Wydział Elektryczny
Zespół Automatyki (ZTMAiPC)

LABORATORIUM KOMPUTEROWYCH UKŁADÓW STEROWANIA

Ćwiczenie 2
Projektowanie układu regulacji rozmytej

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z procedurą projektowania prostego regulatora metodami logiki rozmytej (*fuzzy logic*) i zastosowanie go do układu regulacji z nieliniowym obiektem.

2. Wprowadzenie

Sterowanie rozmyte oferuje nowe możliwości projektowania sterowania obiektami nieliniowymi, szczególnie w przypadku, gdy charakter nieliniowości utrudnia ich opisanie metodami analitycznymi, np. w formie równań różniczkowych lub algebraicznych, i wymagana jest zmiana parametrów regulacji w zależności od punktu pracy. Tradycyjną techniką stosowaną w takich przypadkach jest tzw. programowanie wzmocnienia (*gain scheduling*), ale analiza działania otrzymanego regulatora jest zwykle trudna. Ze względu na możliwość implementacji algorytmu sterowanie rozmyte należy do komputerowych metod regulacji prowadzonej w czasie dyskretnym. Można wyróżnić następujące cechy sterowania rozmytego:

- umożliwia zapisanie problemu w *języku naturalnym* na podstawie doświadczenia "eksperta" (analizy zależności zbioru danych z wejścia i wyjścia procesu), co ułatwia jego zrozumienie,
- umożliwia modelowanie zależności nieliniowych o dużej złożoności, gdzie opis analityczny jest trudny lub niemożliwy,
- umożliwia zastosowanie adaptacyjnej techniki doboru parametrów na podstawie danych uczących (ANFIS - *Adaptive Neuro-Fuzzy Inference Systems*),
- jest elastyczne i odporne na nieprecyzyjne dane,
- nadaje się do stosowania obliczeń równoległych,
- może być łączone z konwencjonalnymi metodami sterowania.

Logika rozmyta opiera się na pojęciu *zbioru rozmytego*. Zbiór rozmyty różni się od klasycznego zbioru logiki dwuwartościowej tym, że nie ma ostrej, dobrze określonej granicy. W przypadku klasycznego zbioru *A* element *x* całkowicie należy do *A* (przynależność równa 1) albo całkowicie jest z *A* wyłączny (przynależność równa 0), czyli należy do zbioru *nie-A* (jest to tzw. zasada wyłączonego środka). W przypadku zbioru rozmytego przynależność elementu może być częściowa i przybierać dowolną wartość z przedziału [0,1]. Wartość ta jest określona przez tzw. funkcję przynależności (*membership function*). W przypadku pojęć nieostrych i nieprecyzyjnych logika rozmyta jest naturalnym sposobem opisu. Ilustruje to rys.1, na którym pokazane są przykłady pojedynczej dyskretnej funkcji przynależności (1a) oraz zbioru ciągłych funkcji pokrywających całą przestrzeń wartości wejściowych (1b). O konkretnym kształcie i położeniu funkcji przynależności decyduje "wiedza eksperta", którym może być doświadczony operator albo np. sieć neuronowa uczona danymi doświadczalnymi z procesu.

Poziomy przynależności do zbiorów rozmytych różne od 0 (*false*) lub 1 (*true*) wymagają rozszerzenia definicji *operacji logicznych*. I tak najprostszym rozszerzeniem operacji iloczynu logicznego

A AND *B*,

z poziomami przynależności elementu o wartości x do zbiorów rozmytych A i B: $0 \le \mu_A(x), \mu_B(x) \le 1$, jest zastosowanie funkcji $min[\mu_A(x), \mu_B(x)]$ wybierającej mniejszą z wartości funkcji przynależności do A i B, lub iloczynu $Prod[\mu_A(x), \mu_B(x)] = \mu_A(x) \cdot \mu_B(x)$

Dla operacji sumy logicznej

A OR B

stosuje się najczęściej funkcję $max[\mu_A(x), \mu_B(x)]$ (większa z wartości funkcji przynależności do A i B), lub sumę probabilistyczną $probOR[\mu_A(x), \mu_B(x)] = \mu_A(x) + \mu_B(x) - \mu_A(x) \cdot \mu_B(x)$.

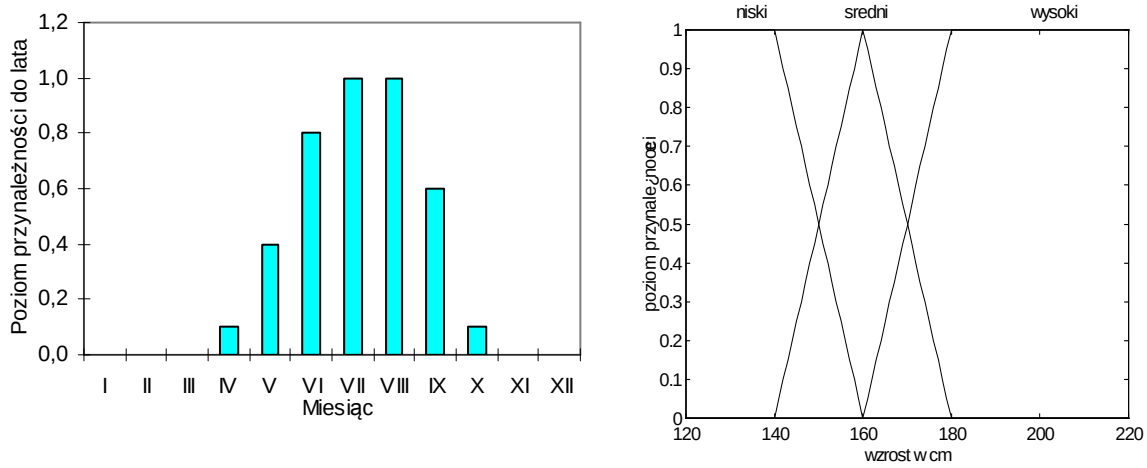
Dla logicznej negacji

NOT A

stosuje się zależność $1 - \mu_A(x)$.

Korzystając z podanych zależności można stworzyć tablice prawdy logiki rozmytej. W ogólności, funkcje dla operatorów logiki rozmytej można wybierać w sposób bardzo dowolny przy zachowaniu ogólnych zasad, w szegółności zgodności z logiką klasyczną dla wartości przynależności równych 0 i 1.

a) b)



Rys1. Przykłady funcji przynależności: a) miesiące należące do lata, b) podział wzrostu człowieka na 3 kategorie: niski, średni i wysoki (trapezowe i trójkątne kawałkami liniowe funkcje przynależności)

Zbiory i operatory rozmyte pełnią funkcje odpowiednio podmiotu i orzeczenia zdań logiki rozmytej. Do konstruowania algorytmów rozmytych wykorzystuje się zdania warunkowe typu *jeżeli-to* (*if-then*). W najprostszym przypadku ma ono formę

*if (**x** is **A**) then (**y** is **B**),*

gdzie A i B są wartościami lingwstycznymi określonymi przez zbiory rozmyte na przestrzeniach X i Y, z których pochodzą elementy x i y. Zdanie po *if* nazywa się przesłanką (poprzednikiem), a zdanie po *then* – konkluzją (następnikiem).

Przykład: *if **prędkość** is **ujemna mała** then **siła** is **dodatnia duża***

Przesłanka zwraca *liczbę* określającą poziom przynależności $\mu_A(x)$ konkretnej wartości wejściowej x do zbioru A, natomiast w konkluzji wartości wyjściowej y przyporządkowuje się *zbiór* rozmyty B (a właściwie funkcję przynależności $\mu_B(y)$) (co można wyrazić w języku C czy MATLAB przez różnicę symboli: *if x == A then y = B).* Klasyczna metoda z rozmytym zbiorem wyjściowym nosi nazwę metody Mamdani'ego. W wielu przypadkach bardziej efektywne jest zastosowanie jako wyjściowej funkcji przynależności

pojedynczego piku (tzw. singletona), co ułatwia opisaną w pkt. 3.5 defuzzyfikację wyjścia (patrz pkt. 4: Układy rozmyte Sugeno).

Przesłanka może składać się z wielu części połączonych operatorami, np.

if (***położenie*** *is not* ***dodatnie***) *and* (***prędkość*** *is* ***ujemna mała***) *then* ***siła*** *is* ***dodatnia mała***

Wszystkie składowe przesłanki mogą być obliczane jednocześnie, a wynik liczbowy otrzymuje się po zastosowaniu operatora logicznego AND. Złożona może być konkluzja, np. (w przypadku dwóch zmiennych wyjściowych):

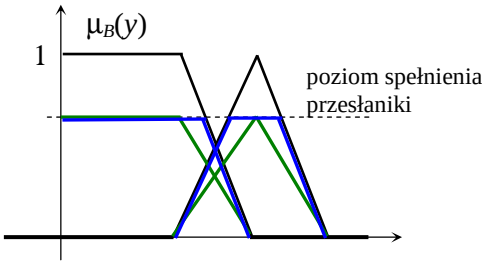
if ***położenie*** *is* ***dodatnie duże*** *then* (***siła*** *is* ***ujemna duża***) *and* (***zawór*** *is* ***otwarty***)

W tym przypadku wynik przesłanki ma jednakowy wpływ na wszystkie składowe konkluzji.

W logice dwuwartościowej implikacja $p \rightarrow q$ ma wartość 0 lub 1 zależnie od wartości przesłanki. W logice rozmytej, jeśli przesłanka spełniona jest częściowo, to konkluzja będąca rezultatem implikacji również, np. $0.5p \rightarrow 0.5q$. Wartość przesłanki *modyfikuje* funkcję przynależności do zbioru rozmytego *B* związanego z wielkością wyjściową układu rozmytego *y* przez zastosowanie przyjętej *metody (funkcji) implikacji*. Najczęściej stosowane metody to (rys. 2):

- obcięcie $\mu_B(y)$ na poziomie prawdy spełnienia przesłanki lub (metoda *min*)

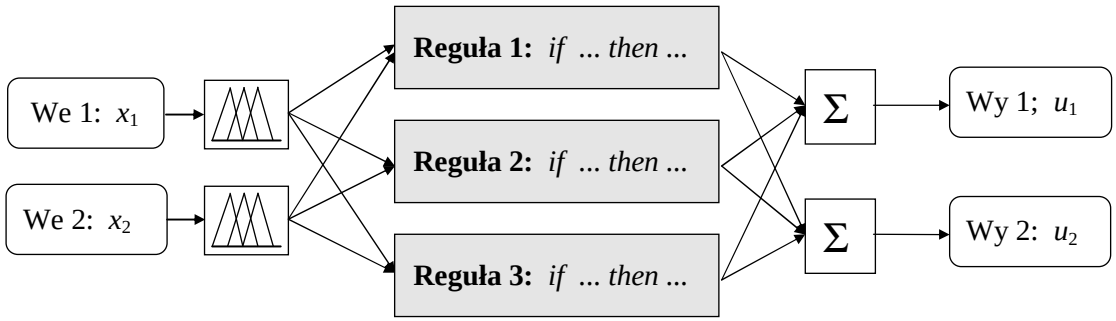
- przeskalowanie $\mu_B(y)$ przez czynnik równy poziomowi prawdy spełnienia przesłanki (metoda *prod*).



Rys.2. Modyfikacja funkcji przynależności do zbioru rozmytego wielkości z konkluzji dla dwóch metod implikacji: obcięcia i skalowania.

3. Etapy projektowania układu rozmytego

Typowy schemat działania klasycznego układu rozmytego Mamdani'ego pokazuje rys.3.



Wejścia - konkretne liczby, podlegają fuzzyfikacji

Reguły obliczane w sposób równolegly z zastosowaniem zasad wnioskowania rozmytego

Wyniki implikacji są łączone (agregacja) i poddawane defuzzyfikacji

Wyjścia - konkretne liczby

Rys. 3. Schemat działania układu rozmytego

Projektowanie układu sprowadza się do zdefiniowania operacji wykonywanych w poszczególnych krokach. Poprawne działanie układu, np. regulatora rozmytego, zależy przede wszystkim od właściwego

określenia liczby i parametrów funkcji przynależności wielkości wejściowych i wyjściowych do zbiorów rozmytych oraz od zdefiniowania operacji wnioskowania rozmytego w poszczególnych regułach, których liczba waha się od kilku do kilkudziesięciu.

1. Fuzzyfikacja wejść. Polega ona na określeniu stopnia przynależności danej wartości wielkości wejściowej do każdego z odpowiadających jej zbiorów rozmytych pokrywających zakres możliwych wartości wejściowych (np. do jakiego stopnia prędkość jest np. duża, a do jakiego mała). Operacja ta sprowadza się na obliczaniu funkcji lub wyszukiwaniu odpowiednich wartości w tabelach.

2. Zastosowanie operatorów logiki rozmytej do określenia stopnia, w jakim spełniona jest przesłanka w każdej z reguł. Wartościami wejściowymi są wartości przynależności sfuzzyfikowanych wejść, na których wykonywane są rozmyte operacje logiczne (AND, OR itp.) tworzące przesłankę. Jako wynik otrzymuje się pojedynczy poziom prawdy spełnienia przesłanki.

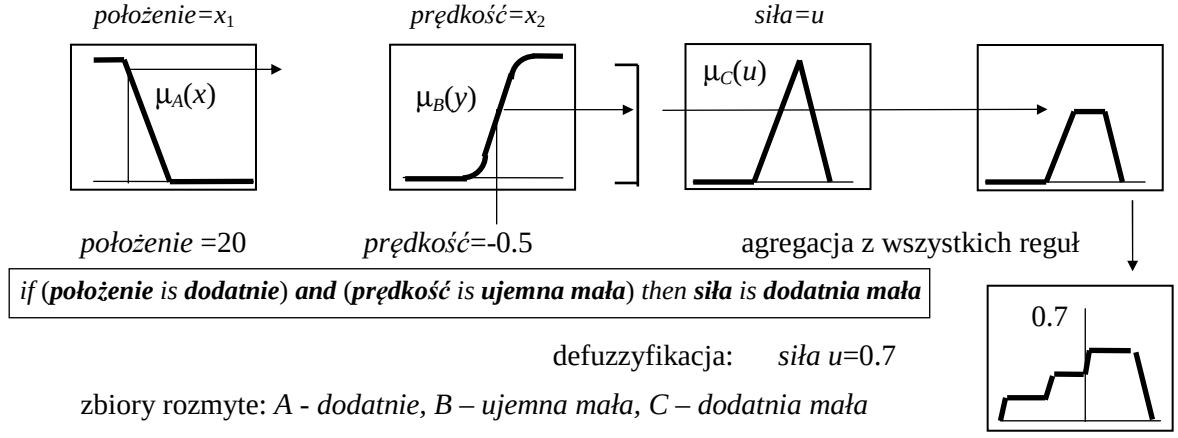
3. Zastosowanie metody implikacji. Operacja ta sprowadza się do zmiany kształtu funkcji przynależności zbioru rozmytego konkluzji zgodnie z poziomem prawdy spełnienia przesłanki (przez obcięcie lub skalowanie). Dodatkowo przesłance każdej z reguł można nadać wagę z zakresu od 0 do 1 wyrażającą jej ważność w porównaniu z innymi. Wynikiem operacji są zbiory rozmyte odpowiadające każdej wielkości wyjściowej występującej w konkluzji.

4. Agregacja wszystkich wyjść. Polega ona na połączeniu odpowiadających jej zbiorów wyjściowych ze wszystkich reguł w jeden zbiór rozmyty dla każdej wielkości wyjściowej. Na wejścu procesu agregacji mamy listę obciętych lub przeskalowanych w wyniku implikacji funkcji przynależności danej wielkości wyjściowej w poszczególnych regułach (niekoniecznie wszystkich).

5. Defuzzyfikacja (wyostrzanie). Polega na wyznaczeniu konkretej wartości dla każdej wielkości wyjściowej ze zbioru rozmytego otrzymanego po agregacji zbiorów z poszczególnych reguł. Najczęściej stosowaną metodą defuzzyfikacji jest obliczanie *środka ciężkości* obszaru pod krzywą zagregowanej funkcji przynależności (*centroid method*). Prowadzi to do wyznaczenia wartości będącej „średnią ważoną" z reguł: wagi reguł są proporcjonalne do poziomu spełnienia ich przesłanek. Inne, prostsze możliwości to: średnia maksimów funkcji zbioru wyjściowego z poszczególnych reguł, wybór największego lub najmniejszego z maksimów czy metoda bisekcji.

Przykładowy przebieg opisanych operacji ilustruje rys.4. Warto zwrócić uwagę na to, że zaprojektowany w opisany sposób regulator rozmyty realizuje w każdym kroku czasowym $t_n=nT_s$ *statyczną* funkcję przejścia $u(n)=f[x_1(n),x_2(n)]$. Działanie dynamiczne można otrzymać przez wykonanie różniczkowania lub całkowania przed układem rozmytym i podanie otrzymanych w ten sposób sygnałów na jego wejścia (np. prędkość wyznaczamy jako pochodną położenia $x_2=dx_1/dt$ i wprowadzamy jako drugą wielkość wejściową).

Fuzzyfikacja 2 wejść i zastosowanie operatora AND - $min(\mu_A,\mu_B)$. Metoda implikacji - obcięcie



Rys.4. Kroki obliczeń w układzie rozmytym (pokazana jest tylko jedna reguła)

Rozmyty regulator PID ma 3 wielkości wejściowe: $x_1 = e, \quad x_2 = \int_0^t e dt = \rho, \quad x_3 = \frac{de}{dt} = \dot{e}$, gdzie $e=e(n)=y_{zad}(n) - y(n)$ jest błędem regulacji (różnicą pomiędzy wartością zadaną y_{zad} a rzeczywistą y wielkości regulowanej w kolejnych krokach czasowych) i wytwarza sygnał sterujący $u(n)$ opisany przez 3-wymiarową *powierzchnię sterowania* $u = f(e, \rho, \dot{e})$. Całkowanie i różniczkowanie błędu regulacji przeprowadza się w każdym kroku czasowym przed wprowadzeniem do części rozmytej regulatora.

4. Układy rozmyte Sugeno

Modyfikacje funkcji przynależności w konkluzjach reguł oraz defuzzyfikacja w klasycznych układach rozmytych Mamdani'ego są operacjami wymagającymi skomplikowanych obliczeń. Równie efektywne a prostsze jest zastosowanie w konkluzjach reguł nie zbiorów rozmytych a zwykłych funkcji zmiennych wejściowych. Konstruowanie przesłanek reguł przebiega tak jak dla układów Mamdani'ego. Takie układy rozmyte nazywają sie układami Sugeno.

i-ta reguła układu rozmytego Sugeno składającego się z M reguł (i=1,2,...,M) z k wejściami zapisanymi w formie wektora $\mathbf{x} = [x_1, x_2, \ldots, x_k]^T$ i jednym wyjściem u ma postać:

$$R^{(i)}: \quad if\,(x_1\ is\ A_1^{(i)})\ and\ (x_2\ is\ A_2^{(i)})\ and \ldots (x_k\ is\ A_k^{(i)})\ then\ \ u^{(i)} = f_i(x_1, x_2, \ldots, x_k)$$

gdzie jako funkcje f_i w konkluzjach stosuje się *funkcje liniowe* o stałych współczynnikach

$$u^{(i)} = c_1^{(i)} x_1 + c_2^{(i)} x_2 + \ldots + c_k^{(i)} x_k + c_0^{(i)}$$

co daje kawałkami liniową charakterystykę wejście-wyjście $u = f(\mathbf{x})$. Szczególnie prostym przypadkiem jest zastosowanie *funkcji stałych*

$$u^{(i)} = c_0^{(i)} = \text{const}$$

tzw. *singletonów* (odpowiadają one funkcjom przynależności w kształcie pików w układach Mamdani'ego, co bardzo upraszcza defuzzyfikację).

Wartość wyjścia u oblicza się jako *średnią ważoną* wkładów $u^{(i)}$ poszczególnych reguł:

$$u = \frac{\sum_{i=1}^{M} w^{(i)} u^{(i)}}{\sum_{i=1}^{M} w^{(i)}} \tag{1}$$

gdzie wagi $w^{(i)}$ są poziomami spełnienia przesłanek reguł, np. $w^{(i)} = \min\left(\mu_{A_1}^{(i)}(x_1), \mu_{A_2}^{(i)}(x_2), \ldots, \mu_{A_n}^{(i)}(x_n)\right)$

dla *AND-min*, lub $w^{(i)} = \mu_{A_1}^{(i)}(x_1) \cdot \mu_{A_2}^{(i)}(x_2) \cdot \ldots \cdot \mu_{A_n}^{(i)}(x_n)$ w przypadku *AND-prod*.

Obliczenia $u(n) = f[\mathbf{x}(n)]$ są powtarzane dla kolejnych kroków czasowych n.

5. *Fuzzy Logic Toolbox* pakietu MATLAB v.5.3

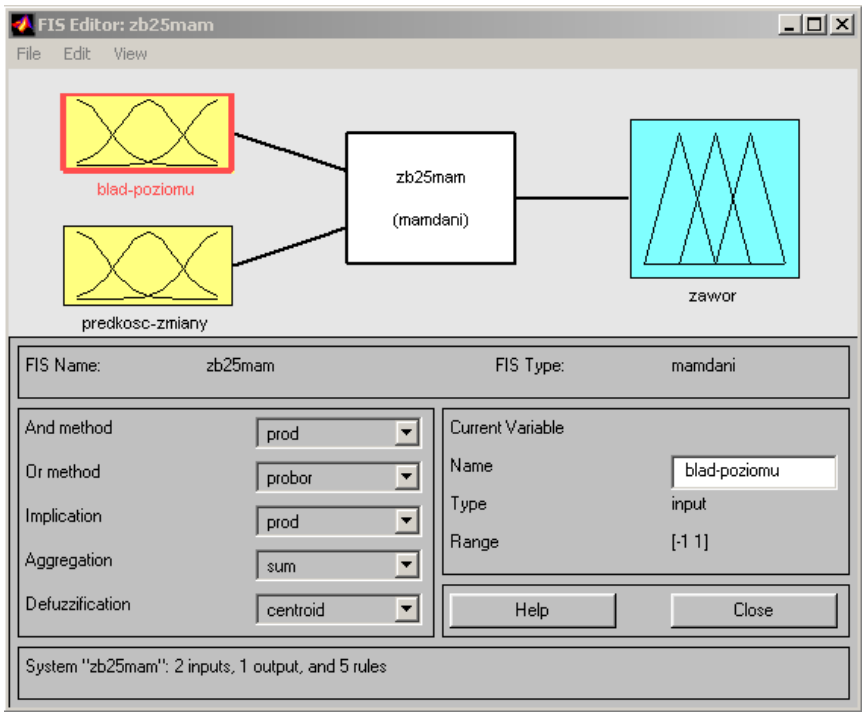
Fuzzy Logic Toolbox jest biblioteką funkcji do projektowania układów wnioskowania rozmytego, tzw. FIS (*Fuzzy Inference Systems*), z graficznym interfejsem użytkownika. Z narzędzia tego można korzystać też poprzez wydawanie poleceń z linii komend MATLABa. Informacja o tworzonym lub modyfikowa-

nym układzie rozmytym jest przechowywana w macierzy, tzw. *FIS matrix*, i może zostać zapisana w pliku `*.fis`. Edycja układu przebiega najprościej w graficznym edytorze FIS (rys.5.), który wywołuje się z linii komend poleceniem:

`>> fuzzy [<nazwa pliku FIS` (bez rozszerzenia) `>]`

Edytor FIS dysponuje szerokim zestawem możliwych do zastosowania kształtów funkcji przynależności, rozmytych operatorów logicznych, metod implikacji i agregacji. każdy z tych elementów może być również zdefiniowany przez użytkownika w postaci funkcji (pliku skryptowego `*.m`) MATLABa. Oprócz układów klasycznych (Mamdani'ego) toolbox umożliwia projektowanie układów Sugeno z wykorzystaniem procedury ANFIS adaptacyjnego doboru parametrów na podstawie danych uczących.

Zmienne wyjściowe i wyjściowe (podobnie jak funkcje przynależności w oknie niższego rzędu) dodaje sie lub usuwa się przy pomocy polecenia menu **Edit | Add input/Add output/Remove variable**. Po dwukrotnym kliknięciu na okienku wybranej zmiennej lub korzystając z menu można przejść do edycji funkcji przynależności wybranej zmiennej: **Edit | Add MF/Remove current MF**. Nazwy i parametry funkcji przynależności można edytować klikając na jej wykres. Dostępnych jest kilka różnych funkcji: trójkątne, trapezowe, gaussowskie, sigmoidalne itp. Reguły podaje się korzystając z edytora reguł (rys.6), który otwiera się po kliknięciu na środkowe okno (*zb25mam*) na rys.5. Reguły mogą być podane w formie językowej lub symbolicznej oraz mieć różne wagi (wszystkie równe 1 na rys.7).



Rys.5. Główne okno edytora FIS *Fuzzy Logic Toolbox*

Bardzo poglądowymi elementami edytora FIS są: okno *Rule Viewer* (rys.7) pokazujące działanie reguł, agregację zbiorów i stan wyjścia dla podanych wartości wejść (można je zmieniać przeciągając myszką pionowe linie) oraz wykres powierzchni sterowania (zmiennej wyjściowej) dla 2 wybranych zmiennych wejściowych.

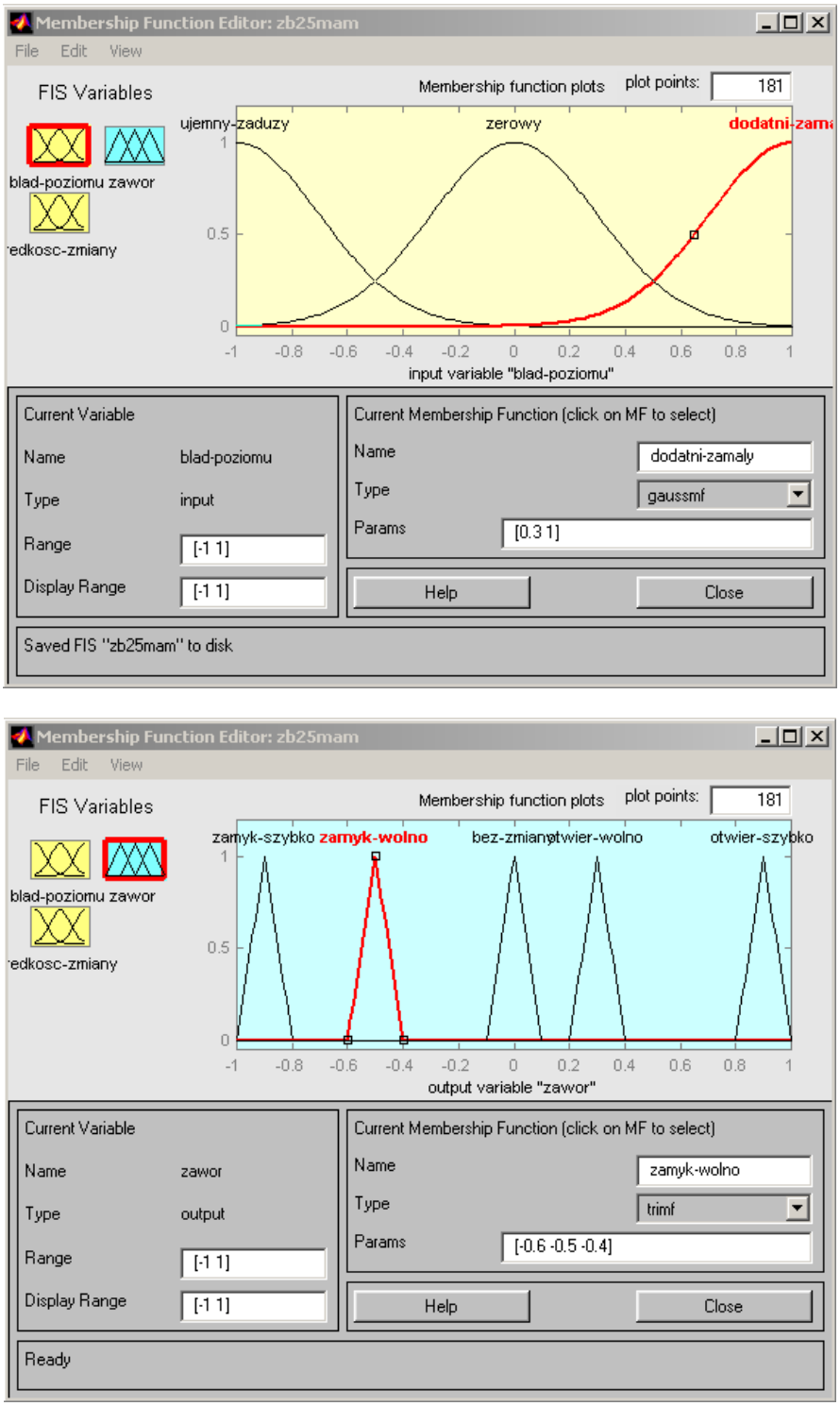
Wybrane funkcje Fuzzy Logic Toolbox jako polecenia linii komend MATLABa:

`>>fismat=readfis('filename')` - wczytanie układu `filename.fis` do zmiennej `fismat`

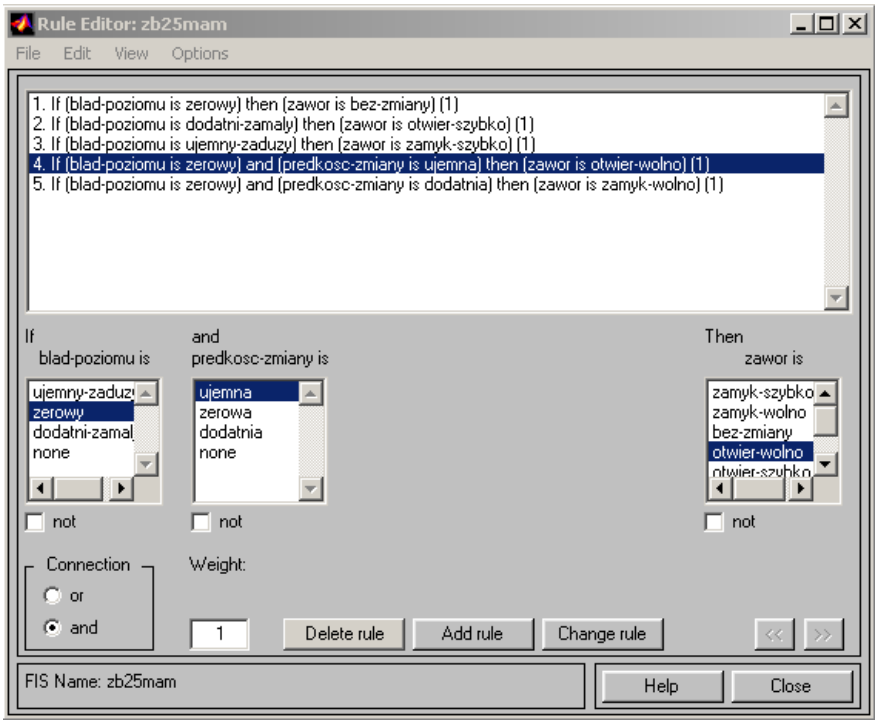
`>>plotfis(fismat)` - drukowanie diagramu wejście-wyjście układu `fismat`

`>>plotmf(fismat,vartype,varindex)`- wykreślanie wszystkich funkcji przynależności zmiennej wejściowej lub wyjściowej: `vartype='input'` lub `'output'`, o numerze `varindex=1,...,k`, gdzie *k* jest liczbą zmiennych na wejściu lub wyjściu.

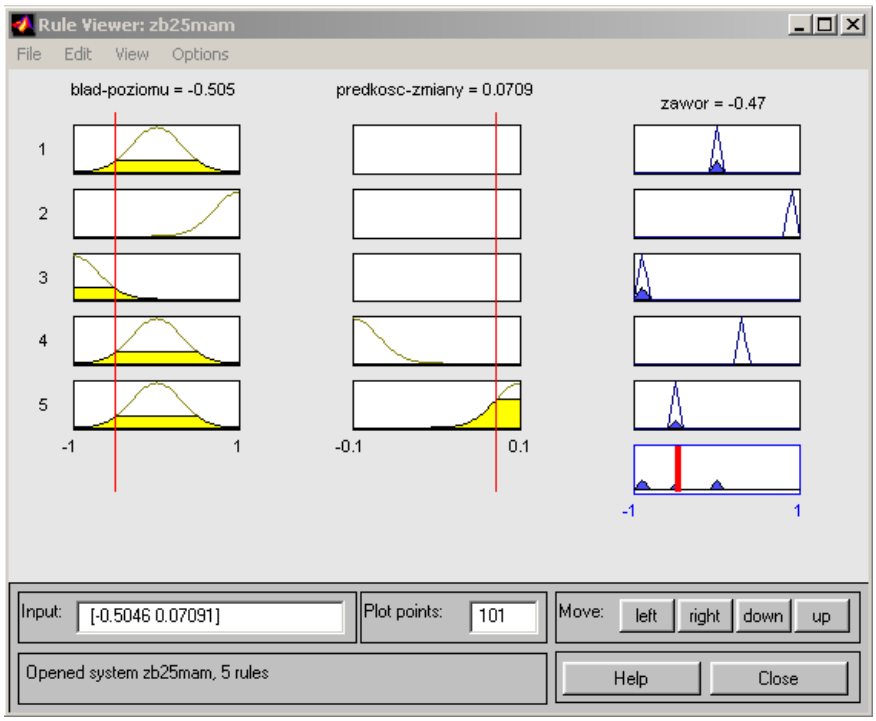
`>>gensurf(fismat,inputs,outputs), surfview(fismat)` - generowanie i rysowanie powierzchni sterowania modelu `fismat` dla dwóch wybranych zmiennych wejściowych (np. `inputs=[1,3]`) i wybranej zmiennej wyjściowej (np. `output=2`).

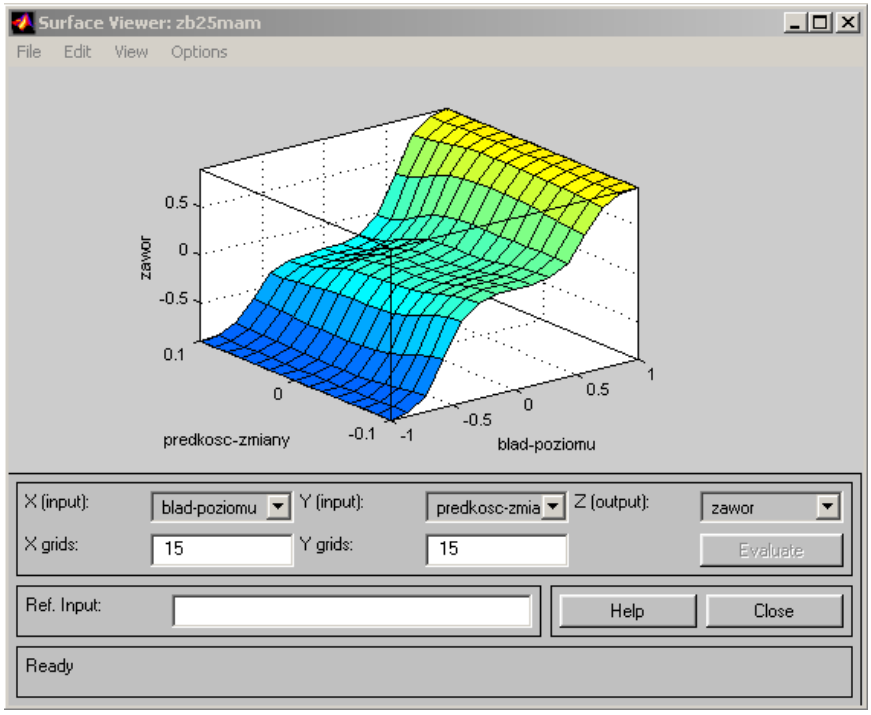


Rys.6. Okna edycji zmiennej wejściowej *blad-poziomu* i zmiennej wyjściowej *zawor* modelu *zb25mam* z rys.5

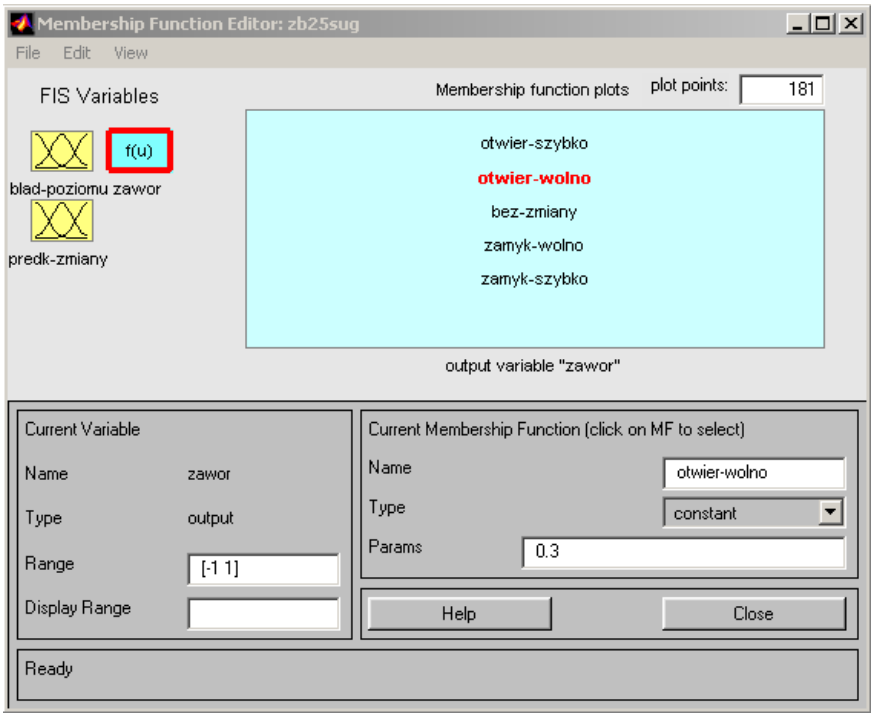


Rys.7. Okno edycji reguł *Rule Editor* modelu *zb25mam*





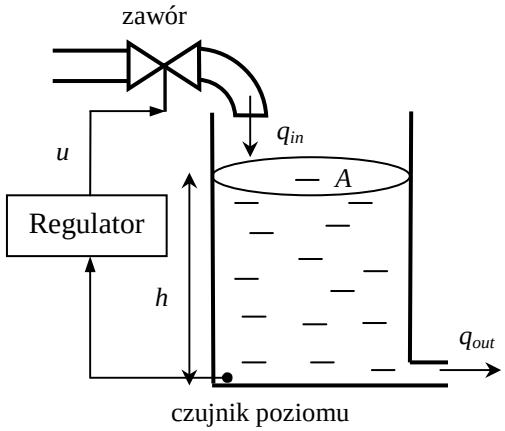
Rys.8. Okno widoku działania reguł *Rule Viewer* i okno wykresu nieliniowej powierzchni sterowania *Surface Viewer* dla modelu *zb25mam*



Rys.9. Okna edycji funkcji zmiennej wyjściowej *zawor* układu rozmytego Sugeno *zb25sug* – odpowiednika układu Mamdani'ego *zb25mam*. Wybrana funkcja *otwier-wolno* jest funkcją stałą (singletonem) w punkcie 0.3. Funkcje przynależności zmiennych wejściowych i reguły obu modeli są takie same.

6. Program ćwiczenia

Zadanie polega na zaprojektowaniu i analizie rozmytego regulatora sterujący poziomem cieczy w nieliniowym modelu zbiornika ze strumieniem wypływającym. **Sygnałem sterującym jest poziom otwarcia zaworu** (znormalizowany do zakresu od 0=zawór zamknięty do 1=zawór w pełni otwarty) sterującego strumieniem wpływającym. Model układu w postaci schematu SIMULINKa jest zapisany w pliku `zbiornik.mdl`.



Rys.10. Schemat układu sterowania poziomem cieczy w zbiorniku z wypływem

Zmiana objętości V (m^3) cieczy w zbiorniku o stałym polu przekroju A (m^2) jest opisana *nieliniowym* równaniem różniczkowym:

$$\frac{dV}{dt} = A\frac{dh}{dt} = q_{in} - q_{out}\,, \tag{2}$$

gdzie: h jest wysokością słupa cieczy w zbiorniku (wielkośc sterowana, m), q_{in} – objętościowym strumieniem wpływającym (m^3/s), q_{out} – objętościowym strumieniem wypływającym ze zbiornika (m^3/s).

Strumień wypływający zależy o ciśnienia słupa cieczy w zbiorniku $\Delta p = \rho g h$ (ρ – gęstość cieczy, kg/m^3, g – przyspieszenie ziemskie, 9.8 m/s^2) i jest opisany równaniem Bernoulliego:

$$q_{out} = \sqrt{\frac{2\cdot\Delta p}{\rho}}\cdot S_{out} = \sqrt{2gh}\cdot S_{out}\,, \tag{3}$$

gdzie S_{out} jest polem przekroju wylotu zbiornika (m^2). Po przekształceniu otrzymujemy równanie dynamiki zmian poziomu zamodelowane w bloku FLUID TANK schematu Simulinka:

$$\frac{dh}{dt} = \frac{1}{A}\left(q_{in} - S_{out}\sqrt{2gh}\right) \tag{4}$$

Równanie zaworu (blok VALVE) regulującego w sposób ciągły strumień wpływający:

$$q_{in} = q_{in\_max}\cdot\int_0^t u\,d\tau \tag{5}$$

gdzie q_{in_max}=const to określony dopływ maksymalny, jest równaniem *układu całkującego z ograniczeniem całkowania* sygnału sterującego do zakresu [0, 1]. W związku z tym regulator jest *regulatorem przyrostowym*, tzn. dla sygnału sterującego u=0 strumień wpływający pozostaje stały: q_{in}=const.

Sygnałem wejściowym regulatora jest błąd poziomu cieczy

$$e = h_{zad} - h\,,$$

gdzie h_{zad} jest zadanym poziomem cieczy w zbiorniku.

1. Otworzyć model symulacyjny Simulinka:

```
>> zbiornik
```

Zarejestrować schemat główny (**Edit | Copy model**) oraz schematy bloków FLUID TANK i VALVE (zaznaczyć blok i wykonac polecenie **Edit | Look under mask,** następnie **Copy model**).

Za pomocą przełącznika *PID/Fuzzy* można wybrać pomiędzy liniowym regulatorem PID (a dokładniej PD z inercją w części różniczkujacej i ograniczeniem sygnału sterującego, położenie górne - bloki pomarańczowe) a jednym z dwóch regulatorów rozmytych (położenie dolne – jeden z bloków niebieskich). Położenie drugiego przełącznika *Fuzzy 1/2* określa, który z dwóch regulatorów rozmytych steruje poziomem przy położeniu *Fuzzy* pierwszego przełącznika.

Regulator rozmyty z jednym wejściem wytwarza sygnał sterujący tylko na podstawie błędu poziomu $e=h_{zad}-h$, jest więc nieliniowym regulatorem proporcjonalnym. Na wejście drugiego regulatora rozmytego (z dwoma wejściami) oprócz błędu poziomu doprowadzona jest również pochodna poziomu cieczy dh/dt z ograniczeniem. Jest to więc nieliniowy regulator PD. Blok *Signal Generator* generuje skokowe zmiany poziomu zadanego h_{zad} w górę i w dół od stałej wartości średniej.

Blok *Scope* pokazuje przebiegi czasowe poziomu zadanego h_{zad} (zielony), poziomu rzeczywistego h (czerwony) i sygnału sterującego u (niebieski).

2. Otworzyć edytor układów rozmytych FIS:

```
>> fuzzy
```

i zaprojektować regulator rozmyty w formie układu Mamdani'ego z uchybem poziomu cieczy jako *jedynym* sygnałem wejściowym. (Opisana poniżej procedura jest skróconym ćwiczeniem tworzenia modelu FIS, jakość działania zaprojektowanego regulatora ma drugorzędne znaczenie.)

- Kliknąć raz na blok sygnału wejściowego (żółty) *input1* i zmienić w oknie *Name* nazwę na *blad-poziomu*.
- Zapisać model FIS do edycji na dysk poleceniem **File | Save to disk as**, np. jako `zb13` (1 wejście, 3 reguły).
- Przejść do edycji funkcji przynależności klikając dwukrotnie na blok wejścia *blad-poziomu*. Zmienić zakres zmienności *Current variable/Range* i *Display range* na [-1 1].
- Utworzyć funkcje przynależności wielkości *blad-poziomu* poleceniem **Edit | Add MFs**: wybrać z listy funkcje gaussowskie `gaussmf` i ich liczbe równą 3.
- Kliknąć na wykresy poszczególnych funkcji przynależności (wykres zostanie pogrubiony) i zmienic ich nazwy domyślne oraz parametry (patrz rys. 6): *mf1→ujemny* (poziom za duży), *Params* [0.4 -1] (szerokość i położenie środka), *mf2→zerowy*, *Params* [0.4 0], *mf3→dodatni* (poziom za duży), *Params* [0.4 1]. Zamknąć okno *Membership Function Editor* i wróćić do okna głównego *FIS Editor*.
- Kliknąć na blok sygnału wyjściowego (niebieski) *output1* i zmienić w oknie *Name* nazwę na *zawor*.
- Przejść do edycji funkcji przynależności klikając dwukrotnie na blok wyjścia `zawor`. Zmienić zakres zmienności *Current variable/Range* i *Display range* na [-1 1].
- Utworzyć 3 funkcje przynależności wielkości *zawor* poleceniem **Edit | Add MFs**: wybrać z listy funkcje trapezowe *trapmf*.
- Kliknąć na wykresy poszczególnych funkcji przynależności i zmienic ich nazwy domyślne oraz współrzędne punktów załamania wykresu: *mf1→zamyk-szybko*, *Params* [-1 -1 -0.8 -0.2], *mf2→bez-zmiany*, *Params* [-0.1 0 0 0.1] (powstaje funkcja trójkątna), *mf3→otwier-szybko*, *Params* [0.2 0.8 1 1]. Zamknąć okno *Membership Function Editor* i wróćić do okna głównego *FIS Editor*.
- Kliknąć dwukrotnie na blok środkowy (biały) z nazwą układu FIS i otworzyć edytor reguł *Rule Editor*. Utworzymy 3 reguły wnioskowania rozmytego:
  1. *if (blad-poziomu is zerowy) then (zawor is bez-zmiany)*
  2. *if (blad-poziomu is dodatni) then (zawor is otwier-szybko)*
  3. *if (blad-poziomu is ujemny) then (zawor is zamyk-szybko)*

Żeby utworzyć regułę nr 1 w oknie edycji przesłanki reguły po lewej stronie (*if blad-poziomu is*) ustawiamy kursor na *zerowy*, a w oknie edycji konkluzji po prawej stronie (*then zawor is*) ustawiamy kursor na *bez-zmiany*. Waga reguły *Weight=1*. Po wyborze tych opcji kliknąć przycisk dodania reguły **Add rule**.

- Pozostałe reguły tworzy się przez wybranie najpierw nazw pożądanych funkcji przynależności w oknach poszczególnych wielkości w przesłance i konkluzji, a następnie dodanie reguły przez **Add rule**. Jeżeli jakaś wielkośc ma nie wystąpić w regule, w jej oknie wybiera się opcję *none*. W przypadku popełnienia błędu można zmienić regułę wybierając ją z numerowanej listy w górnym oknie, wybirac porawne funkcje przynależności w oknach wielkości i dopiero wtedy wprowadzić zmiany naciskając **Change rule**. Przy tworzeniu reguł złożonych w układach z większą liczbą wielkości wejściowych i/lub wyjściowych, w edytorze pojawi się więcej okien z funkcjami przynależności zdefiniowanymi dla każdej z nich. Należy wtedy też wskazać odpowienie logiczne operatory łączące (*Connection*, *and* lub *or*).
- Po utworzeniu wszystkich reguł zamknąć okno *Rule Editor* i wróćić do okna głównego *FIS Editor*.
- W oknie *FIS Editor* wybrać z rozwijanych list metody obliczania reguł (jeżeli chcemy zmienić je na inne niż domyślne): - operacji iloczynu AND (*prod*), operacji sumy OR (*probor*), implikacji (*min* - obciecie), agregacji (*sum* – sumowanie zbiorów rozmytych z konkluzji), defuzzyfikacji (*centroid* – środek ciężkości).
- Zapisać utworzony model FIS na dysk poleceniem **File | Save to disk**.
- Zapisać utworzony model do przestrzeni roboczej Matlaba w pamięci **File | Save to workspace**. **Jest to niezbędne do uruchomienia symulacji modelu** `zbiornik`.
- Zapisać utworzony model FIS na dysk poleceniem **File | Save to disk**.
- Działanie wnioskowania rozmytego utworzonego układu można obejrzeć w oknie podglądu reguł *Rule Viewer* (menu **View | View rules**). Wartość wejścia *blad-poziomu* można zmieniać w oknie liczbowym lub przesuwając myszką pionową czerwoną linię na tle funkcji przynalezności. Spowoduje to przeliczenie zdefiniowanych reguł i wyznaczenie wartości wyjscia *zawor*.
- Powierzchnię nieliniowej statycznej zależności wejście-wyjście (powierzchnię sterowania, w naszym przypadku krzywą *zawor=f*(*blad-poziomu*)) można obejrzeć w oknie *Surface Viewer* (menu **View | View surface**). W przypadku układów z wieloma wejściami i wyjściami można obejrzeć dwuwymiarową powierzchnię zależność wybranego wyjścia od dwóch wybranych wejść.
- Przejść do okna modelu symulacyjnego `zbiornik`, kliknąć dwukrotnie na blok *Regulator Fuzzy-Logic 1 wejscie* i podać jako parametr *FIS matrix* nazwę modelu zapisanego do pamięci, tj. `zb13.`

3. Otworzyć układ rozmyty FIS Mamdani'ego zapisany w pliku `zb25mam.fis`:

```
>> fuzzy zb25mam
```

Jest to regulator poziomu cieczy w zbiorniku z dwoma wejściami: błędem poziomu *e* (*blad-poziomu*) i pochodną poziomu *dh/dt* (*predk-zmiany*), i pięcioma regułami rozmytymi.

- Zapoznać się z funkcjami przynależności wejść i wyjścia oraz regułami układu. Zwrócić uwagę, że dwie spośród reguł mają złożone przesłanki z operatorem rozmytym AND.
- Prześledzić działanie wnioskowania rozmytego w oknie podglądu *Rule Viewer* (menu **View | View rules**).
- Obejrzeć dwuwymiarową powierzchnię sterowania *zawor=f*(*blad-poziomu, predkosc-zmiany*) (menu **View | View surface**).
- Zapisać model do przestrzeni roboczej Matlaba w pamięci **File | Save to workspace**. Zamknąć *FIS Editor*.
- wygenerować i zarejestrować (**Edit | Copy Figure**) schemat układu FIS oraz wykresy funkcji przynależności i powierzchni sterowania. W oknie komend Matlaba wpisać polecenia (po zarejestrowaniu zamykać okno wykresu *Figure*):

```
>>plotfis(zb25mam)              % schemat FIS
>>plotmf(zb25mam,'input',1)     % funkcje przynależności wejścia blad-poziomu
>>plotmf(zb25mam,'input',2)     % funkcje przynależności wejścia predkosc-zmiany
```

```
>>plotmf(zb25mam,'output',1)      % funkcje przynależności wyjścia zawor
>> gensurf(zb25mam,[1 2],1)      % powierzchnia sterowania wy1=f(we1,we2)
>> showrule(zb25mam)             % listing reguł w oknie komend Matlaba
```

- Przejść do okna modelu symulacyjnego `zbiornik`, kliknąć dwukrotnie na blok *Regulator Fuzzy-Logic 2 wejscia* i podać jako parametr *FIS matrix* nazwę modelu, tj. `zb25mam`.

4. Przeprowadzić symulacje modelu `zbiornik` z regulatorami rozmytymi Mamdani'ego.
(**Uwaga:** W przypadku pojawienia się komunikatów o błędach przy starcie symulacji spróbować zamknąć i powtórnie otworzyć model `zbiornik`. Trzeba wtedy pamiętać o wpisaniu właściwych nazw *FIS matrix* jako prametrów bloków *Regulator Fuzzy Logic*)

- Uruchomić symulację z długim horyzontem czasowym: **Simulation | Parameters | Solver***: Stop time*=10000, *Variable-step*, solver *ode23*, *max step size*=0.1. Zaobserwować przebiegi (*Scope* oraz *Wyplyw*, *Pochodna poziomu*) kolejno dla wszystkich trzech regulatorów (PID, *Fuzzy* z 1 wejściem, *Fuzzy* z 2 wejściami) zmieniając *on-line* (w trakcie symulacji) położenia przełączników (dwukrotne kliknięcie na przełącznik).
- Zatrzymać symulację. Skrócić horyzont symulacji zmieniając *Stop time* na 100. Przeprowadzić po kolei symulacje dla trzech regulatorów odpowiednio ustawiając przełączniki i zarejestrować za każdym razem przebiegi wielkości z okna *Scope* (**Edit | Copy Figure**).
  ⇨ Porównać jakość sterowania w poszczególnych przypadkach.

5. Otworzyć w edytorze FIS model Mamdani'ego z dwoma wejściami:

```
>> fuzzy zb25mam
```

i zaprojektować regulator rozmyty w formie *układu Sugeno* będący jego odpowiednikiem.

- Otworzyć nowy układ Sugeno poleceniem **File | New Sugeno FIS**.
- Zapisać nowy „pusty" model na dysk poleceniem **File | Save to disk as**, np. jako `zb25s`.
- Dodać do modelu drugie wejście **Edit | Add input**.
- Zmienić nazwy i zakresy zmiennych wejściowych i zmiennej wyjściowej na takie same jak w układzie *zb25mam*: *input1→blad-poziomu*, *input2→predkosc-zmiany*, *output1→zawor*.
- Dla każdej ze zmiennych *wejściowych* utworzyć w sposób opisany w pkt. 2 funkcje przynależności o nazwach i parametrach takich samych jak w *zb25mam* (części wejściowe obu układów maja być identyczne, wygodnie jest wykorzystać układ *zb25mam* otwarty w edytorze do podglądu):
  1) wejście *blad-poziomu*, zakres *Range* [-1 1], 3 funkcje przynależności *gaussmf* o nazwach *ujemny*, *zerowy*, *dodatni* i parametrach odpowiednio: [0.3 -1], , [0.3 0], [0.3 1].
  2) wejście *predkosc-zmiany*, zakres *Range* [-0.1 0.1], 3 funkcje przynależności *gaussmf* o nazwach *ujemna*, *zerowa*, *dodatnia* i parametrach odpowiednio: [0.03 -0.1], , [0.03 0], [0.03 0.1].
- W oknie edycji funkcji przynależności wyjścia *zawor* ustawić zakres *Range* równy [-1 1] i utworzyć (przez **Edit | Add MFs**) 5 funkcji singletonów (*constant*). Zmienić nazwy tych funkcji na takie jak w *zb25mam* i ustawić w położeniach środków (pików) wąskich trójkątnych funkcji przynależności układu *zb25mam*: *mf1→zamyk-szybko* (*Params* -0.9), *mf2→zamyk-wolno* (*Params* -0.5), *mf3→bez-zmiany* (*Params* 0), *mf4→otwier-wolno* (*Params* 0.3), *mf5→otwier-szybko* (*Params* 0.9).
- Przejść do okna edytora reguł *Rule Editor* (menu **View | Edit rules** lub dwukrotne kliknięcie w okienko z nazwą układu *zb25s (Sugeno)*) i utworzyć w sposób opisany w pkt.2 reguły wnioskowania rozmytego. Ponieważ nazwy wejść i wyjścia oraz ich funkcji przynależności są takie same jak w *zb25mam* forma reguł będzie *identyczna*, tzn. (por. rys. 7):

  1. if (blad-poziomu is zerowy) then (zawor is bez-zmiany) (1)
  2. if (blad-poziomu is dodatni-zamaly) then (zawor is otwier-szybko) (1)
  3. if (blad-poziomu is ujemny-zaduzy) then (zawor is zamyk-szybko) (1)
  4. if (blad-poziomu is zerowy) and (predkosc-zmiany is ujemna) then (zawor is otwier-wolno) (1)
  5. if (blad-poziomu is zerowy) and (predkosc-zmiany is dodatnia) then (zawor is zamyk-wolno) (1)

  Trzeba jednak pamiętać, że w konkluzjach występują tym razem zwykłe funkcje stałe. W obszarze tworzenia przesłanki IF... mamy dwa okna wyboru funkcji przynależności dwóch zmiennych wej-

ściowych. Przy tworzeniu prostych reguł nr 1-3 w oknie wejścia *predk-zmiany* należy wybrać opcję *none*. Połączeniem *Connection* w regułach 4-5 ze złożną przesłanką ma być operacja logicznego iloczynu AND.

- Przejść do okna głównego FIS Editor i ustawić metody obliczania reguł: metoda AND – *prod*, metoda *Defuzzification – wtaver* (średnia ważona, *weighted average*).
- Zapisać utworzony model FIS `zb25s` na dysk poleceniem **File | Save to disk**
- Zapisać utworzony model do przestrzeni roboczej Matlaba w pamięci **File | Save to workspace**.
- Prześledzić działanie wnioskowania rozmytego w oknie podglądu *Rule Viewer* (menu **View | View rules**).
- Obejrzeć dwuwymiarową powierzchnię sterowania *zawor=f*(*blad-poziomu, predkosc-zmiany*) (menu **View | View surface**).
- wygenerować i zarejestrować (**Edit | Copy Figure**) schemat układu FIS oraz wykresy funkcji przynależności wyjścia i powierzchni sterowania. W oknie komend Matlaba wpisać polecenia (po zarejestrowaniu zamykać okno wykresu *Figure*):

```
>>plotfis(zb25s)                    % schemat FIS
>>plotmf(zb25mam,'output',1)        % funkcje przynależności wyjścia zawor
>> gensurf(zb25mam,[1 2],1)         % powierzchnia sterowania wy1=f(we1,we2)
```

6. Przeprowadzić symulacje modelu `zbiornik` z regulatorem rozmytym Sugeno.

- Przejść do okna modelu symulacyjnego `zbiornik`, kliknąć dwukrotnie na blok *Regulator Fuzzy-Logic 2 wejscia* i podać jako parametr *FIS matrix* nazwę modelu Sugeno, tj. `zb25s.`
- Ustawić przełączniki wybierając sterowanie z bloku *Regulator Fuzzy-Logic 2 wejscia*.
- Przeprowadzić symulację jak w pkt. 4 i zarejestrować przebiegi. Zatrzymać symulację.
- Przejść powtórnie do okna edycji funkcji przynależności zmiennej wyjściowej *zawor* modelu *zb25s*, wybrać funkcję *bez-zmiany* i zmienić typ funkcji z *constant* (singleton) na *linear* z parametrami $[c_1\ c_2\ c_0]=[0.5\ 0.5\ 0]$. Oznacza to przyjęcie sterowania liniowego typu PD postaci

$$u = c_1 e + c_2 \frac{dh}{dt}$$

  dla małego błędu poziomu (reguła 1).
- Zapisać zmodyfikowany model do pamięci poleceniem **File | Save to workspace**.
- Przejść do okna modelu symulacyjnego, przeprowadzić symulację ze zmodyfikowanym regulatorem i zarejestrować przebiegi.

7. Zapoznać się z układem stabilizacji (w jednej płaszczyźnie) odwróconego wahadła (*inverted pendulum*) umocowanego dolnym końcem na poruszanej podstawie. Zadaniem sterowania jest generowanie siły *u* zmieniającej położenie *x* podstawy w taki sposób, aby podstawa osiągnęła pozycję zadaną x_{zad} z jednoczesnym zapobiegnięciem przewrócenia się wahadła. Jest to jeden ze standardowych problemów i benchmarków sterowania nieliniowego z niestabilnym obiektem.

- Otworzyć model Simulinka

```
>>fuzinvpen
```

uruchomić symulację i obejrzeć animację działania układu.

W oknie *Scope* pokazane są następujące przebiegi:

- kąt θ odchylenia ramienia wahadła od pionu (niebieski, wejście 1 regulatora rozmytego),
- prędkość kątowa ramienia wahadła $d\theta/dt$ (cyan, wejście 2 regulatora),
- błąd położenia podstawy w stosunku do położenia zadanego x_{zad}- x (magenta, wejście 3 regulatora),
- prędkość liniowa podstawy dx/dt (czerwony, wejście 4 regulatora).

Blok *Control* pokazuje wykres sygnału sterującego *u* z regulatora rozmytego, blok *Position* pokazuje wykres położenia *x* podstawy oraz sygnału położenia zadanego x_{zad}. (Okna wykresów tych bloków są otwierane po dwukrotnym kliknięciu na blok.)

- Zatrzymać symulację, kliknąć na blok *Cart & Pole Dynamics* i obejrzeć zamaskowany model dynamiki wahadła z podstawą przez *Look under mask* z menu kontekstowego (prawy klawisz myszki).
- Otworzyć edytor FIS

```
>>fuzzy
```

i wczytać z pamięci układ FIS regulatora rozmytego poleceniem **File | Open FIS from workspace**, podając nazwę *fismat*. Zapoznać się z regułami i funkcjami przynależności układu Sugeno regulatora. Obejrzeć powierzchnie sterowania w funkcji różnych par zmiennych wejściowych.

7. Opracowanie sprawozdania

W sprawozdaniu należy opisać zarejestrowane wyniki dotyczące symulacji i struktur układów rozmytych FIS. Wyjaśnić problemy wskazane przez prowadzącego.

Literatura

1. Driankov D., Hellendoorn H., Reinfrank M.: *Wprowadzenie do sterowania rozmytego*, WNT, 1996.
2. Łęski J.: *Systemy neuronowo-rozmyte*, WNT, 2008.
3. Piegat A.: *Modelowanie i sterowanie rozmyte*, Akademicka Oficyna Wydawnicza EXIT, 1999.
4. Yager R.R., Filev D.P.: *Podstawy modelowania i sterowania rozmytego*, WNT, 1995.

Opracował:
Dr inż. Janusz Baran