaturze zaleca się różne sposoby określania proporcji wymienionych podzbiorów, przy czym wszystkie podejścia zakładają reprezentatywny rozkład przykładów w zbiorach (Osowski, 2006b; Duch *i in.*, 2000). W głównej mierze metody te zależą od ilości danych, jakimi się dysponuje, ponieważ preferują większą liczebność podzbioru trenującego $\mathcal{L}_T$ niż podzbioru testowego $\mathcal{L}_G$. Biorąc pod uwagę różne sposoby doboru liczebności podzbiorów danych, w pracy przyjęto podział taki, że: $\mathrm{card}(\mathcal{L}_T) = \alpha_T \,\mathrm{card}(\mathcal{L}) = \alpha_T N$ oraz $\mathrm{card}(\mathcal{L}_G) = (1-\alpha_T)N$, przy czym α_T przyjmuje wartość z przedziału $[0.5, 0.7]$.

Bardzo często liczebność zbiorów określa się równolegle ze wstępnym doborem struktury sieci. W ten sposób w trakcie podziału danych możliwe jest stosowanie heurystyk wiążących złożoność struktury sieci i niezbędnej liczby przykładów umożliwiających osiągnięcie dobrych własności generalizacyjnych. W sytuacji bardzo małej liczby danych $N << p$ stosuje się podział wyłącznie na dwa zbiory $\mathcal{L}_T$ i $\mathcal{L}_G$, przy czym podział następuje tak, że próbki sygnałów o indeksach nieparzystych zaliczamy do zbioru trenującego, a o parzystych do zbioru testowego. Innym sposobem w tym przypadku jest wygenerowanie dodatkowych wzorców trenujących przez wtrącenie szumu do sygnałów ze zbioru $\mathcal{L}_T$ (ten sposób zalecany jest również wtedy, gdy dysponuje się licznymi podzbiorami danych, w celu polepszenia zdolności generalizacyjnych). Gdy dysponuje się liczniejszym zbiorem danych, podział następuje z uwzględnieniem atrybutu porządkującego w sposób

klasyczny (pierwsze $\alpha_T N$ próbek tworzy zbiór trenujący, a kolejne $(1-\alpha_T)N$ próbek tworzy zbiór testowy).

Podział danych w zbiorze $\mathcal{L}_G$ można zrealizować stosując na przykład regułę omówioną w pracy (Haykin, 1999). Jeżeli $\mathrm{card}(\mathcal{L}_G) < 30p$, to do podziału danych w zbiorze $\mathcal{L}_G$ można zastosować następującą zależność

$$\alpha_V = \frac{\mathrm{card}\,(\mathcal{L}_V)}{\mathrm{card}\,(\mathcal{L}_G)} = 1 - \frac{\sqrt{2p-1}-1}{2(p-1)}, \tag{B.7}$$

Dla większej liczby przykładów ($\mathrm{card}(\mathcal{L}_G) > 30p$) efektywność stosowania metody wczesnego zatrzymania procesu uczenia w celu zwiększania zdolności generalizacyjnych jest niska. Inna z reguł mówi, że liczba danych trenujących powinna być czterokrotnie większa od liczby danych weryfikujących (Osowski, 2006b).

B.4. Wstępny dobór struktury sieci

Wstępny dobór topologii sieci lokalnie rekurencyjnej, projektowanej do odwzorowania dynamiki rozpatrywanego procesu, wiąże się ze początkowym określeniem: liczby warstw ukrytych, liczby jednostek w warstwach, postaci funkcji wyjściowej neuronów danych warstw oraz koniecznością określenia struktur systemów dynamicznych zagnieżdżonych w neuronach dynamicznych. Zadaniem równoległym podczas budowy modelu neuronowego procesu jest określenie wejść i wyjść istotnych dla danego problemu. Ten aspekt budowy modelu neuronowego był rozważony w głównej części pracy.

Na podstawie przeglądu literaturowego zestawiono proste heurystyki, które mogą być stosowane do wstępnego oszacowania struktury sieci lokalnie rekurencyjnej.

Heurystyka 1

Wystarczy jedna warstwa ukryta z dostateczną liczbą nieliniowych jednostek przetwarzających do aproksymacji dowolnej nieliniowej funkcji ciągłej. Natomiast każdą nieliniową funkcję nieciągłą można aproksymować z zadaną dokładnością, przy użyciu co najmniej dwóch warstw ukrytych z odpowiednią liczbą nieliniowych jednostek przetwarzających (Duch *i in.*, 2000).

Heurystyka 2

Sieć lokalnie rekurencyjna z co najmniej dwiema warstwami ukrytymi może być użyta do aproksymacji (z zadaną dokładnością) trajektorii przestrzeni fazowej wytworzonej z zastosowaniem dowolnej funkcji ciągłej spełniającej warunek Lipschitza (Patan, 2008a).

Heurystyka 3

Aby uzyskać dobre zdolności sieci do uogólnień, liczba wzorców użytych do wyznaczenia parametrów swobodnych sieci powinna być kilkakrotnie większa (min. 10-20 razy) od

miary Vapnika-Chervonenkisa ($VCdim$). Ze względu na to, że nie istnieje prosty związek pomiędzy strukturą sieci neuronowej a miarą $VCdim$, w literaturze można znaleźć następujące oszacowanie jej granic (Hertz *i in.*, 1995):

$$2\left[\frac{n^1}{2}\right] r_{n^1} \leq VCdim \leq 2p\left(1+\log(n^1+n^2+\ldots)\right), \tag{B.8}$$

gdzie n^1 oznacza liczbę neuronów w pierwszej warstwie ukrytej, r_{n^1} oznacza wymiar wektora wejściowego, p jest całkowitą liczbą parametrów w sieci neuronowej, $n^1+n^2+\ldots$ jest całkowitą liczbą neuronów w sieci.

Heurystyka 4

Liczbę neuronów w pierwszej warstwie ukrytej szacuje się na podstawie zależności zaproponowanej w pracy (Tsang *i in.*, 2007):

$$n^1 = \frac{\mathrm{card}\left(\mathcal{L}_T\right)}{10\left(r_{n^1}+n^s\right)}, \tag{B.9}$$

gdzie $\mathrm{card}\left(\mathcal{L}_T\right)$ oznacza liczbę wzorców trenujących, r_{n^1}, n^s oznaczają odpowiednio wymiar wzorca wejściowego i wyjściowego. W kolejnych warstwach można przyjąć liczbę neuronów równą połowie neuronów z warstwy poprzedniej.

Heurystyka 5

W sytuacji, gdy dysponuje się małą liczbą wzorców trenujących, lepszym rozwiązaniem według P. Tsanga i współpracowników, jest zastosowanie zależności w postaci

$$n^1 = kr_{n^1} - 1, \tag{B.10}$$

przy czym $k = 1, 2, \ldots$ wyznacza się do momentu osiągnięcia zadowalającego odwzorowania dynamiki procesu.

Heurystyka 6

Dobrym oszacowaniem liczby neuronów pierwszej warstwy ukrytej, opartym na teorii Kołmogorowa (Osowski, 2006b) jest miara będąca średnią geometryczną liczby wejść i wyjść sieci:

$$n^1 = \sqrt{r_{n^1} n^s}, \tag{B.11}$$

przy czym w kolejnych warstwach przyjmuje się liczbę neuronów równą połowie neuronów z warstwy poprzedniej.

B.5. Wartości początkowe swobodnych parametrów sieci

Ten etap budowy modelu neuronowego danego procesu ma istotny wpływ zarówno na uzyskanie odpowiedniej dokładności końcowej aproksymatora neuronowego, jak również na przebieg procesu uczenia. Najczęściej wartości początkowe swobodnych parametrów sieci przyjmuje się w sposób losowy, generując liczby o wartościach bliskich zeru i określonym rozkładzie (Duch *i in.*, 2000; Korbicz *i in.*, 1994; Rutkowska *i in.*, 1997; Rutkowski, 2005). Takie działanie wymaga jednak stosowania tzw. metody wielokrotnego startu. Istnieją również metody bardziej zaawansowane, które oparte są np. na metodach grupowania danych (Duch *i in.*, 1997). Innym sposobem rozwiązania tego problemu mogą być metody uwzględniające wiedzę o modelowanym procesie (Wieczorek, 2008).

Do generowania początkowych wartości parametrów rozpatrywanych struktur lokalnie rekurencyjnych użyto znanych metod, wprowadzając następującą modyfikację. Parametry, które dotyczą połączeń mających charakter sprzężeń zwrotnych, przyjmują wartość równą zeru (Patan, 2008b). Takie działanie gwarantuje stabilność modelu w początkowej fazie uczenia. Pozostałe wartości parametrów modelu generuje się z zastosowaniem następujących metod:

- **Metoda INIT0**, która zaproponowana została przez K. Patana do inicjalizacji wartości parametrów sieci lokalnie rekurencyjnej. Wartości pozostałych parametrów generuje się losowo z rozkładem jednostajnym $\mathcal{U}\left(-0.7, 0.7\right)$.
- **Metoda INIT1**, która bazuje na metodzie zaproponowanej przez C. Bishopa do inicjalizacji wartości parametrów sieci jednokierunkowych. Pozostałe wartości parametrów danego neuronu generowane są z rozkładem normalnym $\mathcal{N}\left(0, \sqrt{n}\right)$, gdzie n oznacza liczbę wejść neuronu.
- **Metoda INIT2**, w której wartości parametrów neuronów warstw ukrytych sieci przyjmuje się zgodnie z regułą zaproponowaną przez G. Thimma i E. Fieslera: $\omega \sim \mathcal{U}\left(-\sqrt{0.25n}, \sqrt{0.25n}\right)$, gdzie n oznacza liczbę wejść danego neuronu. Natomiast wartości parametrów neuronów wyjściowych są zerowane.
- **Metoda INIT3**, będąca modyfikacją metody zaproponowanej w pracy (Duch *i in.*, 1997), gdzie wartości parametrów neuronów generowane są z rozkładem równomiernym $\mathcal{U}\left(-a/\sqrt{n}, a/\sqrt{n}\right)$, przy czym a oznacza stopień krzywizny funkcji wyjścia danego neuronu, n to liczba wejść tego neuronu.
- **Metoda INIT4**, która jest adaptacją reguły D. Nguyena i B. Widrowa. Wartości początkowe parametrów generuje się zgodnie z rozkładem $\mathcal{U}\left(-\sqrt[n]{n^i}, \sqrt[n]{n^i}\right)$, gdzie n oznacza liczbę wejść neuronu warstwy ukrytej, n^i to liczba neuronów danej warstwy ukrytej. Wartości parametrów neuronów warstwy wyjściowej przyjmuje się losowo zgodnie z rozkładem $\mathcal{U}\left(-0.5, 0.5\right)$.

Bibliografia

Abarbanel H. (1996): *Analysis of observed chaotic data*. Springer-Verlag New York, Inc.

Aihara K. (2002): *Chaos engineering and its application to parallel distributed processing with chaotic neural networks*. Proceedings of the IEEE **90**(5): pp. 919–930.

Aihara K., Takabe T. i Toyoda M. (1990): *Chaotic neural networks*. Physics Letters A **144**(6-7): pp. 333–340.

Anishchenko V. S., Astakhov V., Neiman A., Vadivasova T. i Schimansky-Geier L. (2007): *Nonlinear Dynamics of Chaotic and Stochastic Systems*. Springer Series in Synergetics. Springer-Verlag Berlin/Heidelberg.

Audet C., Charles Audet Jr., J. Dennis Jr. i Dennis J. E. (2000): *Analysis of generalized pattern searches*. SIAM Journal on Optimization **13**: pp. 889–903.

Audet C. i J. Dennis Jr. (2006): *Mesh adaptive direct search algorithms for constrained optimization*. SIAM Journal on Optimization **17**(1): pp. 188–217.

Awrejcewicz J. i Mosdorf R. (2003): *Analiza numeryczna wybranych zagadnień dynamiki chaotycznej*. Wydawnictwo Naukowo-Techniczne. Warszawa.

Ayoubi M. (1994): *Nonlinear dynamic systems identification with dynamic neural networks for fault diagnosis in technical processes*. Humans, Information and Technology. Tom 3: pp. 2120–2125.

Ayoubi M. i Isermann R. (1997): *Neuro-fuzzy systems for diagnosis*. Fuzzy Sets and System **89**(3): pp. 289–307.

Bajramovic F., Gruber C. i Sick B. (2004): *A comparison of first- and second-order training algorithms for dynamic neural networks*. IEEE International Joint Conference on Neural Networks. Tom 2: pp. 837–842.

Barczak S. i Biolik J. (2003): *Podstawy ekonometrii*. Wydawnictwo Akademii Ekonomicznej w Katowicach.

Bartyś M. (2001): *Specification of actuators intended to use for benchmark definition*. Strona projektu DAMADICS (dn. 27.02.2009): `http://diag.mchtr.pw.edu.pl/damadics/`.

Bartyś M., Patton R., Syfert M., Salvador de las Heras i Quevedo J. (2006): *Introduction to the DAMADICS actuator FDI benchmark study*. Control Engineering Practice **14**(6): pp. 577–596.

Bartyś M. i Syfert M. (2002): *Data file description, Ver. 02march2002*. Strona projektu DAMADICS (dn. 12.01.2009): `http://diag.mchtr.pw.edu.pl/damadics/`.

Bednarski M., Cholewa W. i Frid W. (2004): *Identification of sensitivities in bayesian networks*. Engineering Applications of Artificial Intelligence **17**(4): pp. 327–335.

Berenyi P., Horvath G., Pataki B. i Strausz G. (2001): *Hybrid-neural modeling of a complex industrial process*. Instrumentation and Measurement Technology Conference. IEEE: pp. 1424–1429.

Bi-qiang Du, Tang Gui-ji i Song-ling Wang (2008): *Fractal fault diagnosis of rotor system based on morphological de-nosing*. Proceedings of the 2008 Congress on Image and Signal Processing. IEEE Computer Society. Washington, DC, USA: pp. 161–165.

Birx D. i Pipenberg S. (1992): *Chaotic oscillators and complex mapping feed forward networks (CMFFNs) for signal detection in noisy environments*. International Joint Conference on Neural Networks: pp. 881–888.

Bishop C. M. (1995): *Neural Networks for Pattern Recognition*. Oxford University Press.

Bjorvand A. (1997): *Mining time series using rough sets - A case study*. Tom 1263 z *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg: pp. 351–358.

Blanke M., Kinnaert M., Lunze J. i Staroświecki M. (2006): *Diagnosis and Fault-Tolerant Control*. Springer-Verlag Berlin Heidelberg.

Boguś P. i Merkisz J. (2005): *Misfire detection of locomotive diesel engine by non-linear analysis*. Mechanical Systems and Signal Processing **19**(4): pp. 881–899.

Box G. E. i Jenkins G. M. (1983): *Analiza szeregów czasowych*. Państwowe Wydawnictwo Naukowe. Warszawa.

Caccavale F. i Villani L. (2003): *Fault Diagnosis and Fault Tolerance for Mechatronic Systems: Recent Advances*. Springer Tracts in Advanced Robotics. Springer Berlin/Heidelberg.

Calado J., J.M.G. Sa da Costa, Bartyś M. i Korbicz J. (2006): *FDI approach to the DAMADICS benchmark problem based on qualitative reasoning coupled with fuzzy neural networks*. Control Engineering Practice **14**: pp. 685–698.

Cempel C. (1989): *Diagnostyka wibroakustyczna maszyn*. Państwowe Wydawnictwo Naukowe. Warszawa.

Chang W. i Mak M. (1999): *A conjugate gradient learning algorithm for recurrent neural networks - what it does and how to do it*. Neurocomputing **24**(1): pp. 173–189.

Chen J. i Patton R. J. (1998): *Robust Model-Based Fault Diagnosis for Dynamic Systems*. Kluwer Academic Publishers. Norwell, MA, USA.

Cholewa W. (1983): *Metoda diagnozowania maszyn z zastosowaniem zbiorów rozmytych*. Zeszyty naukowe, nr 764. Politechnika Śląska. Gliwice.

Cholewa W. (1996): *The pros and cons of neural networks*. International Conference on Computer Integrated Manufacturing: pp. 43–48.

Cholewa W. (2006): *Sieci stwierdzeń w diagnostyce technicznej*. Diagnostyka **38**(2): pp. 121–129.

Cholewa W. (2008): *Mechanical analogy of statement networks*. International Journal of Applied Mathematics and Computer Science **18**(4): pp. 477–486.

Cholewa W. i Kiciński J. (1997): *Diagnostyka techniczna. Odwrotne modele diagnostyczne*. Wydawnictwo Politechniki Śląskiej. Gliwice.

Cholewa W., Kosmowski K. i Radkowski S. (2008): *Modele systemów oceny ryzyka i diagnostyki technicznej*. Zeszyty naukowe, nr 138. Katedra Podstaw Konstrukcji Maszyn, Politechnika Śląska. Gliwice.

Cholewa W. i Moczulski W. (1993): *Diagnostyka techniczna maszyn. Pomiary i analiza sygnałów*. Skrypty uczelniane, nr 1758. Wydawnictwo Politechniki Śląskiej. Gliwice.

Cholewa W. i Pedrycz W. (1987): *Systemy doradcze*. Skrypty uczelniane, nr 1447. Politechnika Śląska. Gliwice.

Cholewa W. i Rogala T. (2008): *Modele odwrotne i modelowanie diagnostyczne*. Zeszyty naukowe, nr 136, wydanie drugie. Katedra Podstaw Konstrukcji Maszyn, Politechnika Śląska. Gliwice.

Ciupke K. (2007): *Method of choosing the type of the heuristic modelling*. W: Methods of Artificial Intelligence, T. Burczyński, W. Cholewa i W. Moczulski (Red.). AI-METH series. Gliwice: pp. 21–22.

Clouse D., Giles C., Horne B. i Cottrell G. (1997): *Time-delay neural networks: representation and induction of finite-state machines*. Neural Networks **8**(5): pp. 1065–1070.

Cun Y. L., Denker J. S. i Solla S. A. (1990): *Optimal brain damage*. Advances in Neural Information Processing Systems. Morgan Kaufmann: pp. 598–605.

DAMADICS (2002): *Research Training Network - Development and Application of Methods for Actuator Diagnosis in Industrial Control Systems*. Strona projektu RTN-DAMADICS (dn. 20.02.2008): `http://diag.mchtr.pw.edu.pl/damadics/`.

De Jesus O. i Hagan M. (2001): *Backpropagation through time for a general class of recurrent network*. International Joint Conference on Neural Networks: pp. 2638–2643.

De Jesus O. i Hagan M. (2007): *Backpropagation algorithms for a broad class of dynamic networks*. IEEE Transactions on Neural Networks: pp. 14–27.

De Moor B. (2009): *DaISy: Database for the Identification of Systems*. Department of Electrical Engineering, ESAT/SISTA, K.U.Leuven, Belgium, [97-003] Data from a test setup of an industrial winding process. Strona projektu (dn. 10.01.2009): `http://homes.esat.kuleuven.be/~smc/daisy/`.

Ding S. X. (2008): *Model-based Fault Diagnosis Techniques. Design Schemes, Algorithms, and Tools*. Springer.

Ditto W., Murali K. i Sinha S. (2009): *Construction of a chaotic computer chip*. W: Applications of Nonlinear Dynamics. Model and Design of Complex Systems, V. In, P. Longhini i A. Palacios (Red.). Springer-Verlag Berlin Heidelberg: pp. 3–13.

Drapała J., Świątek J. i Brzostowski K. (2008): *Stable learning algorithm of global neural network for identification of dynamic complex systems*. W: Artificial Intelligence and Soft Computing, L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh i J. M. Żurada (Red.). Tom 5097 z *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg: pp. 123–134.

Duch W., Adamczak R. i Jankowski N. (1997): *Initialization and optimization of multilayered perceptrons*. 3rd Conference on Neural Networks and Their Applications: pp. 105–110.

Duch W., Korbicz J., Rutkowski L. i Tadeusiewicz R. (2000): *Sieci neuronowe*. Tom 6 z *Biocybernetyka i inżynieria biomedyczna*. Akademicka Oficyna Wydawnicza EXIT. Warszawa.

Eckmann J., Kamphorst S. i Ruelle D. (1987): *Recurrence plots of dynamical systems*. Europhysics Letters (5): pp. 973–977.

Elman J. L. (1990): *Finding structure in time*. Cognitive Science **14**(2): pp. 179–211.

Fradkov A. i Evans R. (2005): *Control of chaos: Methods and applications in engineering*. Annual Reviews in Control **29**: pp. 33–56.

Frank P. i Koppen-Seliger B. (1997): *Fuzzy logic and neural network applications to fault diagnosis*. International Journal of Approximate Reasoning **16**(1): pp. 67–88.

Frasconi P., Gori M. i Soda G. (1992): *Local feedback multilayered networks*. Neural Computing **4**(1): pp. 120–130.

Garfinkel A., Spano M., Ditto W. i Weiss J. (1992): *Controlling cardiac chaos*. Science **257**: pp. 1230–1235.

Garzon M. i Botelho F. (1999): *Dynamical approximation by recurrent neural networks*. Neurocomputing **29**(1): pp. 25–46.

Goldberger A. (1992): *Chaos and fractals in human physiology*. Scientific American **226**(2): pp. 42–49.

Graham-Rowe D. (2009): *Logic from chaos. New chips use chaos to produce potentially faster, more robust computing*. Strona projektu ChaoLogix (dn. 08.03.2009): `http://www.chaologix.com/`.

Griño R., Cembrano G. i C.Torras (2000): *Nonlinear system identification using additive dynamic neural networks - two on-line approaches*. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. Tom 47. IEEE: pp. 150–165.

Gruber C. i Sick B. (2003): *Fast and efficient second-order training of the dynamic neural network paradigm*. Proceedings of the International Joint Conference on Neural Networks. Tom 4. IEEE: pp. 2482–2487.

Gupta M. M., Jin L. i Homma N. (2003): *Static and Dynamic Neural Networks: From Fundamentals to Advanced Theory*. John Wiley and Sons, Inc. Hoboken, New Jersey.

Gutenbaum J. (2003): *Modelowanie matematyczne systemów*. Akademicka Oficyna Wydawnicza EXIT. Warszawa.

Hagan M. T., Demuth H. B. i Beale M. H. (1995): *Neural Network Design*. PWS Publishing Company.

Haykin S. (1999): *Neural Networks: A Comprehensive Foundation*. 2nd edition. Prentice Hall International.

Hertz J., Krogh A. i Palmer G. (1995): *Wstęp do teorii obliczeń neuronowych*. Wydawnictwo Naukowo-Techniczne. Warszawa.

Hirota K. (1995): *Fuzzy-neuro-chaos: research and industrial applications in Japan*. Systems, Man and Cybernetics **3**: pp. 2446–2459.

Iokibe T. (1997): *Industrial application of chaos engineering*. W: Soft Computing in Engineering Design and Manufacturing, R. Rajkumar i P. Chawdhry (Red.). Springer-Verlag New York, LLC: pp. 141–150.

Isermann R. (2005): *Model-based fault-detection and diagnosis – status and applications*. Annual Reviews in Control **29**(1): pp. 71–85.

Isermann R. (2006): *Fault-Diagnosis Systems. An Introduction from Fault Detection to Fault Tolerance*. Springer.

Isermann R. i Ballé P. (1997): *Trends in the application of model-based fault detection and diagnosis of technical processes*. Control Engineering Practice **5**(5): pp. 709–719.

Janczak A. (2003): *Identification of Wiener and Hammerstein systems with neural network and polynomial models: methods and applications*. University of Zielona Góra Press.

Jang J., Sun C.-T. i Mizutani E. (1997): *Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence*. Prentice Hall. US ed edition.

Jankowski N. (2003): *Ontogeniczne sieci neuronowe. O sieciach zmieniających swoją strukturę*. Akademicka Oficyna Wydawnicza EXIT. Warszawa.

Jarvis P., Wilson I. i Kemp S. (2006): *The application of a new attribute selection technique to the forecasting of housing value using dependence modelling*. Neural Computing and Applications **15**(2): pp. 136–153.

Jones A. J. (2004): *New tools in non-linear modelling and prediction*. Computational Management Science **1**(2): pp. 109–149.

Jones A. J., Evans D. i Kemp S. (2007): *A note on the Gamma test analysis of noisy input/output data and noisy time series*. Physica D: Nonlinear Phenomena **229**(1): pp. 1–8.

Jordan M. I. (1986): *Serial order: A parallel distributed processing approach. Technical report*. California Univ., San Diego, La Jolla. Inst. for Cognitive Science.

Kaczorek T. (1999): *Teoria sterowania i systemów*. Wydawnictwo Naukowe PWN. Warszawa.

Kaczorek T., Dzieliński A., Dąbrowski W. i Łopatka R. (2005): *Podstawy teorii sterowania*. WNT. Warszawa.

Kamiya A., Ovaskab S. J., Roy R. i Kobayashi S. (2005): *Fusion of soft computing and hard computing for large-scale plants: a general model*. Applied Soft Computing **5**(3): pp. 265–279.

Kantz H. i Schreiber T. (1999): *Nonlinear Time Series Analysis*. Cambridge University Press.

Kapitaniak T. i Wojewoda J. (1994): *Bifurakcje i chaos*. Politechnika Łódzka. Łódź.

Kemp S. E., Wilson I. D. i J. Andrew Ware (2005): *A Tutorial on the Gamma Test*. International Journal of Simulation Systems, Science and Technology **6**(1-2): pp. 67–75.

Keogh E., Xi X., Wei L. i Ratanamahatana C. (2009): *UCR Time Series Classification/Clustering Page*. University of California - Computer Science & Engineering Department, Riverside CA, Strona projektu (03.02.2009): `http://www.cs.ucr.edu/~eamonn/time_series_data/`.

Kiciński J. (Red.) (2005): *Modelowanie i diagnostyka oddziaływań mechanicznych, aerodynamicznych i magnetycznych w turbozespołach energetycznych*. Wyd. Instytutu Maszyn Przepływowych PAN.

Konishi S. i Kitagawa G. (2008): *Information Criteria and Statistical Modeling*. Springer Science+Business Media, LLC.

Korbicz J. (2006): *Robust fault detection using analytical and soft computing methods*. Bulletin of the Polish Academy of Sciences : Technical Sciences **54**(1): pp. 75–88.

Korbicz J., Kościelny J. M., Kowalczuk Z. i Cholewa, W. (Red.) (2004): *Fault diagnosis. Models, artificial intelligence, applications*. Springer Berlin/Heidelberg.

Korbicz J. i Kowal M. (2007): *Neuro-fuzzy networks and their application to fault detection of dynamical systems*. Engineering Applications of Artificial Intelligence **20**(5): pp. 609–617.

Korbicz J. i Mrugalski M. (2008): *Confidence estimation of GMDH neural networks and its application in fault detection systems*. International Journal of Systems Science **39**(8): pp. 783–800.

Korbicz J., Obuchowicz A. i Uciński D. (1994): *Sztuczne sieci neuronowe. Podstawy i zastosowania*. Akademicka Oficyna Wydawnicza PLJ. Warszawa.

Korbicz J. i Witczak M. (2005): *Uncertain soft computing models in robust fault detection*. Networked control systems and fault tolerant control: 1st NeCST Workshop. [B. m.]. Ajaccio, France: pp. 199–204.

Kościelny J. M. (2001): *Diagnostyka zautomatyzowanych procesów przemysłowych*. Akademicka Oficyna Wydawnicza EXIT. Warszawa.

Kościelny J. M. i Bartyś M. (2004): *Models in diagnostics of industrial processes*. Diagnostyka **30**: pp. 291–296.

Kościelny J. M., Bartyś M., Rzepiejewski P. i da Costa J. S. (2006): *Actuator fault distinguishability study for the DAMADICS benchmark problem*. Control Engineering Practice **14**(6): pp. 645–652.

Kowal M. (2005): *Optimization of Neuro-Fuzzy Structures in Technical Diagnostics Systems*. University of Zielona Góra Press.

Krishnaiah J., Kumar C. i Farugi M. (2006a): *Modelling and control of chaotic processes through their bifurcation diagrams generated with the help of recurrent neural network models. Part 1: Simulation studies*. Journal of Process Control **16**(1): pp. 53–66.

Krishnaiah J., Kumar C. i Farugi M. (2006b): *Modelling and control of chaotic processes through their bifurcation diagrams generated with the help of recurrent neural network models. Part 2: An industrial study*. Journal of Process Control **16**(1): pp. 67–79.

Kuschewski J., Hui S. i Żak S. (1993): *Application of feedforward neural networks to dynamical system identification and control*. IEEE Transactions on Control Systems Technology. Tom 1. IEEE: pp. 37–49.

Lehtoranta J. i Koivo H. (2005): *Fault diagnosis of induction motors with dynamical neural networks*. Systems, Man and Cybernetics. IEEE Press: pp. 2979–2984.

Lee C.-H. i Ching-Cheng T. (2000): *Identification and control of dynamic systems using recurrent fuzzy neural networks*. IEEE Transactions on Fuzzy Systems: pp. 349–366.

Leontitsis A. (2009): *Chaotic Systems Toolbox. Matlab functions about the simulation of chaotic systems.* Department of Education, University of Ioannina, Greece. Strona domowa autora (dn. 17.04.2009): `http://www.geocities.com/CapeCanaveral/Lab/1421/`.

Levenberg K. (1944): *A method for the solution of certain non-linear problems in least squares*. Quart. Appl. Math. **2**: pp. 164–168.

Lewis R. M., Torczon V. i Trosset M. W. (2000): *Direct search methods: then and now*. Journal of Computational and Applied Mathematics **124**(1-2): pp. 191–207.

Li C. i Cheng K.-H. (2007): *Recurrent neuro-fuzzy hybrid-learning approach to accurate system modeling*. Fuzzy Sets and Systems **158**(2): pp. 194–212.

Li C. i Qu L. (2007): *Applications of chaotic oscillator in machinery fault diagnosis*. Mechanical Systems and Signal Processing **21**(1): pp. 257–269.

Li Z. (2006): *Fuzzy chaotic systems: Modeling, control and applications*. Tom 199 z *Studies in Fuzziness and Soft Computing*. Springer-Verlag Berlin/Heidelberg.

Li Z., Halang W. A. i Chen G. (2006): *Integration of Fuzzy Logic and Chaos Theory*. Studies in Fuzziness and Soft Computing. Springer-Verlag Berlin/Heidelberg.

Liang Jin, Nikiforuk P. i Gupta M. (1995): *Approximation of discrete-time state-space trajectories using dynamic recurrent neural networks*. Automatic Control. Tom 40. IEEE: pp. 1266–1270.

Lin D., Dayhoff J. E. i Ligomenides P. A. (1992): *Trajectory recognition with a time-delay neural network*. International Joint Conference on Neural Networks. Tom 3: pp. 197–202.

Lingras P. (1998): *Comparison of neofuzzy and rough neural networks*. Information Sciences: an International Journal **110**(4): pp. 207–215.

Lisheng Wang i Zongben Xu (2006): *Sufficient and necessary conditions for global exponential stability of discrete-time recurrent neural networks*. Circuits and Systems I: Regular Papers. Tom 53. IEEE.

Liu H., Tuo H. i Liu Y. (2004): *Rough neural network of variable precision*. Neural Processing Letters **19**(1): pp. 73–87.

Łęski J. (2008): *Systemy neuronowo-rozmyte*. Wydawnictwo Naukowo-Techniczne.

MacKay D. J. C. (1992): *A practical bayesian framework for backpropagation networks*. Neural Comput. **4**(3): pp. 448–472.

Mandic D. P. i Chambers J. A. (2001): *Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability*. Adaptive and Learning Systems for Signal Processing, Communications and Control Series. John Wiley and Sons Ltd.

Marquardt D. (1963): *An algorithm for least-squares estimation of nonlinear parameters*. SIAM J. Appl. Math. **11**: pp. 431–441.

Marques F., Souza R., Rebolho D., Caporali A. i Belo E. (2005): *Application of time-delay neural and recurrent neural networks for the identification of a hingeless helicopter blade flapping and torsion motions*. J. of the Braz. Soc. of Mech. Sci. and Eng. **27**(2): pp. 97–103.

Marwan N. (2009): *Cross Recurrence Plot Toolbox. Version 5.14 (R27.1)*. Potsdam University, Germany, Strona projektu (dn. 17.03.2009): `http://www.agnld.uni-potsdam.de`.

Marwan N., Kurths J. i Saparin P. (2007): *Generalised recurrence plot analysis for spatial data*. Physics Letters A **360**: pp. 545–551.

Marwan N., Romano M. C., Thiel M. i Kurths J. (2007): *Recurrence plots for the analysis of complex systems*. Physics Reports **438**: pp. 237–329.

Mastorocostas P. A. i Theocharis J. B. (2002): *A recurrent fuzzy-neural model for dynamic system identification*. IEEE Transactions on Systems, Man and Cybernetics. Tom 32: pp. 176–190.

Maydl W. i Sick B. (2000): *Recurrent and non-recurrent dynamic network paradigms: a case study*. Proceedings of the International Joint Conference on Neural Networks. Tom 6. IEEE. Italy: pp. 73–78.

Medsker L. R. i Jain L. C. (1999): *Recurrent Neural Networks: Design and Applications*. CRC.

Moczulski W. (2002): *Diagnostyka techniczna. Metody pozyskiwania wiedzy*. Wydawnictwo Politechniki Śląskiej. Gliwice.

Moczulski W. (2005a): *Methodology of heuristic modelling of dynamic objects and processes for diagnostics and control*. W: Recent Developments in Artificial Intelligence Methods, W. Cholewa, T. Burczyński i W. Moczulski (Red.). Gliwice: pp. 123–126.

Moczulski W. (2005b): *Metodyka heurystycznego modelowania obiektów i procesów dynamicznych w diagnostyce i sterowaniu*. Pomiary, Automatyka, Kontrola. Tom 9BIS: pp. 39–42.

Moczulski W. i Hanzel M. (2007): *System diagnostyki małych silników prądu stałego z wykorzystaniem metod identyfikacji parametrów elektromechanicznych silnika*. W: Diagnostyka Procesów i Systemów, J. Korbicz, K. Patan i M. Kowal (Red.): pp. 251–262.

Moczulski W. i Szulim R. (2004): *On case-based control of dynamic industrial processes with the use of fuzzy representation*. Engineering Applications of Artificial Intelligence **17**(4): pp. 371–381.

Moon F. C. (2004): *Chaotic Vibrations. An Introduction for Applied Scientists and Engineers*. John Wiley and Sons, Inc.

Moon F. C. i Kalmár-Nagy T. (2001): *Nonlinear models for complex dynamics in cutting materials*. Phil. Trans. R. Soc. Lond. A **359**: pp. 695–711.

Morrison F. (1996): *Sztuka modelowania układów dynamicznych - deterministycznych, chaotycznych, stochastycznych*. Wydawnictwo Naukowo-Techniczne. Warszawa.

Mrugalski M., Witczak M. i Korbicz J. (2008): *Confidence estimation of the multi-layer perceptron and its application in fault detection systems*. Engineering Applications of Artificial Intelligence **21**: pp. 895–906.

Narendra K. S. i Parthasarathy K. (1990): *Identification and control of dynamic systems using neural networks*. IEEE Transactions on Neural Networks. IEEE: pp. 4–27.

Nguyen D. i Widrow B. (1990): *Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights*. IJCNN International Joint Conference on Neural Networks: pp. 21–26.

Nichols J., Trickey S. i Seaver M. (2006): *Damage detection using multivariate recurrence quantification analysis*. Mechanical Systems and Signal Processing **20**(2): pp. 421–437.

Nocedal J. i Wright S. (2006): *Numerical Optimization*. Springer Science+Business Media, LLC.

Noura H. i Bastogne T. (1997): *Tension optimal control of a multivariable winding process*. Proceedings of the American Control Conference: pp. 2499–2503.

Obuchowicz A. i Korbicz J. (2004): Evolutionary methods in designing diagnostic systems. W: Fault diagnosis: models, artificial intelligence, applications, J. Korbicz, J. M. Kościelny, Z. Kowalczuk i W. Cholewa (Red.). Springer-Verlag. Berlin: pp. 301–331.

Osowski S. (1996): *Sieci neuronowe w ujęciu algorytmicznym*. Wydawnictwo Naukowo-Techniczne. Warszawa.

Osowski S. (2006a): *Modelowanie i symulacja układów i procesów dynamicznych*. Oficyna Wydawnicza Politechniki Warszawskiej.

Osowski S. (2006b): *Sieci neuronowe do przetwarzania informacji*. Oficyna Wydawnicza Politechniki Warszawskiej.

Osowski S. i Cichocki A. (1999): *Learning in dynamic neural networks using signal flow graphs*. International Journal of Circuit Theory and Applications **27**(2): pp. 209–228.

Ott E. (1997): *Chaos w układach dynamicznych*. Wydawnictwo Naukowo-Techniczne. Warszawa.

Ovaska S., Kamiya A. i YangQuan Chen (2006): *Fusion of soft computing and hard computing: computational structures and characteristic features*. IEEE Transactions on Systems, Man, and Cybernetics. Tom 36. IEEE: pp. 439–448.

Pasemann F. (1997): *A simple chaotic neuron*. Physica D **104**: pp. 205–211.

Patan K. (2007): *Stability analysis and the stabilization of a class of discrete-time dynamic neural networks*. IEEE Transactions on Neural Networks.

Patan K. (2008a): *Approximation of state-space trajectories by locally recurrent globally feed-forward neural networks*. Neural Networks **21**(1): pp. 59–64.

Patan K. (2008b): *Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes*. Tom 377 z *Lecture Notes in Control and Information Sciences*. Springer Berlin/Heidelberg.

Patan K. i Parisini T. (2002): *Stochastic learning methods for dynamic neural networks: simulated and real-data comparisons*. Proceedings of the American Control Conference: pp. 2577–2582.

Patan K., Witczak M. i Korbicz J. (2008): *Towards robustness in neural network based fault diagnosis*. International Journal of Applied Mathematics and Computer Science **18**(4): pp. 443–454.

Patton R. J., Frank P. M. i Clark R. N. (2000): *Issues of Fault Diagnosis for Dynamic Systems*. Springer-Verlag Berlin and Heidelberg.

Patton R., Uppal F. i Lopez-Toribio C. (2000): *Soft computing approaches to fault diagnosis for dynamic systems: A survey*. IFAC Symposium SAFEPROCESS: pp. 298–311.

Pawlak Z. (1983): *System informacyjny: podstawy teoretyczne*. Wydawnictwa Naukowo-Techniczne. Warszawa.

Peitgen H., Jürgens H. i Saupe D. (1995): *Granice chaosu. Fraktale*. Wydawnictwo Naukowe PWN. Warszawa.

Perruquetti W. i Barbot J.-P. (2005): *Chaos in Automatic Control*. Control Engineering. A Series of Reference Books and Textbooks. CRC Taylor and Francis.

Pokropińska A., Nowicki R. i Scherer R. (2006): *Isolines of statistical information criteria for relational neuro-fuzzy system design*. W: Artificial Intelligence and Soft Computing, L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh i J. M. Żurada (Red.). Lecture Notes in Computer Science. Springer Berlin/Heidelberg: pp. 288–296.

Poznyak A., Wen Y. i Sanchez E. (1999): *Identification and control of unknown chaotic systems via dynamic neural networks*. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. Tom 46: pp. 1491–1495.

Prudêncio R. i Ludermir T. (2001): *Neural Network Hybrid Learning: Genetic Algorithms and Levenberg-Marquardt*. CiteSeer.IST - Copyright Penn State and NEC.

Przystałka P. (2006): *Heuristic modeling of objects and processes using dynamic neural networks*. Diagnostyka **2/38**: pp. 15–18.

Przystałka P. (2007a): *Heuristic modeling using recurrent neural networks: simulated and real-data experiments*. Computer Assisted Mechanics and Engineering Sciences **14**(4): pp. 715–727.

Przystałka P. (2007b): *Hybrid learning algorithm for locally recurrent neural networks*. W: Fault diagnosis and fault tolerant control, J. Korbicz, K. Patan i M. Kowal (Red.). Academic Publishing House EXIT. Warsaw: pp. 255–262.

Przystałka P. (2008): *Model-based fault detection and isolation using locally recurrent neural networks*. W: Artificial Intelligence and Soft Computing, L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh i J. M. Żurada (Red.). Tom 5097 z *Lecture Notes in Computer Science*. Springer Berlin/Heidelberg: pp. 123–134.

Przystałka P. (2009): *Fault detection and isolation for dynamic processes using recurrent neural networks*. Diagnostyka **49**(1): pp. 33–40.

Przystałka P. i Moczulski W. (2005): *Koncepcja metodyki tworzenia heurystycznych modeli obiektów przemysłowych*. Pomiary, Automatyka, Kontrola: pp. 136–138.

Puig V., Witczak M., Nejjari F., Quevedo J. i Korbicz J. (2007): *A GMDH neural network-based approach to passive robust fault detection using a constraint satisfaction backward test*. Engineering Applications of Artificial Intelligence **20**(7): pp. 886–897.

Qipeng L., Xiaoling Y. i Quanke F. (2003): *Fault diagnosis using wavelet neural networks*. Neural Processing Letters **18**(2): pp. 115–123.

Radziszewski B., Arczewski B., Kapitaniak T., Nizioł J., Marynia J., Osiecki J. i Książek M. (2005): *Dynamika układów mechanicznych*. Tom II. Komitet Mechaniki PAN - IPPT PAN. Warszawa.

Rosenstein M., Collins J. i C.J. De Luca (1993a): *A practical method for calculating largest Lyapunov exponents from small data sets*. Physica D **65**: pp. 117–134.

Rosenstein M., Collins J. i C.J. De Luca (1993b): *Reconstruction expansion as a geometry-based framework for choosing proper delay times*. Physica D **73**: pp. 82–98.

Roy R., Murphy T., Maier T., Gills Z. i Hunt E. (1992): *Dynamical control of a chaotic laser: Experimental stabilization of a globally coupled system*. Phys. Rev. Lett. **68**(9): pp. 1259–1262.

Ruiz D., Nougues J. i Pulgjaner L. (1999): *Artificial neural networks applied to on-line fault diagnosis in chemical plants*. IEEE International Conference on Emerging Technologies and Factory Automation: pp. 977–986.

Rutkowska D., Piliński M. i Rutkowski L. (1997): *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte*. Wydawnictwo Naukowe PWN. Warszawa.

Rutkowski L. (2005): *Metody i techniki sztucznej inteligencji*. Wydawnictwo Naukowe PWN. Warszawa.

Rutkowski L. (2007): *Flexible Neuro-Fuzzy Systems*. Kluwer Academic Publishers.

Rzydzik S. (2007): *Identyfikacja stanu w układach rozproszonych*. Zeszyty Katedry Podstaw Konstrukcji Maszyn, nr 135. Wydawnictwo Politechniki Śląskiej.

Sang-Hee Kim, Su-Dong Hong i Won-Woo Park (2001): *An adaptive neurocontroller with modified chaotic neural networks*. International Joint Conference on Neural Networks. Tom 1: pp. 509–514.

Sang-Hee Kim i Won-Woo Park (2003): *Convergence analysis of chaotic dynamic neuron*. Proceedings of the International Joint Conference on Neural Networks: pp. 858–863.

Sato Y. i Nagaya S. (1996): *Evolutionary algorithms that generate recurrent neural networks for learning chaos dynamics*. Proceedings of IEEE International Conference on Evolutionary Computation. IEEE. Nagoya, Japan: pp. 144–149.

Satoh S., Shaikh M. i Dote Y. (2001): *Fault diagnosis for dynamical systems using soft computing*. IEEE International Conference on Systems, Man, and Cybernetics: pp. 1448–1452.

Schlang M., Lang B., Poppe T., Runkler T. i Weinzierl K. (2001): *Current and future development in neural computation in steel processing*. Control Engineering Practice **9**(9): pp. 975–986.

Schuster H. (1993): *Chaos deterministyczny. Wprowadzenie*. Wydawnictwo Naukowe PWN. Warszawa.

Siegelmann H., Horne B. i Giles C. (1997): *Computational capabilities of recurrent NARX neural networks*. IEEE Transactions on Systems, Man, and Cybernetics, Part B. Tom 27. IEEE: pp. 208–215.

Sinha K., Gupta M. i Rao H. (2000): *Dynamic neural networks: An overview*. Proceedings of IEEE International Conference on Industrial Technology. Tom 1: pp. 491–496.

Skupnik D. (2008): *Podejście wieloaspektowe do modelowania w diagnostyce technicznej*. Diagnostyka **46**(2): pp. 117–120.

Söderström T. i Stoica P. (1997): *Identyfikacja systemów*. Wydawnictwo Naukowe PWN. Warszawa.

Song K.-l., Luo Z., an Xin-xin Tang F. Y. i xian Yuan S. (2009): *Chaotic oscillator detection system about weak signals in spot welding*. Frontiers of Materials Science in China **3**(1): pp. 93–97.

Song-Ming J., Han P., Li-Hui Z. i Jian-Bo L. (2004): *Recurrent neural network applied in dynamic process identification based on RPROP and chaos optimization coupling algorithm*. Proceedings of the Third Intemational Conference on Machine Learning and Cybemetics. Shanghai: pp. 3339–3343.

Spall J. (2003): *Introduction to stochastic search and optimization. Estimation, simulation and control*. John Wiley & Sons, Inc.

Stachurski A. i Wierzbicki A. (2001): *Podstawy optymalizacji*. Oficyna Wydawnicza Politechniki Warszawskiej.

Stavroulakis P. (2005): *Chaos Applications in Telecommunications*. CRC Taylor and Francis.

Stefánsson A., Končar N. i Jones A. J. (1997): *A note on the Gamma test*. Neural Computing and Applications **5**(3): pp. 131–133.

Szulim R. i Moczulski W. (2004): *A method of mining knowledge to aid control of complex industrial processes*. W: Recent Developments in Artificial Intelligence Methods, W. Cholewa, T. Burczyński i W. Moczulski (Red.). Silesian University of Technology. AI-METH Series. Gliwice: pp. 273–276.

Szulim R. i Moczulski W. (2005): *Metoda pozyskiwania wiedzy do prowadzenia złożonego procesu technologicznego*. Pomiary, Automatyka, Kontrola. Tom 9BIS: pp. 145–148.

Tadeusiewicz R. (1993): *Sieci neuronowe*. Akademicka Oficyna Wydawnicza RM. Warszawa.

The MathWorks (2007): *MATLAB and Simulink for Technical Computing*. Strona domowa oprogramowania (dn. 21.04.2007): `http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/`.

Thiel M., Romano M., Kurths J., Meucci R., Allaria E. i Arecchi F. (2002): *Influence of observational noise on the recurrence quantification analysis*. Physica D: Nonlinear Phenomena **171**(3): pp. 138–152.

Thimm G. i Fiesler E. (1995): *Neural network initialization*. Proceedings of the International Workshop on Artificial Neural Networks: From Natural to Artificial Neural Computation. Springer-Verlag: pp. 535–542.

Tipsuwanporn V., Tarasantisuk C., Numsumran A. i Sawaengsinkasikit W. (2001): *Motor speed identification using multilayer feedforward neural networks*. International Conference on Power Electronics and Drive Systems. Tom 1. IEEE: pp. 62–65.

Tokuda I., Tomokazu Yanai i Kazuyuki Aihara (2001): *Reconstruction of chaotic dynamics via a network of stochastic resonance neurons and its application to speech*. Artificial Life and Robotics **5**(1): pp. 33–39.

Tomanek A., Przystałka P. i Adamczyk M. (2006): *Optimization of structure of neural models using distributed computing environment*. Diagnostyka **4/40**: pp. 15–18.

Tomanek A., Przystałka P. i Wyczółkowski R. (2007): *Optimization of Jordan and Elman neural networks through distributed computing environment*. W: Methods of Artificial Intelligence, W. Cholewa, T. Burczyński i W. Moczulski (Red.). Silesian University of Technology. AI-METH Series. Gliwice: pp. 69–70.

Tsang P. M., Kwok P., Choy S., Kwan R., Ng S., Mak J., Tsang J., Koong K. i Wong T.-L. (2007): *Design and implementation of NN5 for Hong Kong stock price forecasting*. Engineering Applications of Artificial Intelligence **20**(4): pp. 453–461.

Tsoi A. C. i Back A. (1994): *Locally recurrent globally feed-forward networks: a critical review of architectures*. IEEE Transactions on Neural Networks. Tom 5: pp. 229–239.

Tsung-Nan Lin, Giles C. L., Bill G. i Kung S. (1999): *A delay damage model selection algorithm for NARX neural networks*. IEEE Transactions on Signal Processing. Tom 45. IEEE: pp. 2719–2730.

Tykierko M. (2008): *Using invariants to change detection in dynamical system with chaos*. Physica D **237**: pp. 6–13.

Wachla D. (2006): *Identyfikacja dynamicznych modeli diagnostycznych metodami odkryć wiedzy w bazach danych*. Zeszyty Katedry Podstaw Konstrukcji Maszyn, nr 130. Politechnika Śląska. Gliwice.

Wachla D. i Moczulski W. (2007): *Identification of dynamic diagnostic models with the use of methodology of knowledge discovery in databases*. Engineering Applications of Artificial Intelligence **20**(5): pp. 699–707.

Waibel A., Hanazawa T., Hinton G., Shikano K. i Lang K. J. (1989): *Phoneme recognition using time-delay neural networks*. IEEE Transactions on Acoustics, Speech, and Signal Processing. Tom 37: pp. 328–339.

Wang G., Chen D., Lin J. i Chen X. (1999): *The application of chaotic oscillators to weak signal detection*. IEEE transactions on industrial electronics : pp440–444.

Wang L. (2000): *From Plant Data to Process Control: Ideas for Process Identification and PID Design*. Taylor and Francis.

Weimin Shen, Gu J. i Yanjun Shen (2004): *Stability analysis for high-order dynamic neural networks with time delays*. IEEE International Conference on Robotics and Biomimetics. IEEE: pp. 966–971.

Werbos P. (1990): *Backpropagation through time: what it does and how to do it*. Proceedings of the IEEE: pp. 1550–1560.

West B. (1991): *Fractal Physiology and Chaos in Medicine*. World Scientific Publishing Company.

Wieczorek T. (2008): *Neuronowe modelowanie procesów technologicznych*. Wydawnictwo Politechniki Śląskiej. Gliwice.

Wiggins S. (2003): *Introduction to Applied Nonlinear Dynamical Systems and Chaos*. Springer-Verlag New York, Inc.

Williams R. J. i Zipser D. (1989): *A learning algorithm for continually running fully recurrent neural networks*. Neural Computation **1**: pp. 270–280.

Wilson W. (1993): *A comparison of architectural alternatives for recurrent networks*. Proceedings of the 4th Australian Conference on Neural Networks. Sydney University, Faculty of Electrical Engineering: pp. 189–192.

Wilson W. (1995): *Stability of learning in classes of recurrent and feedforward networks*. Proceedings of the 6th Australian Conference on Neural Networks. Sydney University, Faculty of Electrical Engineering: pp. 142–145.

Witczak M. (2003): *Identification and Fault Detection of Non-Linear Dynamic Systems*. University of Zielona Góra Press.

Witczak M. (2007): *Modelling and Estimation Strategies for Fault Diagnosis of Non-Linear Systems*. Springer-Verlag Berlin Heidelberg.

Witczak M., Korbicz J., Mrugalski M. i Patton R. J. (2006): *A GMDH neural network-based approach to robust fault diagnosis: application to the DAMADICS benchmark problem*. Control Engineering Practice **14**(6): pp. 671–683.

Wojtusik J. (2006): *Wielomodele diagnostyczne maszyn wirnikowych*. Zeszyty Katedry Podstaw Konstrukcji Maszyn, nr 128. Politechnika Śląska. Gliwice.

Yazdizadeh A. i Khorasani K. (1997): *Identification of a class of nonlinear systems using dynamic neural network structures*. IEEE International Conference on Neural Networks. Tom 1: pp. 194–198.

Zadeh L. A. (1965): *Fuzzy sets*. Information and Control **8**: pp. 338–353.

Zadeh L. A. (1994): *Fuzzy logic, neural networks and soft computing*. Communications of the ACM **37**(3): pp. 77–84.

Zbilut J. i Webber C. (1992): *Embeddings and delays as derived from quantification of recurrence plots*. Physics Letters A **171**: pp. 199–203.

Żebrowski J., Popławska W., Baranowski R. i Buchner T. (2000): *Symbolic dynamics and complexity in a physiological time series*. Chaos, Solitons and Fractals **11**(7): pp. 1061–1075.

Zieliński R. i Neumann P. (1986): *Stochastyczne metody poszukiwania minimum funkcji*. Wydawnictwo Naukowo-Techniczne, Warszawa.

Żółtowski B. (1996): *Podstawy diagnostyki maszyn*. Wydawnictwo ATR. Bydgoszcz.

Żurada J., Barski M. i Jędruch W. (1996): *Sztuczne sieci neuronowe*. Wydawnictow Naukowe PWN. Warszawa.

Metodyka modelowania neuronowego w diagnostyce procesów z uwzględnieniem elementów teorii chaosu

Praca doktorska - Streszczenie

Autor: mgr inż. Piotr Przystałka
Promotor: prof. dr hab. Wojciech A. Moczulski
Politechnika Śląska w Gliwicach, Wydział Mechaniczny Technologiczny

Niniejsza praca dotyczy detekcji uszkodzeń bazującej na modelu neuronowym procesu. Detekcja uszkodzenia jest bezwarunkowo konieczna dla wielu obiektów rzeczywistych. Głównym celem badań było opracowanie i weryfikacja metodyki modelowania neuronowego w diagnostyce procesów z zastosowaniem inżynierii chaosu. Opracowana metodyka umożliwia tworzenie odpornych systemów detekcji uszkodzeń.

W wyniku przeglądu istniejącego stanu literatury w zakresie metod detekcji uszkodzeń bazujących na modelu neuronowym obiektu stwierdzono, że niezbędny jest rozwój ogólnej metodyki modelowania neuronowego procesów technicznych w celu ograniczenia różnego rodzaju niepewności. Z drugiej strony wiadomo, że nie jest możliwe całkowite wyeliminowanie takich czynników, jak szumy pomiarowe, zakłócenia niemierzalne oraz niepewność modelu. Dlatego też niezbędne jest opracowywanie tzw. pasywnych metod odpornej detekcji uszkodzeń. W związku z powyższym sformułowano tezę, że opracowana metodyka modelowania neuronowego procesów uwzględniająca wybrane elementy teorii chaosu deterministycznego umożliwi tworzenie skutecznie działających systemów detekcji uszkodzeń, odpornych na zakłócenia i błędy modelowania.

W celu wykazania słuszności tej tezy opracowano metodykę modelowania neuronowego procesów, w której zastosowano wybrane elementy teorii chaosu. Do najważniejszych wyników pracy należy opis formalny lokalnie rekurencyjnej sieci neuronowej złożonej z jednostek przetwarzających, dla których możliwe jest uzyskanie zachowania chaotycznego. Dla zaproponowanej sieci opracowano różne schematy uczenia hybrydowego, wzbogacając je o elementy inżynierii chaosu. Problem wyboru relewantnych wejść modelu neuronowego został rozwiązany poprzez rozszerzenie znanej metody wskaźników pojemności informacyjnej. W celu wyboru struktury modelu dla danego problemu stosowano metodę izolinii kryterialnych oraz wybrane metody przycinania struktury sieci. Rozważono również problem analizy stabilności uzyskiwanych modeli neuronowych. Odporna detekcja uszkodzeń wykorzystuje metodę bazującą na analizie ilościowej diagramów rekurencyjnych wyznaczanych dla residuów.

Weryfikację wstępną prowadzono dla danych uzyskanych w wyniku symulacji modeli numerycznych oraz danych zgromadzonych na wybranych obiektach przemysłowych. Zasadnicze badania weryfikacyjne prowadzono na podstawie problemu testowego udostępnionego w ramach europejskiego projektu DAMADICS. Wyniki otrzymane podczas obu stadiów weryfikacji potwierdzają słuszność sformułowanej w pracy tezy.

Słowa kluczowe: diagnostyka procesów technicznych, diagnostyka uszkodzeń, odporna detekcja uszkodzeń, lokalnie rekurencyjne sieci neuronowe, inżynieria chaosu.

Methodology of neural modeling in process diagnostics with the use of elements of the chaos theory

PhD thesis - Summary

Author: Piotr Przystałka, MSc, Eng.
Supervisor: Prof. Wojciech A. Moczulski, PhD, DSc
Silesian University of Technology at Gliwice, Faculty of Mechanical Engineering

The PhD thesis deals with neural model-based fault detection. It is well-known that fault detection is an absolute must for any practical systems. The main aim of the thesis was to elaborate the methodology of neural modeling with the use of chaos engineering. This methodology can be used for creating robust fault detection systems.

Taking into account the current state-of-the-art in the area of the neural model-based fault detection schemes it was stated that it was necessary to develop more advanced neural modeling methods in order to reduce uncertainties caused by various sources. On the other hand, it is well-known that it is impossible to entirely eliminate such origins of uncertainties as measuring noise, non-measurable disturbances and modeling errors. Therefore, there is also the need to elaborate robust fault detection methods by enhancing the robustness of the decision making block. These considerations were the basis for the hypothesis that the proposed methodology of neural modeling of processes with the use of some elements of the chaos theory, allowed creating efficient fault detection systems, to be robust enough to disturbances and modeling errors.

To prove the correctness of the hypothesis, the methodology of neural modeling with the use of some elements of the chaos theory was elaborated. The main part of the proposed approach is a locally recurrent neural network that is composed of complex dynamic neural units for which chaotic behavior can be obtained. For this network several global and local optimization methods were included into hybrid training schemes. Chaos engineering was incorporated into both the evolutionary algorithm and the simulated annealing algorithm in order to improve the efficiency of the tuning procedure. The problem of relevant inputs selection was solved by means of the method of extended Hellwig's coefficient of integral capacity of information. Criteria isolines and some sensitive methods were used to find the suitable architecture of a network. Moreover, the issue of stability analysis of neural models was also considered. Recurrence quantification analysis of residual signals was proposed to robust fault detection, too.

The preliminary verification of the elaborated methodology in modeling tasks was carried out for both simulated and industrial data. The fundamental verification was conducted for the data made available within DAMADICS benchmark problem. The achieved results proved the correctness of the formulated hypothesis.

Key words: diagnostics of technical processes, fault diagnosis, robust fault detection, locally recurrent neural networks, chaos engineering.