Projektowanie regulatorów rozmytych w środowisku MATLAB-Simulink

Bogumiła Mrozek

Celem pracy jest pokazanie możliwości projektowania regulatorów rozmytych, w tym doboru ich parametrów, z wykorzystaniem bibliotek środowiska MATLAB-Simulink.

1. Wprowadzenie

Regulatory rozmyte umożliwiają tworzenie adaptacyjnych układów sterowania dla dużych zmian parametrów obiektu regulowanego, bez potrzeby korzystania ze skomplikowanych lub trudnych do uzyskania modeli tego obiektu.

Regulatory rozmyte o strukturze i parametrach zaprojektowanych z wykorzystaniem bibliotek MATLAB-a można stosować w aplikacjach przemysłowych. Drogę od projektu regulatorów w środowisku MATLAB-Simulink do ich implementacji w docelowym urządzeniu opisano w [6] – rozdział *Szybkie prototypowanie sterowników*.

Fuzzy Logic Toolbox jest biblioteką pakietu MATLAB-Simulink i zawiera zestaw funkcji oraz interfejsy, które ułatwiają projektowanie regulatorów rozmytych. Biblioteka ta ma także bloki Simulinka [2], które można użyć jako model zaprojektowanego regulatora rozmytego.

Fuzzy Logic Toolbox zawiera interfejs **ANFIS Editor**, który wykorzystuje algorytmy uczenia sztucznych sieci neuronowych dla strojenia regulatorów rozmytych.

Ponadto, *Fuzzy Logic Toolbox* zawiera interfejs **Clustering** i funkcje umożliwiające tworzenie modeli rozmytych, przy użyciu technik klasteryzacji. Metody klasteryzacji określają „ważne" punkty układu rozmytego w obszarach maksymalnego zagęszczenia próbek pomiarowych i w tych punktach lokalizują reguły oraz wyznaczają dla nich parametry funkcji przynależności.

Do strojenia regulatorów rozmytych algorytmami uczenia sieci neuronowych oraz metodami klasteryzacji niezbędne są odpowiednie (wiarygodne) zestawy danych, najlepiej pomiarowych.

Jakość działania rozmytego regulatora można opisać wskaźnikiem jakości określonym przez użytkownika.

Dla zdefiniowanego wskaźnika jakości można dobrać strukturę i parametry regulatora rozmytego przy użyciu technik przeszukiwań.

Algorytmy genetyczne i ewolucyjne są to uniwersalne techniki optymalizacji oparte na przeszukiwaniu przestrzeni rozwiązań. Naśladują one genetyczną adaptację występującą w naturalnych procesach ewolucyjnych, zachodzących w przyrodzie [4].

dr inż. Bogumiła Mrozek – Politechnika Krakowska

W odróżnieniu od klasycznych metod optymalizacji, algorytmy te nie wymagają szczegółowej wiedzy o strukturze optymalizowanego problemu. Konieczne jest jedynie określenie funkcji celu. Dla regulatorów rozmytych, jako funkcję celu można przyjąć wskaźnik jakości.

Biblioteka *Genetic Algorithm and Direct Search Toolbox* zawiera algorytm genetyczny dostępny poprzez interfejs *Genetic Algorithm Tool* (`gatool`) oraz poprzez funkcję `ga` [1].

Celem pracy jest pokazanie możliwości projektowania regulatorów rozmytych, w tym doboru ich parametrów, z wykorzystaniem bibliotek środowiska MATLAB-Simulink.

2. Projektowanie regulatorów rozmytych w środowisku MATLAB-Simulink

W *Fuzzy Logic Toolbox* są dostępne dwa podstawowe modele rozmyte: Mamdani i Sugeno [2]. Są one opisane zbiorem reguł. Każda reguła składa się *z przesłanki* (część ***Jeśli***) i konkluzji (część ***to***). Przesłanka zawiera zbiór warunków, a konkluzja wniosek.

Reguły obu modeli mają identyczną postać przesłanek, różnią się konkluzjami. W modelu rozmytym typu Mamdani konkluzja reguł zawiera zbiór rozmyty. Model typu Sugeno w konkluzji ma funkcję. Różnice między oboma rodzajami modeli pokazano na przykładzie rozmytego regulatora PI.

2.1. Regulatory rozmyte PI typu Mamdani

Rozmyte regulatory PI typu Mamdani zwykle używają dwóch zmiennych wejściowych E(*error*) i SE(*sum of errors* – całka błędu) oraz jednej zmiennej wyjściowej U(*output*). Baza reguł regulatora rozmytego PI typu Mamdani przyjmuje postać jak poniżej [8].

$$\textbf{\textit{Jeśli}}\ (E \text{ jest } A_j)\ \textbf{\textit{i}}\ (SE \text{ jest } B_j),\ \textbf{\textit{to}}\ (U \text{ jest } C_j)$$

gdzie: $j = 1,2,...K$, K – liczba reguł modelu, A_i, B_j, C_j – zbiory rozmyte jako wartości lingwistyczne (np. small, large itp.).

Model rozmyty typu Mamdani często jest stosowany w układach opartych na wiedzy eksperta [2]. Jest to model:

- tworzony intuicyjnie
- powszechnie znany i akceptowany
- dobrze dostosowany do sposobu postrzegania zmiennych przez człowieka.

Jeżeli są dostępne dane uczące, to modele rozmyte typu Mamdani można stroić, po ich przekształceniu w sieć neuronową [8] w sposób podobny do opisanego w rozdziale 3.

Można też określać ich bazę reguł i kształt funkcji przynależności modelu rozmytego z wykorzystaniem algorytmów genetycznych, według zasad opisanych w rozdziale 4. Jednakże żadna z tych metod strojenia nie jest zaimplementowana w *Fuzzy Logic Toolbox*.

2.2. Regulatory rozmyte PI typu Sugeno

Typowa reguła Sugeno dla modelu rozmytego z dwoma wejściami ma postać [8]:

$$\textbf{\textit{Jeśli}}\ (x \text{ jest } A_j)\ \textbf{\textit{i}}\ (y \text{ jest } B_j),\ \textbf{\textit{to}}\ \ z_j = f(x,y)$$

gdzie: x, y – zmienne wejściowe, A_j, B_j, – zbiory rozmyte, $f(x,y)$ – funkcja, z – zmienna wyjściowa, określana ze wzoru defuzyfikacji (wyostrzania) [2, 7, 8].

W *Fuzzy Logic Toolbox* funkcja $f(x,y)$ jest wielomianem rzędu pierwszego lub zerowego. Konkluzję j-tej reguły można zapisać w postaci:

$$z_j = f(x,y) = p_j x + q_j y + r_j$$

gdzie: p_j, q_j, r_j – współczynniki wielomianu, w modelu rzędu zerowego: $z_j = r_j$

Dla regulatora rozmytego PI typu Sugeno w postaci ogólnej – j-tą regułę można zapisać:

$$\textbf{\textit{Jeśli}}\ (w1 \text{ jest } A_j)\ \textbf{\textit{i}}\ (w2 \text{ jest } B_j)\ \textbf{\textit{i}}\ (w3 \text{ jest } C_j),\ \textbf{\textit{to}}\ u_j = p_{1j} w1 + p_{2j} w2 + p_{3j} w3 + r_j \qquad (1)$$

gdzie: A_j, B_j, C_j – zbióry rozmyte, $w1$ – zmienna wejściowa określająca nieliniowości regulatora, $w2 = E(error)$, $w3 = SE(sum\ of\ error)$, p_{1j}, p_{2j}, p_{3j}, r_j – współczynniki w konkluzji.

Regulator rozmyty o bazie reguł w postaci (1) jest rozpatrywany w punkcie 5.2, jako model rozmyty generowany przez funkcję `genfis2`.

Możliwy jest także prostszy zapis reguł regulatora rozmytego PI typu Sugeno, z j-tą regułą w postaci jak niżej [2]:

$$\textbf{\textit{Jeśli}}\ (w1 \text{ jest } A_j),\ \textbf{\textit{to}}\ u_j = p_{1j} w2 + p_{2j} w3 + r_j \qquad (2)$$

gdzie: A_j, – zbiór rozmyty, $w1$ – zmienna wejściowa (określa nieliniowości regulatora), $w2 = E(error)$, $w3 = SE(sum\ of\ error)$, p_{1j}, p_{2j}, r_j – współczynniki w konkluzjach.

Regulator rozmyty o bazie reguł w postaci (2) jest rozpatrywany w punkcie 5.3.

Ważne zalety modelu typu Sugeno to [2]:

- duża wydajność i skuteczność obliczeniowa
- dobre działanie w połączeniu z metodami stosowanymi w układach liniowych (np. regulator PID)
- dobra współpraca z metodami adaptacyjnymi i metodami optymalizacji
- struktura ułatwiająca przeprowadzenie dowodów stabilności układów sterowania.

Model Sugeno jest bardziej zwięzły i skuteczniejszy obliczeniowo w porównaniu z modelem Mamdani. Ponadto model rozmyty typu Sugeno umożliwia łatwe stosowanie metod adaptacyjnych do doboru jego parametrów. Kilka z tych metod jest zaimplementowanych w *Fuzzy Logic Toolbox*.

Na podstawie modelu Sugeno można zbudować układ rozmyty, który dokonuje przełączeń pomiędzy kilkoma optymalnymi liniowymi regulatorami jako silnie nieliniowy układ poruszający się wokół ich punktu pracy. Uwzględniając powyższe właściwości, do dalszych rozważań przyjęto model rozmyty typu Sugeno.

2.3. Dobór struktury i parametrów początkowych regulatora rozmytego

Określenie wstępnej struktury i parametrów rozmytego regulatora typu Sugeno można zrealizować na kilka sposobów:

a. Na podstawie wiedzy eksperta można arbitralnie określić liczbę reguł oraz rodzaj, kształt (parametry) i liczbę funkcji przynależności dla każdej zmiennej wejściowej.
   Wartości współczynników wielomianu w konkluzjach reguł można wyznaczyć z wykorzystaniem metod stosowanych do doboru parametrów regulatorów konwencjonalnych, np. przy użyciu *Simulink Response Optimization* (dawniej *NCD Blockset*) [5].
b. Na podstawie dostępnych danych uczących – można użyć funkcji lub interfejsu **ANFIS**. Wygenerowany w ten sposób model typu Sugeno ma określoną liczbę reguł, parametry funkcji przynależności w przesłankach oraz wartości współczynników wielomianu w konkluzjach reguł.
c. Na podstawie dostępnych danych uczących – można zastosować funkcje biblioteczne: `genfis1`, `genfis2` lub `genfis3`. Funkcje te generują struktury reprezentujące układy rozmyte typu Sugeno z ustaloną liczbą reguł oraz z parametrami funkcji w przesłankach i konkluzjach.

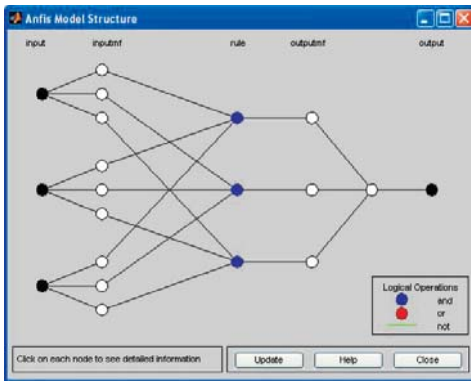
3. Projektowanie regulatorów rozmytych na podstawie danych uczących

Sieci neuronowe mają zdolność uczenia się na bazie danych uzyskanych na podstawie identyfikacji obiektu lub odpowiedzi modelu. W bibliotece *Fuzzy Logic Toolbox* istnieje możliwość strojenia regulatorów rozmytych typu Sugeno metodami uczenia stosowanymi w sieciach neuronowych. W tym celu model rozmyty jest

przekształcany w równoważną wielowarstwową sieć neuronową (perceptron wielowarstwowy).

Takie strojenie realizuje funkcja anfis lub interfejs **ANFIS Editor** (`anfisedit`). Akronim **ANFIS** pochodzi od *Adaptive Neuro-Fuzzy Inference System*.

Algorytm strojenia zastosowany w funkcji `anfis` zaproponował Jang [3]. Przekształcony w sieć neuronową model rozmyty jest reprezentowany przez 6 warstw neuronów następujących po sobie (rys. 1). Warstwa 1 to wartości wejściowe (**input**). Warstwa 2 odpowiada za fuzyfikację (rozmywanie) wartości wejściowych i reprezentuje funkcje przynależności w przesłankach (**inputmf** – najczęściej są to funkcje Gaussa). Warstwa 3 (**rule**) reprezentuje reguły i na wyjściu każdego jej neuronu jest określany stopień aktywacji odpowiedniej reguły. Pozostałe warstwy (**outputmf** i **output**) realizują wzór defuzyfikacji dla modelu Sugeno.



Rys. 1. Okno Anfis Model Structure z wizualizacją regulatora rozmytego opisanego przez trzy reguły w postaci (1), po przekształceniu w sieć neuronową

Uczenie sieci neuronowej polega na stopniowej zmianie jej wag w taki sposób, aby doprowadzić do minimalizacji kryterium nauczania. Jest nim zwykle średni kwadratowy błąd wyjścia sieci względem danych uczących.

Funkcja `anfis` przekształca model rozmyty w sieć neuronową, jak opisano powyżej. Wagi tej sieci odpowiadają parametrom funkcji przynależności w przesłankach i współczynnikom wielomianu w konkluzjach [2].

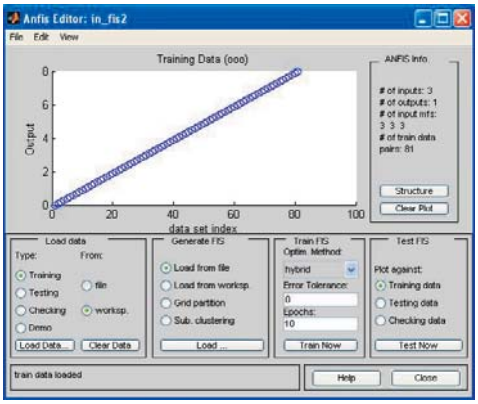
Na podstawie danych uczących i kontrolnych uzyskanych na podstawie identyfikacji lub odpowiedzi modelu, realizuje się proces uczenia sieci neuronowej. Wagi jej wyznacza się za pomocą algorytmu wstecznej propagacji błędu (*backpropagation*) i metody spadku gradientu albo w połączeniu z metodą najmniejszych kwadratów (*linear least squares estimation*) – metoda hybrydowa.

W ten sposób można uzyskać samostrojący się regulator rozmyty na bazie danych pomiarowych typu wejście-wyjście.

Za samostrojący się regulator rozmyty uznaje się regulator o ustalonej liczbie reguł, zbiorów rozmytych w przesłankach i wielomianów w konkluzjach. Zmianie (strojeniu) podlegają jedynie parametry funkcji przynależności w przesłankach i współczynniki wielomianów w konkluzjach.

3.1. Ograniczenia funkcji i interfejsu ANFIS

Okno interfejsu **ANFIS Editor** pokazano na rys. 2. Interfejs i funkcja `anfis` [2] działa poprawnie tylko na modelach rozmytych typu Sugeno rzędu zerowego lub pierwszego. Oznacza to, że w konkluzjach modelu mogą występować tylko stałe lub funkcje liniowe.



Rys. 2. Okno interfejsu ANFIS Editor z wizualizacją danych uczących w funkcji indeksów wektora danych

Inne ograniczenia:

- wszystkie funkcje wyjściowe w konkluzjach muszą być tego samego typu i muszą być albo stałymi albo funkcjami liniowymi
- funkcje w konkluzjach nie mogą się powtarzać, ich liczba musi być równa liczbie reguł
- należy określić funkcje przynależności dla wszystkich zmiennych wejściowych
- układ rozmyty musi mieć wagę jednostkową dla każdej reguły.

3.2. Określenie reguł na podstawie danych pomiarowych

Określenie reguł (bazy reguł), liczby funkcji przynależności oraz ich parametrów zależy od umiejętności wykrywania klasterów (*cluster* – grono, pęk, rój) w próbkach pomiarowych i określania ich środków ciężkości. *Fuzzy Logic Toolbox* zawiera funkcje `fcm` i `subclust`, które realizują algorytmy klasteryzacji odpowiednio metodą *c*-środków (*c-means clustering*) i klasteryzacji różnicowej (*subtractive clustering*).

Fuzzy Logic Toolbox zawiera też dwie funkcje (`genfis2` i `genfis3`), które generują struktury reprezentujące układy rozmyte typu Sugeno. Liczba i parametry funkcji przynależności w przesłankach są określane poprzez algorytmy odpowiednio: klasteryzacji różnicowej i klasteryzacji metodą *c*-środków. Wartości współczynników wielomianu w konkluzjach są wyznaczane metodą najmniejszych kwadratów.

Funkcja `genfis1` generuje modele rozmyte typu Sugeno na podstawie siatki podziału (*grid partition*) – bez klasteryzacji. Wartości współczynników wielomianu w konkluzjach reguł mają wartości zerowe, a liczba reguł jest duża.

Wygenerowane za pomocą funkcji `genfis1`, `genfis2` i `genfis3` modele rozmyte są zazwyczaj stosowane jako modele początkowe (wstępne) dla funkcji lub interfejsu **ANFIS**.

4. Algorytm genetyczny w MATLAB-ie

Algorytm genetyczny to termin ogólny używany do określenia pewnej grupy metod rozwiązywania zagadnień optymalizacji.

Wspólną podstawą koncepcyjną tych metod jest symulowanie ewolucji indywidualnych struktur poprzez proces selekcji i stosowanie operatorów genetycznych. Za pomocą tych operatorów są tworzone nowe warianty (populacje) rozwiązań [4].

W algorytmach genetycznych stosuje się pojęcia zapożyczone z genetyki naturalnej [4].

Istnieje *populacja rozwiązań*. Każde rozwiązanie jest nazywane *osobnikiem*, zwykle reprezentuje go jeden *chromosom*, który zawiera zakodowane potencjalne rozwiązanie problemu. Elementy składowe chromosomu to uporządkowane ciągi *genów*.

W *populacji* dokonuje się selekcji (słabe osobniki giną) i stosuje się operatory genetyczne *krzyżowania* i *mutacji*. Schemat działania algorytmu genetycznego pokazano na rys. 3.

W każdej *generacji* (iteracji algorytmu genetycznego) za pomocą *funkcji przystosowania* (funkcja celu) jest oceniane przystosowanie każdego osobnika. Na tej podstawie jest tworzona nowa populacja osobników. Reprezentują one zbiór potencjalnych rozwiązań problemu. Reprodukcja skupia się na osobnikach o najwyższym stopniu przystosowania.

Krzyżowanie i mutacja mieszają osobniki, realizując w ten sposób eksplorację przestrzeni rozwiązań. Kryteria zatrzymania algorytmu genetycznego (ewolucyjnego) to: wykonanie zadanej liczby generacji albo uzyskanie odpowiedniej wartości przystosowania osobników.

4.1. Genetic Algorithm and Direct Search Toolbox

Genetic Algorithm and Direct Search Toolbox poszerza możliwości funkcji realizujących optymalizację, dostępnych w MATLAB-ie oraz w bibliotece *Optimization Toolbox*. Zawiera dwa algorytmy: genetyczny (funkcja `ga`) i bezpośrednich przeszukiwań (funkcja `patternsearch`) [1].

Algorytmy te można używać do problemów trudnych do rozwiązania tradycyjnymi technikami optymalizacji, tj. trudnych do opisania modelem matematycznym. Stosuje się je też w przypadku funkcji nieciągłych, silnie nieliniowych, stochastycznych lub takich, które mają niepewną lub nieokreśloną pochodną.

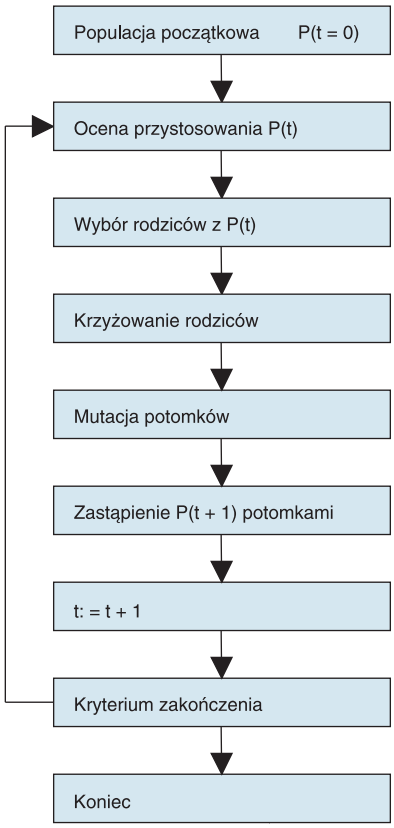
Ważniejsze możliwości *Genetic Algorithm and Direct Search Toolbox* [1]:

- graficzne interfejsy użytkownika *Genetic Algorithm Tool i Pattern Search Tool* umożliwiają szybkie wybranie opcji dla funkcji `ga` i `patternsearch` oraz monitorowanie procesu optymalizacji
- algorytm genetyczny (funkcja `ga`) zawiera opcje tworzenia populacji, selekcji, krzyżowania, mutacji i inne
- algorytm bezpośredniego przeszukiwania (funkcja `patternsearch`) z metodą poszukiwania wzorca
- opcje integrujące *Optimization Toolbox* i funkcje MATLAB-a z algorytmem genetycznym i bezpośrednich przeszukiwań.

Domyślnie przyjęto, że geny w chromosomach osobników są liczbami rzeczywistymi. Można też wybrać opcję kodowania binarnego albo zdefiniować własną strukturę populacji.

Poprzez opcje funkcji `ga` dla poszczególnych etapów (kroków) działania algorytmu jest możliwy wybór implementacji kilku metod selekcji, mutacji, krzyżowania i innych.

Należy pamiętać, że funkcja `ga` poszukuje minimum funkcji przystosowania.



Rys. 3. Schemat działania algorytmu genetycznego

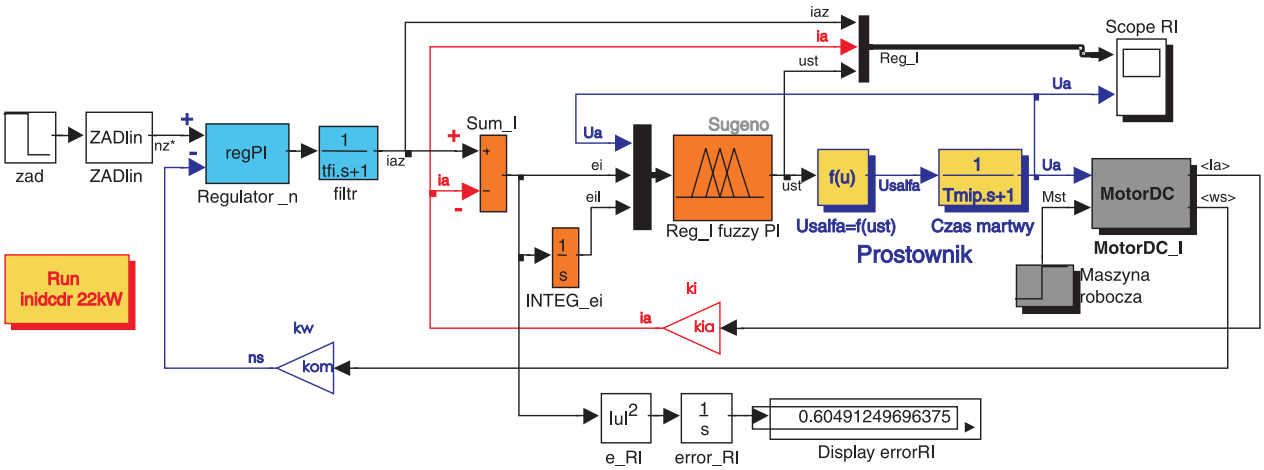
4.2. Genetyczne systemy rozmyte

Celem genetycznych systemów rozmytych jest zautomatyzowanie kroków pozyskiwania wiedzy przy projektowaniu systemów rozmytych.

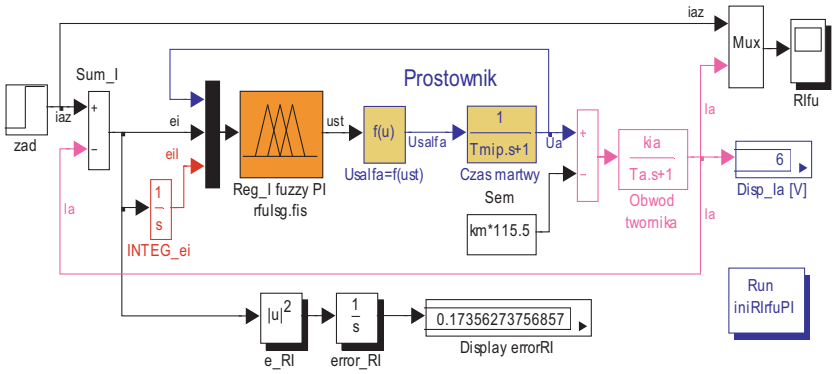
Wiedza zawarta w układach rozmytych składa się z bazy danych i bazy reguł. Przy rozróżnianiu pomiędzy bazą danych i bazą reguł istnieją dwa główne podejścia do genetycznych układów rozmytych. Są one rozróżniane poprzez realizację procesów genetycznego strojenia oraz procesy genetycznego uczenia maszynowego (*genetic based machine learning systems*) [4].

Genetyczny proces uczenia maszynowego dotyczy zagadnień projektowania bazy reguł systemu rozmytego i jest równoważny poszukiwaniu optymalnej konfiguracji zbiorów rozmytych i/lub reguł.

Genetyczny proces strojenia dotyczy optymalizacji działania już zdefiniowanych systemów rozmytych. Sprowadza się on do określenia zbioru parametrów funkcji przynależności w przesłankach i konkluzjach.



Rys. 4. Model Simulinka napędu prądu stałego z rozmytym regulatorem prądu (blok Reg_I fuzzy PI)



Rys. 5. Model Simulinka dla doboru rozmytego regulatora

Algorytm genetyczny dostępny w funkcji `ga` nadaje się do strojenia regulatorów rozmytych typu Mamdani i Sugeno z ustaloną strukturą. Można go łatwo stosować w końcowym etapie strojenia regulatora rozmytego. Funkcję przystosowania należy zdefiniować jako wybraną funkcję błędu określającą jakość sterowania. W tym przypadku wyznaczone minimum funkcji przystosowania jest także minimum funkcji błędu.

5. Badania symulacyjne

Do rozważań przyjęto model napędu prądu stałego, zbudowany z wykorzystaniem pakietu MATLAB-Simulink [5]. Badania symulacyjne wykonano dla napędu z silnikiem prądu stałego o mocy 22 kW, zasilanego z sześciopulsowego przekształtnika (prostownika) tyrystorowego. Inne dane znamionowe silnika: napięcie zasilania U_{aN} = 440 V, prąd silnika I_{aN} = 56,2 A, obroty znamionowe n_N = 1500 obr/min.

Przyjęto strukturę układu sterowania identyczną jak dla napędu z regulatorami klasycznymi [8]. Parametry klasycznego regulatora prędkości (PI) dobrano przy użyciu *NCD Blockset* (obecnie *Simulink Response Optimization*) [5].

Tutaj, w miejsce regulatora prądu PI (o stałych parametrach) zaprojektowano rozmyty regulator prądu PI typu Sugeno (rys. 4). Parametry tego regulatora dobierano metodami uczenia stosowanymi w sieciach neuronowych (funkcja `anfis`) oraz z wykorzystaniem algorytmu genetycznego (funkcja `ga`).

5.1. Model obiektu regulacji

Transmitancja obiektu regulowanego składa się z połączonych szeregowo transmitancji obwodu prądu silnika G_{mot} i przekształtnika tyrystorowego G_{conv} (rys. 5).

Model obwodu prądu silnika opisuje transmitancja G_{mot} prądu silnika i_a względem napięcia zasilania U_a:

$$G_{mot}(s) = \frac{i_a(s)}{U_a(s)} = \frac{k_{ia}}{T_a s + 1} \tag{3}$$

gdzie: k_{ia} – wzmocnienie obwodu prądu, $T_a = L_a/R_a$ – stała czasowa obwodu prądu.

Model przekształtnika tyrystorowego opisuje transmitancja:

$$G_{conv}(s) = \frac{U_{s\alpha}(s)}{u_{st}(s)} = \frac{k_p}{T_{mip} s + 1} \tag{4}$$

Licznik transmitancji odpowiada współczynnikowi wzmocnienia przekształtnika k_p. Stała czasowa T_{mip} odpowiada średniemu czasowi martwemu przekształtnika

(dla sześciopulsowego układu mostkowego przyjmuje się T_{mip} = 1,67 ms).

Charakterystyka przekształtnika tyrystorowego jest nieliniowa. Jego współczynnik wzmocnienia k_p zmienia się i zależy od kąta wysterowania α, tj. napięcia sterującego u_{st}. Napięcie wyjściowe przekształtnika tyrystorowego $U_{s\alpha} = f(u_{st})$ opisuje wzór:

$$U_{s\alpha} = U_{s0}\cos\alpha = U_{s0}\cos\left(\frac{\pi}{2}\left(1-\frac{u_{st}}{u_{st\max}}\right)\right) \qquad (5)$$

gdzie: U_{s0}, α – napięcie przekształtnika bez obciążenia, kąt wysterowania przekształtnika, u_{st}, $u_{st\,max}$ – napięcie sterujące dla $\alpha = 0$.

5.2. Regulator neuronowo-rozmyty

Zaprojektowano rozmyty regulator prądu (PI) typu Sugeno dla obwodu prądu silnika. Regulator ten uwzględnia nieliniowość współczynnika wzmocnienia prostownika tyrystorowego (k_p).

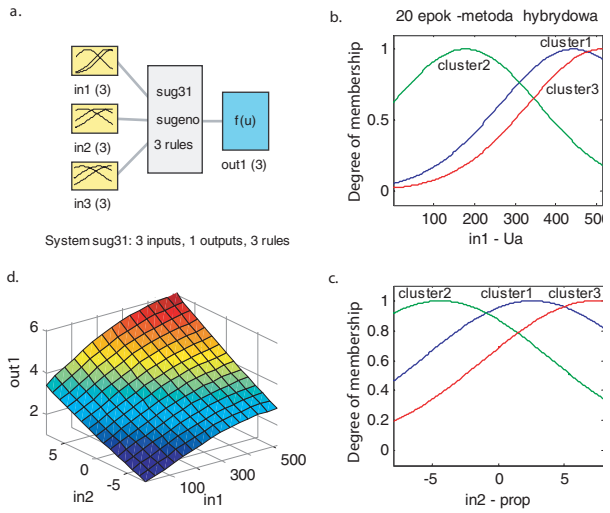
Przyjęto jako dane uczące i kontrolne (*Training, Checking data*) wartości uzyskane ze wzoru (5).

Uwzględniając ograniczenia funkcji `anfis` opisane wyżej, wstępne modele rozmytego regulatora prądu (PI) typu Sugeno określono przy użyciu funkcji: `genfis2` i `genfis3`.

Funkcja `genfis1` generuje modele rozmyte o dużej liczbie reguł, stąd uznano, iż jej sposób działania nie daje optymalnych rozwiązań rozpatrywanego zagadnienia.

Obie funkcje, tj. `genfis2` i `genfis3` tworzą modele rozmyte typu Sugeno, których wszystkie funkcje przynależności w przesłankach są symetrycznymi funkcjami Gaussa (`gaussmf`), a funkcje w konkluzjach są liniowe.

Rozmyty regulator PI prądu stojana typu Sugeno używa, jako sygnałów wejściowych, trzech zmiennych: in1 = U_a (napięcie zasilania), in2 = *prop* (błąd prądu) oraz in3 = *int* (całka błędu prądu). Na wyjściu ma jedną zmienną u_{st} (napięcie sterujące przekształtnika).



Rys. 6. Model rozmyty utworzony funkcją `genfis2` a – diagram regulatora rozmytego dla bazy reguł (6), b – funkcje przynależności dla zmiennej in1 = U_a ,c – funkcje przynależności dla zmiennej in2 = *prop*, d – powierzchnia sterowania uzyskana metodą hybrydową

Przyjęto, że w danych uczących wystarczy wydzielić trzy klastery. Uzyskano w ten sposób regulator rozmyty, opisany przez 3 reguły i 30 parametrów (18 parametrów dla przesłanek i 12 parametrów dla konkluzji – **Anfis Model Structure** pokazuje rys. 1).

Każda ze zmiennych lingwistycznych in1, in2, in3 jest określona trzema funkcjami przynależności. Na rys. 6b pokazano funkcje reprezentujące zbiory rozmyte dla in1 = U_a – small (cluster1), medium (cluster2), large (cluster3).

Dla zmiennej in2 = *prop* (rys. 6c) nazwy zbiorów rozmytych to: in2cl1 (cluster1), in2cl2 (cluster2), in2cl3 (cluster3). Odpowiednio dla in3 = *int* – in3cl1(cluster1), in3cl2 (cluster2), in3cl3 (cluster3).

Baza reguł utworzona przez funkcję `genfis2` zawiera trzy następujące reguły:

Jeśli (U_a jest small) ***i*** (*prop* jest in2cl1) ***i*** (*int* jest in3cl1),
to $u_{st1} = a_1\,U_a + k_1\,prop + t_1\,int + c_1$

Jeśli (U_a jest medium) ***i*** (*prop* jest in2cl2) ***i*** (*int* jest in3cl2), ***to*** $u_{st2} = a_2\,U_a + k_2\,prop + t_2\,int + c_2$

Jeśli (U_a jest large) ***i*** (*prop* jest in2cl3) ***i*** (*int* jest in3cl3), ***to*** $u_{st3} = a_3\,U_a + k_3\,prop + t_3\,int + c_3$ (6)

gdzie: $a_1, k_1, t_1, c_1, a_2, k_2, c_2, t_2, a_3, k_3, t_3, c_3$ – współczynniki wielomianu w konkluzjach.

Modele rozmyte wygenerowane przez `genfis2` i `genfis3` różnią się wartościami parametrów funkcji przynależności w przesłankach i wielomianów w konkluzjach. Wynika to głównie z różnych metod klasteryzacji zaimplementowanych w obu tych funkcjach.

Na rys. 6 pokazano diagram wejść i wyjść zaprojektowanego regulatora rozmytego oraz powierzchnię sterowania i funkcje przynależności dwóch zmiennych wejściowych, które uzyskano po 20 epokach uczenia metodą hybrydową.

5.3. Parametry regulatora rozmytego dobierane algorytmem genetycznym

Możliwości strojenia regulatorów rozmytych z wykorzystaniem funkcji `ga` sprawdzono dla regulatora typu Sugeno projektowanego dla układu opisanego w punkcie 5.1.

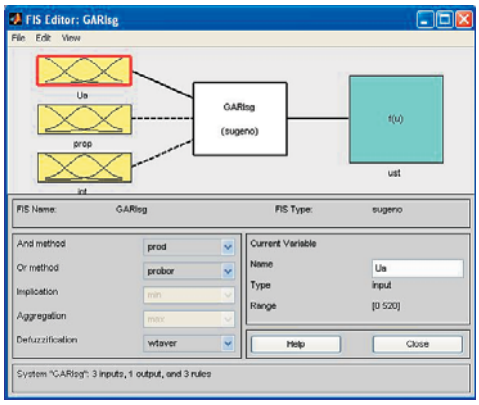
Do rozważań przyjęto strukturę regulatora rozmytego, który reprezentuje implementację trzech różnych regulatorów PI. Parametry tych regulatorów odpowiadają wybranym punktom pracy rozpatrywanego obwodu regulacji prądu silnika [5].

Sposób działania regulatora rozmytego opisują zmienne lingwistyczne: U_a (napięcie zasilania silnika), *prop* (błąd prądu), *int* (całka błędu prądu) oraz trzy następujące reguły:

Jeśli U_a jest small, ***to*** $u_{st1} = k_1\,prop + t_1\,int + c_1$ (7)
Jeśli U_a jest medium, ***to*** $u_{st2} = k_2\,prop + t_2\,int + c_2$
Jeśli U_a jest large, ***to*** $u_{st3} = k_3\,prop + t_3\,int + c_3$

gdzie: $k_1, t_1, c_1, k_2, c_2, t_2, k_3, t_3, c_3$ – współczynniki wielomianu w konkluzjach.

Zmienne wejściowe *prop* (błąd prądu) i *int* (całka błędu prądu) nie pojawiają się w przesłankach. Zmienna wyjściowa u_{st} jest to napięcie sterujące przekształtnika. Diagram wejść-wyjść tego regulatora pokazano na rys. 7.



Rys. 7. Diagram regulatora rozmytego dla bazy reguł (7)

Zmienna lingwistyczna U_a jest określona trzema funkcjami przynależności (rys. 6b). Funkcje te reprezentują zbiory rozmyte small (cluster1), medium (cluster2), large (cluster3).

Zwraca się uwagę, iż liczbę reguł i wielomian w konkluzji tego regulatora określono według wiedzy eksperta (arbitralnie). Parametrów regulatora rozmytego o strukturze opisanej wyżej nie można dobierać przy użyciu funkcji `anfis` (brak funkcji przynależności dla zmiennych wejściowych *prop* i *int*).

Algorytm genetyczny poszukiwał wartości 9 współczynników k_1, t_1, c_1, k_2, c_2, t_2, k_3, t_3, c_3 w konkluzjach regulatora typu Sugeno, aby osiągnąć minimum funkcji przystosowania.

Funkcję przystosowania określono jako sumę błędu kwadratowego odpowiedzi badanego układu (model Simulinka z rys. 5) w ciągu 2 sekund symulacji. Projektowany rozmyty regulator typu Sugeno jest reprezentowany przez blok Simulinka. Na jego wejście wprowadzono skokową zmianę sygnału zadawania.

Osobnika określa jeden chromosom, który składa się z genów reprezentujących poszukiwane współczynniki. Geny w chromosomie są liczbami rzeczywistymi (reprezentacja domyślna w funkcji `ga`). Wartości wstępne parametrów regulatora rozmytego (genów) zapisano w postaci wektora o podwójnej precyzji, z następującymi wartościami jego elementów [5]:

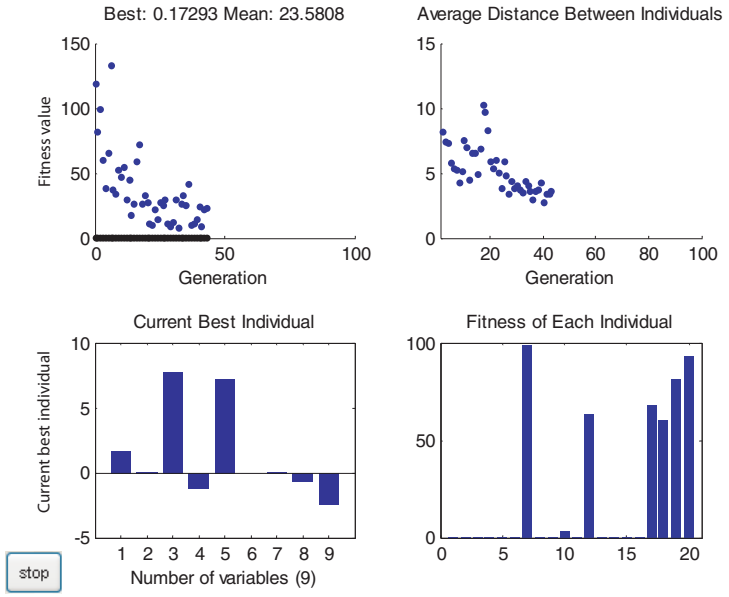
$$xp = [1{,}21 \;\; 0{,}031 \;\; 0 \;\; 1{,}58 \;\; 0{,}03 \;\; 0 \;\; 3{,}17 \;\; 0{,}049 \;\; 0]$$

(wartość błędu wynosiła 1,5253).

Funkcja `ga` umożliwia wizualizację parametrów istotnych dla przebiegu i zbieżności procesu genetycznej optymalizacji. W każdej generacji tworzy się 20 osobników (wartość domyślna).

Na rys. 8 pokazano zmiany funkcji przystosowania (*Fitness Value*) i średnią odległość między osobnikami (*Average Distance Between Individuals*) w kolejnych generacjach.

Pokazano też wartości aktualnie najlepszego osobnika (*Current Best Individual*) i wartości funkcji przystosowania (*Fitness of Each Individual*) wszystkich 20 osobników w aktualnej generacji.

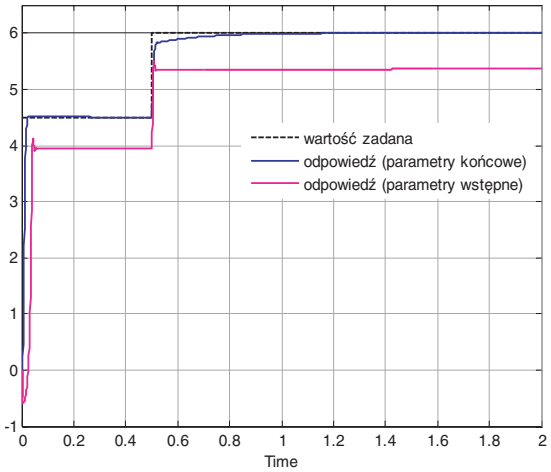


Rys. 8. Wizualizacja procesu genetycznej optymalizacji

Otrzymane parametry regulatora rozmytego, będące rezultatem optymalizacji przy użyciu algorytmu genetycznego (funkcja `ga`), zawiera wektor:

$$xk = [1.61 \;\; 0.07 \;\; 3.61 \;\; 0.79 \;\; 3.43 \;\; 2.87 \;\; 0.27 \;\; 1.78 \;\; 5.41]$$

Po 38 generacjach błąd miał wartość 0,1720. Odpowiedzi regulatora rozmytego na zadany skok jednostkowy dla parametrów wstępnych (wektor *xp*) i parametrów końcowych (wektor *xk*) pokazano na rys. 9.

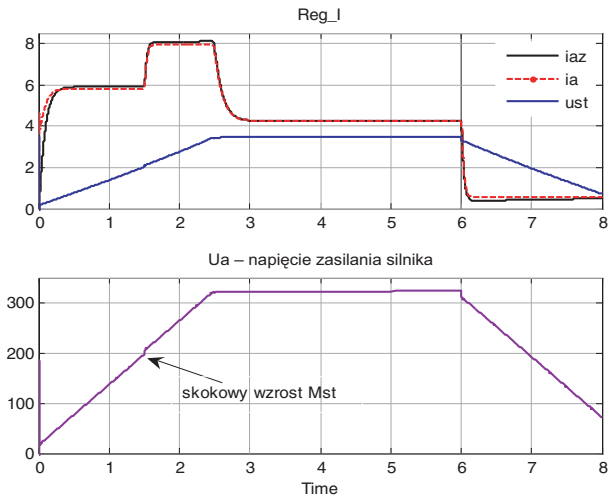


Rys. 9. Odpowiedzi regulatora na skok jednostkowy dla parametrów z wektora *xp* i *xk* (dla modelu z rys. 5)

6. Uwagi końcowe

Struktury rozmytego regulatora prądu dobrano z wykorzystaniem funkcji `anfis` i `ga`. Następnie testowano je w modelach Simulinka pokazanych na rys. 4 i 5.

W obu modelach uzyskano lepszą jakość sterowania (mniejszy błąd) dla regulatora rozmytego, jeśli parametry dobierano za pomocą funkcji `ga`.



Rys. 10. Przebiegi wejścia-wyjścia rozmytego regulatora prądu i napięcia U_a dla parametrów z wektora *xk* (dla modelu z rys. 4 – blok Scope RI)

Etap 1: **Zmienne lingwistyczne**
- Liczba i zakres zmian sygnałów na wejściu i wyjściu
- Wybór funkcji przynależności opisującej zbiory rozmyte (`mfdemo, trimf, gaussmf, ...`)
- Wybór wielomianu w konkluzjach (`constant, linear`)

↓

Etap 2: **Baza reguł regulatora rozmytego**
- Wiedza eksperta
- Funkcje `genfis1, genfis2, genfis3 (fcm, subclust)`

↓

Etap 3: **Algorytmy implikacji, wnioskowania itp.**
- Opcje interejsu Fis Editor
- Pola obiektu `(ImpMethod, AggMethod, ...)`

↓

Etap 4: **Dobór parametrów regulatora rozmytego**
- Na podstawie danych numerycznych `(anfis)`
- Na podstawie kryterium jakości sterowania `(ga)`

Rys. 11. Etapy procesu projektowania regulatora rozmytego typu Sugeno

Na rys. 10 pokazano rezultaty testów symulacyjnych napędu prądu stałego (dla modelu z rys. 4) w fazie rozruchu, wzrostu obciążenia, pracy ustalonej i hamowania.

Algorytm genetyczny zaimplementowany w funkcji `ga` umożliwia strojenie rozmytych regulatorów typu Sugeno i Mamdani oparte na zdefiniowanych kryteriach błędu, bez potrzeby stosowania danych uczących.

Funkcja `ga` może być używana do doboru parametrów regulatorów rozmytych o dowolnych strukturach. Nie podlegają one takim ograniczeniom, jak w przypadku stosowania funkcji `anfis`.

Funkcja `anfis` może być przydatna do strojenia [8] np. układów sterowania pośrednich z modelem wzorcowym lub bezpośrednich z modelem odwrotnym. Jakość strojenia regulatorów rozmytych z wykorzystaniem funkcji `anfis` zależy w sposób istotny od użytych danych uczących.

W połączeniu z dostępnymi w *Fuzzy Logic Toolbox* metodami klasteryzacji i neuronowo-rozmytymi technikami uczenia, *Genetic Algorithm Tool* jest narzędziem, które ułatwia znacząco projektowanie regulatorów rozmytych typu Sugeno, w środowisku MATLAB-Simulink.

Etapy procesu projektowania regulatora rozmytego typu Sugeno z uwzględnieniem funkcji bibliotek MATLAB-a wspomagających to projektowanie pokazano na rys. 11.

Bibliografia

1. *Genetic Algorithm and Direct Search Toolbox User's Guide*, v. 2.0.1 (R2006a), The MathWorks, Inc.
2. *Fuzzy Logic Toolbox User's Guide*, v. 2.2.3 (R. 2006a), The MathWorks, Inc.
3. J.-S.R. Jang, ANFIS: *Adaptive-Network-based Fuzzy Inference Systems*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 3, (1993).
4. Z. Michalewicz, *Algorytmy genetyczne + struktury danych = programy ewolucyjne*. WNT, Warszawa 2003
5. B. Mrozek, *Regulatory rozmyte dla napędu prądu stałego*, Pomiary Automatyka Robotyka 2/2003.
6. B. Mrozek, Z. Mrozek, *MATLAB i Simulink. Poradnik użytkownika*, Helion, Gliwice, 2004.
7. T.J. Ross, *Fuzzy logic with engineering applications*, John Wiley&Sons, 2004.
8. P. Vas, *Artificial Intelligence Based Electrical Machines and Drives*, Oxford University Press, 1999. ■