Slovenská technická univerzita v Bratislave
Fakulta chemickej a potravinárskej technológie
Oddelenie informatizácie a riadenia procesov

Bc. Lenka Blahová

Diplomová práca

Výučbový systém pre predmet Inteligentné systémy riadenia

Vedúci diplomovej práce: Doc. Ing. Ján Dvoran, CSc.

Bratislava, máj 2007

ABSTRAKT

Slovenská technická univerzita v Bratislave
Fakulta chemickej a potravinárskej technológie
Oddelenie: Informatizácie a riadenia procesov
Autor: Lenka Blahová
Diplomová práca: Výučbový systém pre predmet Inteligentné systémy riadenia
Vedúci diplomovej práce: Doc. Ing. Ján Dvoran, CSc.
Máj, 2007

Cieľom tejto práce je vytvorenie modulov pre e-learningový výučbový systém pre predmet Inteligentné systémy riadenia v oblasti fuzzy riadenia a umelých neurónových sietí. Práca obsahuje teoretické aspekty problematiky a základy modelovania a riadenia procesov pomocou fuzzy systémov a umelých neurónových sietí. Fuzzy riadenie sa zaoberá hlavne návrhom fuzzy regulátora, popisom jeho častí a typov, ako aj príkladmi využitia fuzzy systémov inač ako spätnoväzbového regulátora. V práci sú uvedené modely neurónových sietí, problémy pri ich výbere a metódy riadenia s popisom ich vlastností. Príklady majú pomôcť pri vytváraní aplikácií modelovania a riadenia procesov v prostredí MATLAB ©.

ABSTRACT

Slovak University of Technology in Bratislava
Faculty of Chemical and Food Technology
Department of Information Engineering and Process Control
Author: Lenka Blahová
Thesis: The learning system for Intelligent Control Systems course
Supervisor: Doc. Ing. Ján Dvoran, CSc.
May, 2007

The present thesis aims at providing the modules for e-learning tutorial system for the course of Intelligent Control Systems, with focus on fuzzy controlling and artificial neural networks. It discusses the necessary theoretical aspects along with the fundamentals of processes' modelling and control by means of fuzzy systems and artificial neural networks. Fuzzy controlling deals mainly with designing the fuzzy controller, describing its components and types as well as providing examples of fuzzy systems usage other than feedback controllers. The work also presents various models of neural networks, addresses the difficulty choosing the proper one, and explains the methods of control including their properties. The examples should help when creating applications of processes' modelling and control in MATLAB ©.

Čestné prehlásenie

Týmto prehlasujem, že túto prácu som vypracovala sama, s pomocou literatúry uvedenej v zozname použitej literatúry a pod odborným dohľadom vedúceho diplomovej práce.

V Bratislave, máj 2007

Poďakovanie

Úprimné poďakovanie za odbornú pomoc a rady pri písaní diplomovej práce by som rada vyjadrila svojmu diplomovému vedúcemu Doc. Jánovi Dvoranovi.

V Bratislave, máj 2007

1. Úvod

Žijeme v dobe, ktorej súčasťou je vyvíjanie množstva technológii slúžiacej k skvalitneniu ľudského života. Vyvíjanie nových technológii vedie nielen k pohode, ale hlavne k zvýšeniu produktivity a pracovného nasadenia. Dôležitou súčasťou technického vývoja sú inteligentné systémy riadenia, ktorých rozvoj napreduje a ponuka nám nespočetné možnosti na použitie, či už v oblastiach technických alebo prírodných. Inteligentné systémy riadenia využívajú umelú inteligenciu alebo jej technológie. Významným rysom inteligentných systémov je schopnosť vytvárať vnútorný model prostredia a pracovať s ním. Medzi inteligentné systémy riadenia patria riadiace techniky založené na fuzzy teórii, umelých neurónových sietí, evolučných algoritmoch, expertných systémoch a pod. Systémy inteligentného riadenia by sa mali vyznačovať plánovaním, rozhodovaním, prácou so symbolickou informáciou atď..

Prvoradou úlohou tejto práce bola snaha o vytvorenie modulov pre e-learningový výučbový systém pre predmet inteligentné systémy riadenia, ktoré by mali pomôcť študentom venujúcim sa problematike umelých neurónových sietí a fuzzy riadeniu a tým zjednodušiť ich ďalší profesionálny vývoj v tomto odbore. Práca má napomáhať pri vytváraní aplikácií v prostredí MATLAB ©, ktoré využívame pri modelovaní a riadení procesov. V dnešnej technicky napredujúcej dobe, kedy sa bez riadenia procesov nezaobíde žiaden priemysel, je dôležité vyškoliť odborníkov venujúcim sa práve tomuto odboru.

Človek sa zriedkavo vyjadruje v presných pojmoch. Používa skôr pojmy približné a nepresné, ako napríklad vysoká teplota či rýchly prietok, a tomu sa venuje fuzzy riadenie. Má široké uplatnenie pri riadení procesov. Používa sa pri procesoch, kde je výhodnejšie použiť veličiny vyjadrené slovne. Tento druh riadenia sa zakladá na teórii fuzzy množín, ktorá je bližšie popísaná a rozvinutá v kapitole 2.2. Ďalej je v práci popísaný fuzzy regulátor a návrh jeho syntézy, ako aj spôsob zapojenia do regulačného obvodu. Široké využitie fuzzy riadenia je možné v spojení s ďalšími inteligentnými systémami riadenia, ako sú napr. neurónové siete.

Ďalšou nemenej dôležitou problematikou, ktorá je v práci popísaná sú umelé neurónové siete, ktoré tak trochu poznáme z sci-fi filmov, kde táto technológia tvorí

základ všetkých umelo mysliacich a umelo vytvorených bytostí. Používame ich v systémoch, ktorých popis je veľmi zložitý alebo mimoriadne náročný, pretože aj keď nemáme systém popísaný máme dáta, ktoré do systému vstupujú a vystupujú. V práci sú uvedené rôzne modely umelých neurónových sietí, ako aj rôzne spôsoby riadenia pomocou týchto sietí.

2. Fuzzy riadenie

2.1 Fuzzy logika

Fuzzy logika je matematická metóda riadenia a regulácie. Dôležitosť fuzzy logiky vyplýva z faktu, že väčšina spôsobov ľudského myslenia a predovšetkým myslenie zdravého rozumu, sú prirodzene približné.

Fuzzy logika sa po prvýkrát objavila v roku 1965 v článku, ktorého autorom bol profesor Lotfi A. Zadeh. Vtedy bol definovaný základný pojem fuzzy logiky a to fuzzy množina. Slovo fuzzy znamená neostrý, matný, hmlistý, neurčitý. Odpovedá tomu i to, čím sa fuzzy teória zaujíma: snaží sa pokryť realitu v jej nepresnosti a neurčitosti. [2]

Booleovská logika uvádza, že pohár môže byť buď naplnený vodou, alebo prázdny. Predstavme si, že máme pohár naplnený vodou do polovice. Potom pohár môže byť poloplný, ale aj poloprázdny. Tento príklad jednoznačne vyvracia Aristotelov zákon o bivalencii. Tento koncept určitého stupňa alebo viachodnotovosti (multivalencie) je základný koncept, ktorý využil *prof. Zadeh* pri predstavení fuzzy logiky. Základné charakteristiky fuzzy logiky podľa neho sú:

- Presné myslenie sa chápe ako špeciálny prípad približného myslenia
- Všetko je otázkou stupňa
- Každý logický systém môže byť fuzzyfikovaný
- Znalosť je reprezentovaná ako súbor elastických, alebo ekvivalentných fuzzy ohraničení na súbore premenných
- Inferencia je chápaná ako proces množenia elastických ohraničení

Booleovská logika je podmnožinou fuzzy logiky. [4]

2.2 Fuzzy množiny

Základným predpokladom je prevod verbálnych prvkov (nepresné pojmy, napr. vysoký človek, nízka rýchlosť) do kvantifikovanej stupnice. Idea fuzzy množiny je veľmi jednoduchá a prirodzená. Ak nie sme schopný stanoviť presné hranice množiny určenej nepresným pojmom, nahradíme rozhodnutie o zaradení alebo nezaradení konkrétneho prvku do danej množiny určitou mierou vybranou z predtým definovaného intervalu. V klasickej teórii množín prvok do množiny buď patrí (úplné členstvo v množine) alebo nepatrí (žiadne členstvo v množine). Fuzzy množina je množina, ktorá okrem úplného alebo žiadneho členstva pripúšťa aj členstvo čiastočné. To znamená, že prvok patrí do množiny s istou pravdepodobnosťou (stupeň príslušnosti). Funkcia, ktorá každému prvku priradí stupeň príslušnosti sa nazýva funkcia príslušnosti. Vzhľadom ku klasickej teórii množín máva stupeň príslušnosti hodnotu z intervalu <0; 1>. [2]

Nech U je súbor objektov (prvkov) označovaných symbolom x. Nech M je množina čísel, na ktorej je definovaný zväz. Fuzzy množina A je množina usporiadaných dvojíc

$$A = \{(x, \mu_A(x)), x \in U\} \tag{2.1}$$

kde $\mu_A(x): U \to M$ je funkcia príslušnosti, $\mu_A(x)$ je stupeň príslušnosti objektu (prvku) do fuzzy množiny A. Súbor U sa nazýva aj univerzálna množina - univerzum. Ak $M = \{0,1\}$ tak sa jedná o klasickú (ostrú) množinu. T.j. klasické množiny sú špeciálnym prípadom fuzzy množiny. Zvyčajne je M uzavretý interval reálnych čísel $\langle 0,1 \rangle$, kde $\mu_A(x) = 0$ je najmenší a $\mu_A(x) = 1$ je najväčší stupeň príslušnosti. Fuzzy množina A je prázdna ak $\forall x \in U$ je $\mu_A(x) = 0$. Ak $\mu_A(x) = 0$, potom prvok x nepatrí do fuzzy množiny A ($x \notin A$).

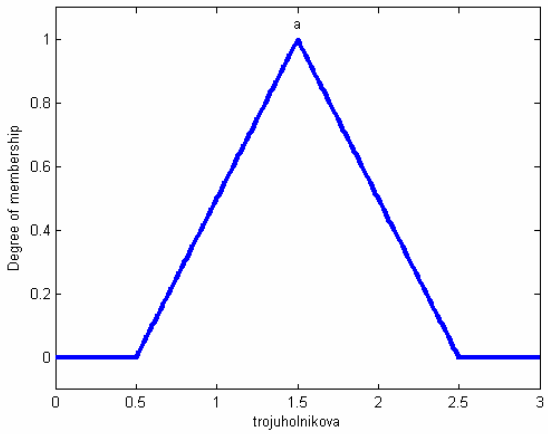
Fuzzy premenná je usporiadaná trojica $(X, U, R(X,u))$, kde X je názov premennej, U je univerzum a $R(X,u)$ je fuzzy podmnožina univerza, ktorá predstavuje fuzzy ohraničenie hodnôt premennej. Lingvistická premenná je usporiadaná pätica $(T, T(T), U, G, M)$, kde T je názov premennej; $T(T)$ je term-množina, t.j. množina všetkých názvov lingvistických hodnôt premennej T, pričom každá z týchto hodnôt je

fuzzy premennou X, a nadobúda hodnoty z univerza U; G je syntaktické pravidlo (zvyčajne gramatika), podľa ktorého je zostavený názov X hodnoty lingvistickej premennej T; M je sémantické pravidlo, ktoré priraďuje každej fuzzy premennej X jej význam $M(X)$, t.j. fuzzy podmnožinu $M(X)$ univerza U [3].

Typy funkcií príslušnosti:

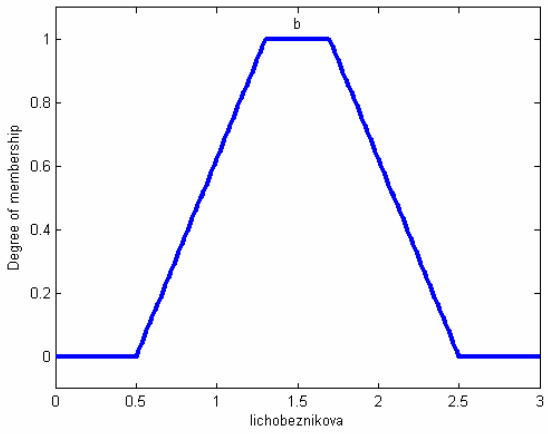
Trojuholníková funkcia príslušnosti

$$\mu(x) = a_1 x + b_1 \qquad x \in \langle x_1, x_2 \rangle \qquad a_1 > 0$$
$$\mu(x) = a_2 x + b_2 \qquad x \in \langle x_2, x_3 \rangle \qquad a_2 > 0$$
$$\mu(x) = 0 \qquad ine\ x$$



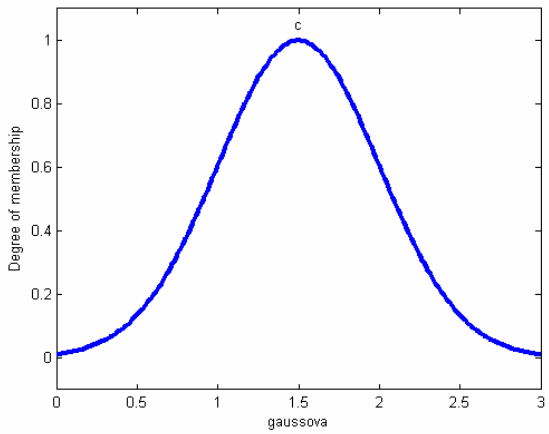
Lichobežníková funkcia príslušnosti

$$\mu(x) = a_1 x + b_1 \qquad x \in \langle x_1, x_2 \rangle \qquad a_1 > 0$$
$$\mu(x) = 1 \qquad x \in \langle x_2, x_3 \rangle$$
$$\mu(x) = a_2 x + b_2 \qquad x \in \langle x_3, x_4 \rangle \qquad a_2 > 0$$
$$\mu(x) = 0 \qquad ine\ x$$



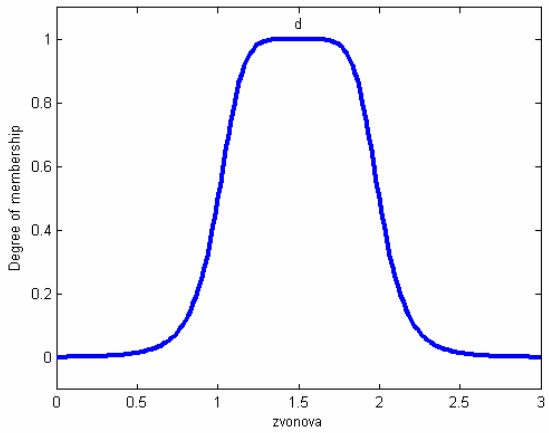
Gaussova funkcia príslušnosti

$$\mu(x) = e^{-K(s-x)^2}$$



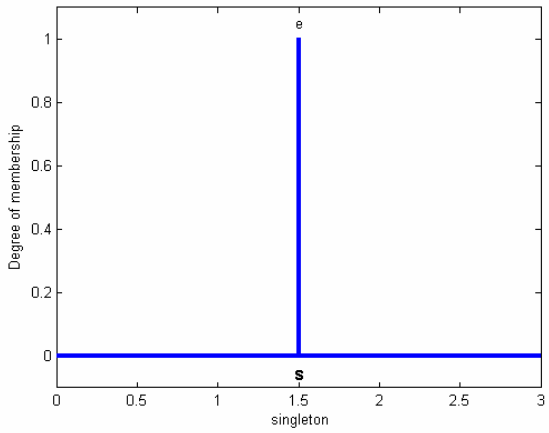
Zvonová funkcia príslušnosti

$$\mu(x) = \frac{1}{1+\left(\frac{x-s}{\beta}\right)^2}$$



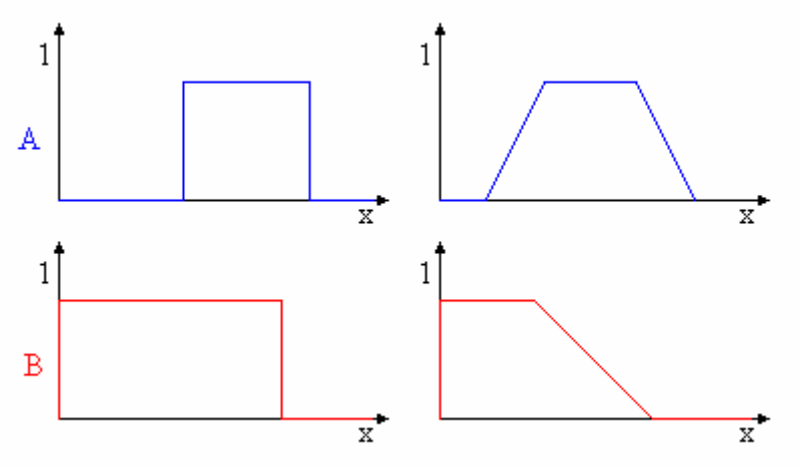
Singleton

$$\mu(x) = 1 \qquad x = s$$
$$\mu(x) = 0 \qquad x \neq s$$



2.2.1 Operácie s fuzzy množinami

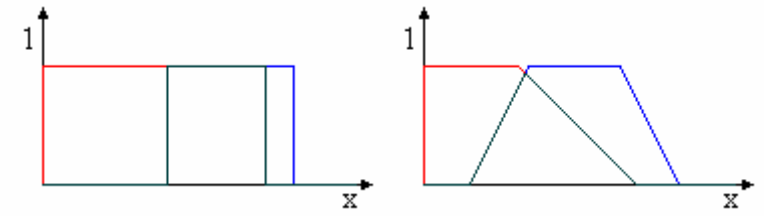
Majme A a B fuzzy podmnožiny neprázdnej množiny X (obr. 2.1).



Obr. 2.1 Ostré a fuzzy množiny A a B

Prienik A a B je definovaný nasledovne (obr. 2.2):

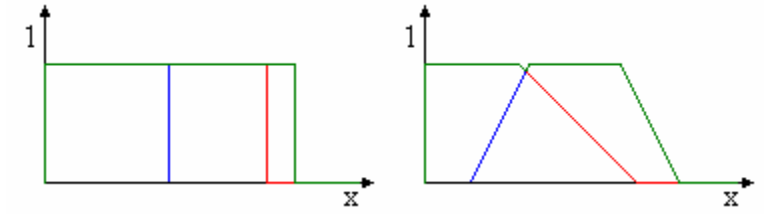
$$A \cap B : \mu_{A \cap B} = \min\{\mu_A(X); \mu_B(X)\} pre \forall x \in X \qquad (2.2)$$



Obr. 2.2 Prienik A a B

Zjednotenie množín A a B je definované takto (obr. 2.3):

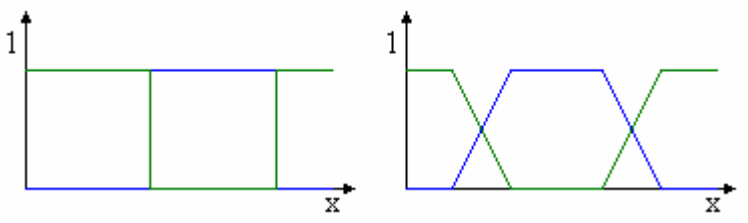
$$A \cup B : \mu_{A \cup B} = \max\{\mu_A(x); \mu_B(x)\} pre \forall x \in X \qquad (2.3)$$



Obr. 2.3 Zjednotenie A a B

Doplnok fuzzy množiny A je definovaný ako (obr. 2.4):

$$A': \mu_{A'} = 1 - \mu_A(x) pre \forall x \in X \quad (2.4)$$



Obr. 2.4 Doplnok fuzzy množiny A

2.3 Lingvistické premenné

Lingvistické (jazykové) premenné sú fuzzy premenné. Napríklad výraz *John je vysoký* implikuje, že lingvistická premenná *John* obsahuje lingvistickú premennú *vysoký*. Vo fuzzy expertných systémoch sú lingvistické pravidlá používané vo fuzzy pravidlách. Napr.

IF vietor je silný THEN plavba bude dobrá

IF trvanie projektu je dlhé THEN risk je vysoký

IF rýchlosť je pomalá THEN brzdná dráha je krátka.

Aj v neostrej logike je možné definovať kvantifikátory, ktoré zvyšujú vyjadrovaciu účinnosť. Zaujímavé je, že okrem kvantifikátorov pre „*všetky*" a „*existuje*" možno definovať kvantifikátory ako *spravidla, často, mnoho, málo*.

Príkladom modifikátorov vo fuzzy:

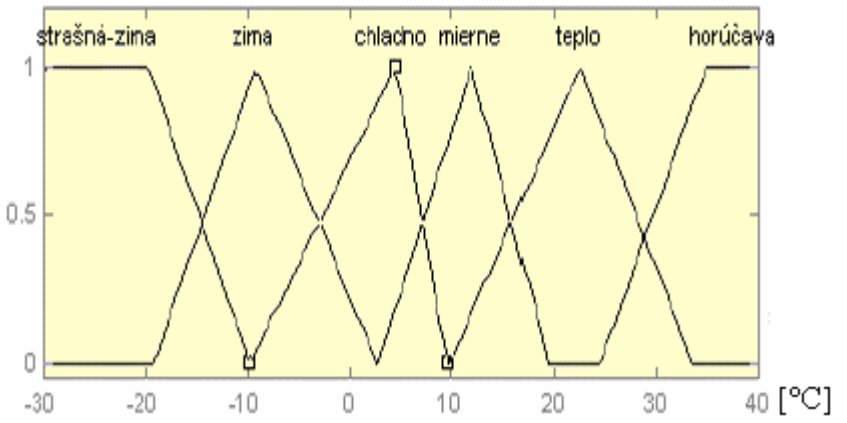
- *veľmi, úplne, extrémne* – všeobecné modifikátory
- *úplne pravdivé, väčšinou pravdivé* – pravdivostné modifikátory
- *pravdepodobne, nie veľmi pravdepodobne* – pravdepodobnostné modifikátory
- *často, niekedy, málo* – kvantitatívne modifikátory
- *skoro nemožné, úplne nemožné* – prijateľnostné modifikátory

Takéto lingvistické ohraničenia môžu byť reprezentované matematickými operátormi. Ak napr. μ je charakteristická funkcia, tak ohraničenie „*veľmi*“ (very) modifikuje túto funkciu kvadraticky: μ^2, a modifikátor *trocha* (little), odmocninou charakteristickej funkcie: $\sqrt{\mu}$ [5].

Lingvistická premenná (jazyková premenná)

- je slovne vyjadriteľná veličina (premenná) napr. teplota v miestnosti, rýchlosť nárastu tlaku, výkonnosť zamestnancov firmy, cena auta, ...
- je definovaná:
  - *názvom* – krátky, výstižný, jednoznačný
  - *množinou lingvistických hodnôt* (tzv. „termov“), ktoré sú jednoznačne určené fuzzy množinami (funkcie príslušnosti)
  - *univerzom* (definičným oborom), univerzum je ľubovolná usporiadaná množina (reálne-číselná, celočíselná, ... , množina zamestnancov firmy, ...)

Príklad: „teplota okolia“



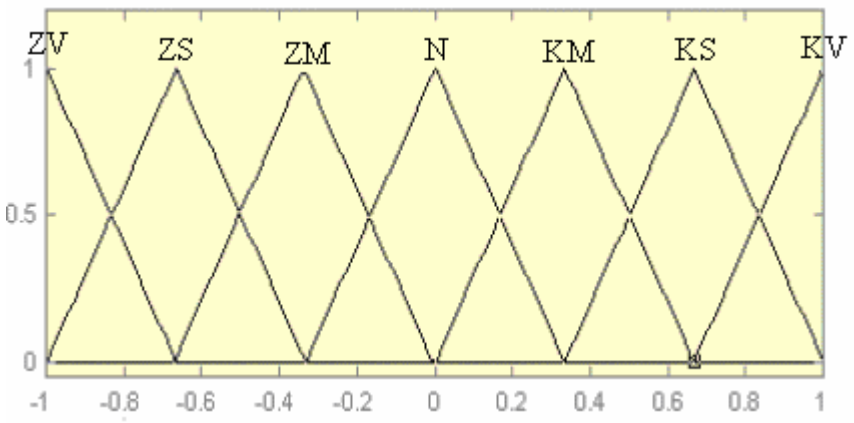
Obr. 2.5 Príklad počtu a rozloženia funkcií príslušnosti a lingvistických premenných

Počet lingvistických hodnôt (termov, fuzzy množín) sa volí tak, aby jednotlivé prípady boli dostatočne rozlíšené [11].

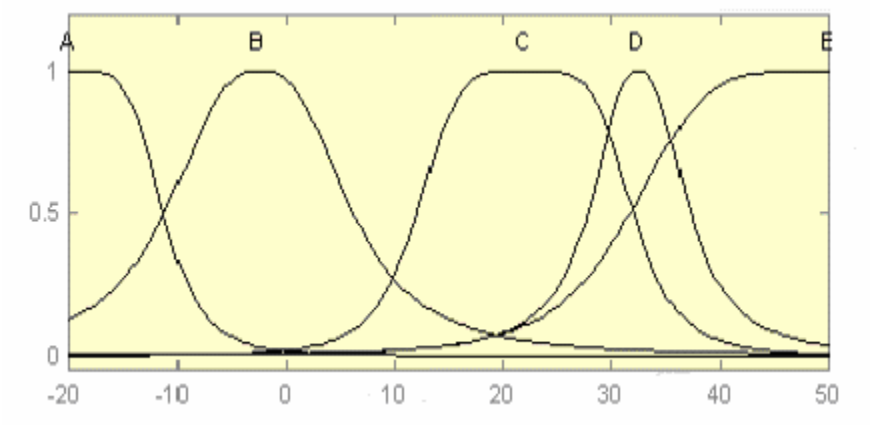
Všeobecné označenie termov:

- Z – záporný (negative)

- K – kladný (positive)
- N – nulový (zero)
- V – veľký (big)
- S – stredný (medium)
- M – malý (small)



Obr. 2.6 Rovnomerné rozloženie, symetrické, trojuholníkové funkcie príslušnosti v normovanom tvare $\langle -1;1 \rangle$ (nemusí byť)



Obr. 2.7 Nerovnomerné rozloženie, nepravidelné funkcie príslušnosti

2.4 Fuzzy riadenie a modelovanie

Klasické modely systémov sú postavené na základe vzťahu medzi vstupom a výstupom systému. Fuzzy modely toto klasické pojatie transformujú do skupiny pravidiel. Lingvisticky popísaný model je vyjadrený ako skupina pravidiel IF – THEN s neurčitým tvrdením. Je teda založený na znalostiach.

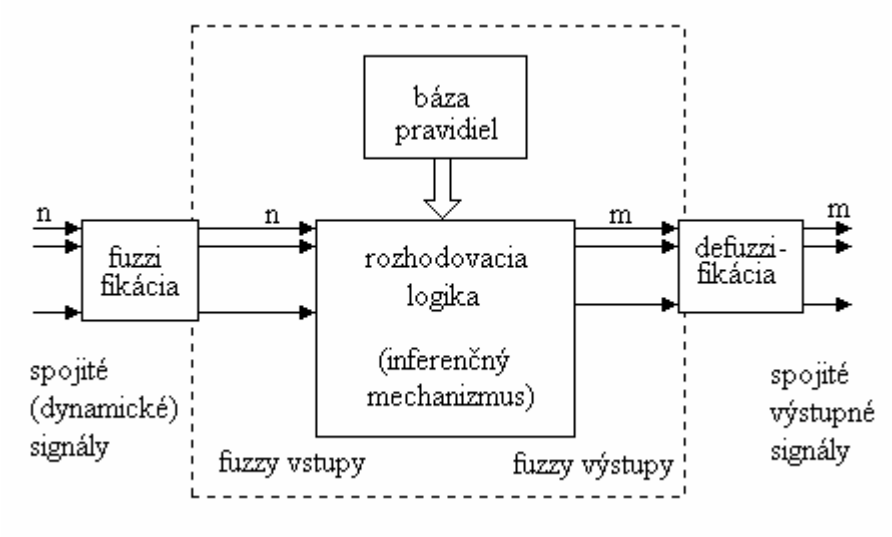
Fuzzy systém pri riadení [11]

- reprezentuje riadiaci alebo rozhodovací systém, alebo reprezentuje model objektu riadenia
- umožňuje pracovať so systémami formou podobnou prirodzenému ľudskému jazyku a ľudskému uvažovaniu
- je možné ľahko aplikovať znalosti (skúsenosti, intuíciu, technický cit ...) ľudského typu – „expertné znalosti“
- fuzzy veličiny sú kvantifikovateľné lingvistickými hodnotami (fuzzy množinami opísanými pomocou funkcií príslušnosti)

Z praktického pohľadu využitia fuzzy aproximácií a ich vlastností pre riadenie a reguláciu je treba modifikovať nasledujúce kroky, ktoré sú zobrazené na obr. 33:

1. Zmerať vstupné veličiny.
2. Zobraziť zmerané veličiny vo vhodnej mierke na aplikované univerza.
3. Previesť vstupné ostré dáta na fuzzy dáta - fuzzifikácia.
4. Nájsť výstupnú fuzzy množinu.
5. Priradiť – nájsť k výstupnej množine vhodnú ostrú hodnotu akčnej veličiny - defuzzifikácia.

Fuzzy systém slúži na interpretáciu lingvisticky formulovaných znalostí, ktoré sú obsiahnuté v báze pravidiel (obr. 2.8).



Obr. 2.8 Fuzzy systém

Problémy klasického riadenia [6]

- nedostatočné teoretické znalosti dejov technologického procesu (z týchto dôvodov sa riadenie realizuje ručne na základe empirických informácií a skúseností)
- nelinearita
- obmedzenie riadiacich veličín
- ďalšie požiadavky na riadenie (napr. nulové preregulovanie)
- parametre nie sú presne známe (alebo sú obtiažne merateľné)
- citlivosť na zmeny parametrov a vstupných hodnôt (každé aj malé zhoršenie kvality riadenia prináša straty energie alebo zhoršenie kvality produktu)
- diskretizácia
- oneskorenie (napr. pri výpočte riadiacich veličín)
- model nepostihuje všetky dôležité vzťahy alebo je drasticky zjednodušený
- problémy s riešiteľnosťou
- chýba interpretácia parametrov regulátorov

2.4.1 Fuzzy pravidlá

Človek získava poznatky (učí sa) [11]:

- empiricky, vlastnými experimentami alebo na základe pozorovania iných
- verbálne získanými poznatkami (výchova, rozhovor, rádio, TV,...)
- štúdiom literatúry, internet...
- zhromažďovaním a spracovaním poznatkov sa formujú „znalosti“

Všeobecne je logické riadenie založené na vyhodnocovaní rozhodovacích pravidiel vo forme

AK ... POTOM

Pre fuzzy riadenie a reguláciu je podmienka vyjadrená formou implikácie dvoch fuzzy výrokov väčšinou ako

AK <fuzzy výrok> POTOM <fuzzy výrok>

V anglickej verzii potom

IF <fuzzy výrok> THEN <fuzzy výrok>.

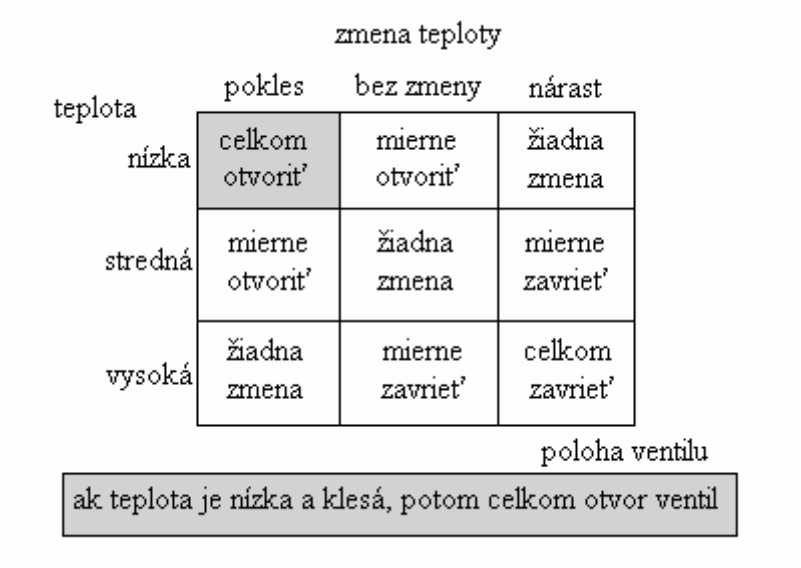
Táto podmienka je označovaná ako „produkčné pravidlo, ak-potom“. Prvá fuzzy výroková množina, ktorou je často zložený výrok, sa nazýva antecedent, kde jednotlivé časti výrokov sú viazané logickými spojkami. Druhý fuzzy výrok je konsekvent.

Štruktúra pravidla

Ak predpoklad potom dôsledok

(if antecedent then consequent)

V niektorých prípadoch je možné pravidlá vyjadriť vo forme tabuľky (obr. 2.9):



Obr. 2.9 Pravidlá vyjadrené formou tabuľky

Množina všetkých pravidiel v danej aplikácii sa nazýva „báza pravidiel“.

Počet pravidiel

Pre dvojrozmernú funkčnú závislosť lingvistických premenných X, Y inferenčné pravidlá tvoria dvojice, ktoré patria do množiny A x B, ktorá je daná karteziánskym súčinom

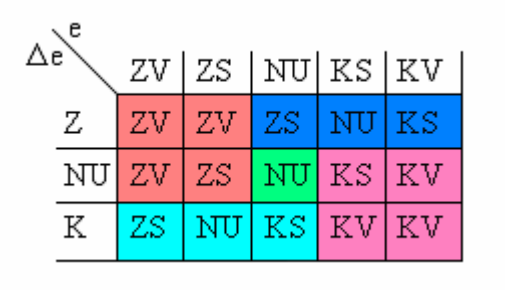
$$P = \{(x, y) / x \in A, y \in B\} \tag{2.5}$$

Počet pravidiel pre dve fuzzy veličiny (dvojrozmerná závislosť: regulačná odchýlka e a zmena regulačnej odchýlky Δe) vysvetlíme nasledovne.

Regulačná odchýlka e má 5 lingvistických hodnôt – termov (ZV, ZS, NU, KS, KV). Zmena regulačnej odchýlky Δe má 3 lingvistické hodnoty – termy (Z, NU, K). Pretože „regulačná odchýlka e“ je fuzzyfikovaná piatimi termami a „zmena regulačnej odchýlky Δe“ má tri termy, je celkový počet pravidiel P 5x3=15, viď obr. 2.10. Pre počet pravidiel platí

$$P = nxm \tag{2.6}$$

kde m a n je počet termov fuzzy množín. Zmena akčnej veličiny Δu má 5 lingvistických hodnôt termov (ZV, ZS, NU, KS, KV).



Δe \ e	ZV	ZS	NU	KS	KV
Z	ZV	ZV	ZS	NU	KS
NU	ZV	ZS	NU	KS	KV
K	ZS	NU	KS	KV	KV

Obr. 2.10 Báza pravidiel

V báze pravidiel je možné rozlíšiť 5 skupín pravidiel:

1.	skupina - Táto skupina pravidiel sa použije vtedy, ak regulačná odchýlka e a jej zmena Δe je nulová alebo blízka nule. Znamená to, že regulovaná sústava je v ustálenom stave alebo v jeho blízkosti. *Akčná veličina sa nemá meniť, čiže zmena akčnej veličiny je nulová alebo blízka nule*.
2.	skupina - Pre aplikáciu pravidiel tejto skupiny platí, že regulačná odchýlka e je záporná (veľká alebo stredná) a jej zmena Δe je kladná alebo blízka nule. Dôsledkom toho je, že regulačná odchýlka e sa **zmenšuje alebo sa nemení.** *Akčný zásah má zrýchliť alebo spomaliť približovanie k ustálenej hodnote.*
3.	skupina - Pre túto skupinu platí, že regulačná odchýlka e je kladná (blízka nule, stredná, veľká). Zmena Δe je kladná (veľká alebo stredná), čo znamená, že regulovaná veličina sa **bude** vzďaľovať od žiadanej hodnoty – ustáleného stavu. *Kladnou zmenou akčnej veličiny Δu je treba zaistiť približovanie k ustálenému stavu.*
4.	skupina - Pre aplikáciu pravidiel tejto skupiny je charakteristické, že regulačná odchýlka e je kladná (veľká alebo stredná) a jej zmena Δe je záporná alebo nulová. To znamená, že regulačná odchýlka e sa **zmenšuje alebo sa nemení**. *Akčný zásah má zrýchliť alebo spomaliť približovanie k ustálenej hodnote.*
5.	skupina - Pre túto skupinu platí, že regulačná odchýlka e je záporná (blízka nule, stredná, veľká). Zmena Δe je **záporná veľká alebo stredná**. To znamená, že regulovaná veličina sa **bude vzďaľovať** od žiadanej hodnoty – ustáleného stavu. *Zápornou zmenou akčnej veličiny Δu je treba zaistiť približovanie k ustálenému stavu*

Veľká väčšina jednoduchých fuzzy regulátorov má bázu pravidiel založenú na použití uvedených pravidiel. Bázu pravidiel možno ľahko modifikovať pre iný počet termov regulačnej odchýlky a jej zmeny.

Priebeh regulačného pochodu ovplyvňuje okrem bázy pravidiel tiež zvolené tvary funkcií príslušnosti a zvolená metóda defuzzifikácie. V prípade, že priebehy regulačných pochodov nevyhovujú našim požiadavkám, je treba hľadať nové rozhodovacie pravidlá, použiť iné metódy defuzzifikácie a vhodne upraviť funkcie príslušnosti [1].

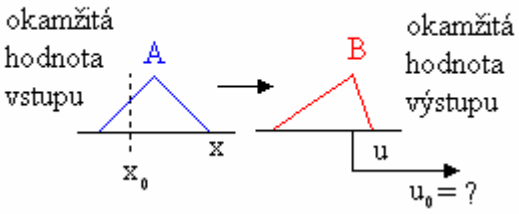
2.4.2 Fuzzifikácia

Proces priraďovania meraných hodnôt vstupujúcich veličín do fuzzy množín pomocou funkcií príslušnosti sa označuje ako fuzzifikácia.

2.4.3 Inferenčný mechanizmus [11]

Fuzzy implikácia

Ak x je A potom u je B

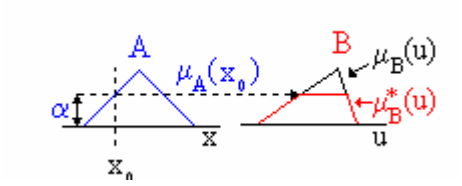


Mamdaniho implikácia

$\mu_B^*(u) = \mu_B(u) \quad ak\ \mu_B(u) < \alpha, \quad inak\ \mu_B^*(u) = \alpha$

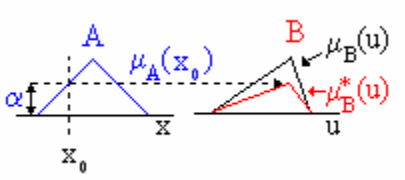
„obmedzenie funkcie príslušnosti na alfa hodnotu“

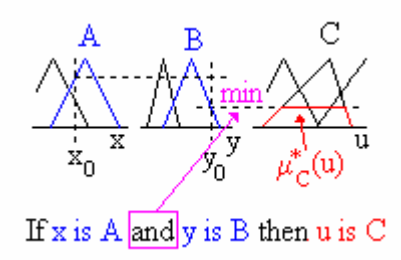
$\mu_B^*(u) = \min[\alpha, \mu_B(u)]$

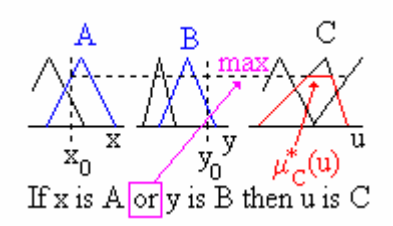


Larsenova implikácia

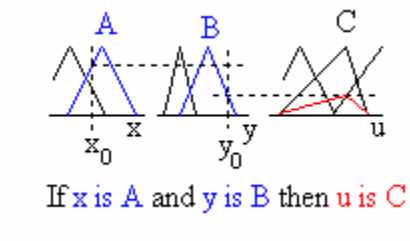
$\mu_B^*(u) = \mu_B(u).\alpha$ .... „zníženie $\mu_B(u)$ alfa krát“







Pre Larsenovu implikáciu



Obr. 2.11 Viac podmienok v antecedente

2.4.4 Kompozícia

Zjednotenie konsekventov jednotlivých pravidiel. Vo fuzzy systéme medzi jednotlivými pravidlami platí operácia alebo [11].

P_1: ak x je A_1 a y je B_1 ... potom u je C_1

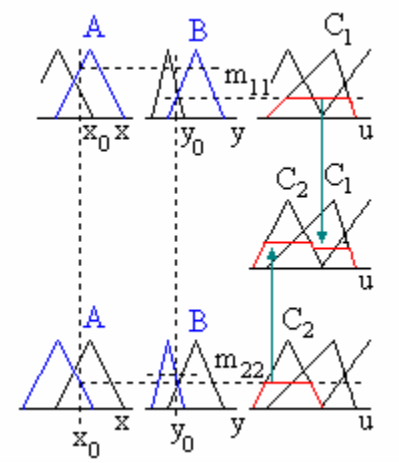
Alebo

P_2: ak x je A_2 a y je B_2 ... potom u je C_2 ...

... P_m

Uplatňujú sa všetky pravidlá, ktorých funkcie príslušnosti boli adresované.

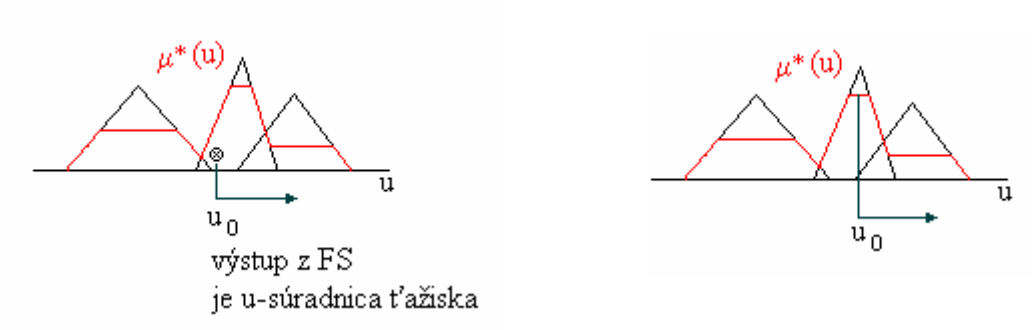
Kompozícia: $\mu(u) = \max\{\mu_1(u), \mu_2(u), ..., \mu_m(u)\}$ (2.7)



Obr. 2.12 Zjednotenie konsekventov jednotlivých pravidiel

2.4.5 Defuzzifikácia

Pre praktickú modifikáciu akčných zásahov je treba priradiť výstupným lingvistickým premenným ostrú hodnotu akčnej veličiny v prípustnom rozsahu. Tento proces „aproximácie neostrých termov“ ostrou hodnotou akčnej veličiny sa nazýva defuzzifikácia. Existuje cely rad metód defuzzifikácie, ktoré vychádzajú z empirického overenia až po heuristické prístupy. Pri voľbe metódy defuzzifikácie môžeme zvoliť buď metódy, ktoré hodnotu akčnej veličiny určia výpočtom ako najlepší kompromis (metódy ťažiska) alebo metódy hľadajúce prijateľné riešenie (metódy najvýznamnejšieho maxima), viď obr. 2.13 [1].



a) b)

Obr. 2.13 a) metóda ťažiska, b) metóda maxima

Požiadavky na defuzzifikáciu [6]

- spojitosť
- jednoznačnosť
- nízka výpočtová zložitosť
- prijateľnosť (výsledná hodnota by mala byť zhruba uprostred nosiča a mať veľký stupeň príslušnosti)
- spočítanie váh? (Ak sa prekrývajú konsekventy niekoľkými použiteľnými pravidlami, má sa ich účinok spočítať?)

Problémy defuzzifikácie [6]**:**

- viacnásobné maximá
- spojitý prechod medzi pravidlami
- ak nosiče konsekventov nie sú obmedzené, rozšírenie univerza môže spôsobiť zmenu výstupov

2.4.6 Fuzzy riadenie

Pri priamom fuzzy riadení je fuzzy systém použitý ako prvok regulačného obvodu, ktorý nejakým spôsobom priamo ovplyvňuje riadený objekt [11].

Vstupmi pri fuzzy riadení sú obyčajne:

- e(t) alebo y(t)
- $\frac{de(t)}{dt}$ resp. $\Delta e(k)$ (de)
- $\frac{d^2e(t)}{dt}$ resp. $\Delta^2 e(k)$ (d^2e)
- $\int e(t)dt$ resp. $\sum e$ ($\int e$)

Prípadne iné veličiny podľa typu aplikácie ako poruchy, žiadané hodnoty, pomocné informácie ...

Jeho výstupy sú vo forme:

- absolútnej (polohovej): výstup z FS je priamo u(t) resp. u(k)
- alebo prírastkovej (rýchlostnej): výstup z FS je $\frac{du(t)}{dt}$ resp. $\Delta u(k)$
- vstup do riadeného systému je $u(k) = u(k-1) + \Delta u(k)$

Najčastejšia realizácia fuzzy systému pri riadení je:

- Max-Min inferencia

$$\mu(u) = \max_{k=1\ldots n}\left\{\min_{\substack{i=1\ldots m\\ j=1\ldots r}}\left[\mu_{i,k}(x_1), \mu_{j,k}(x_2)\right]\right\} \tag{2.8}$$

  k – index funkcií príslušnosti pre výstup u

  i – index funkcií príslušnosti pre vstup x_1

  j – index funkcií príslušnosti pre vstup x_2

- Ťažisková defuzzifikácia

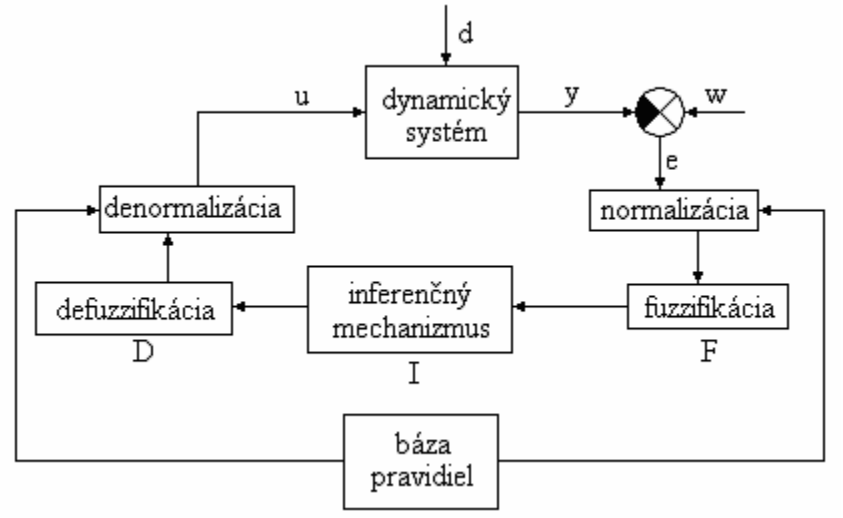
Fuzzy regulátor všeobecného typu

- typ, vstupy, výstupy a báza pravidiel sa navrhujú na mieru danej aplikácie
- ladenie a tvorba štruktúry prebieha experimentálne (intuitívne) na základe pozorovania správania sa riadeného objektu
- napodobňuje sa riadenie človekom (alebo iným objektom)

Charakteristickým znakom fuzzy riadenia je možnosť bezprostredného použitia empirických znalostí človeka – operátora v riadenom procese, ktoré označujeme ako báza znalostí. Bázu znalostí tvoria:

a) informácie o stacionárnych stavoch, intervaloch, v ktorých sa pohybujú hodnoty vstupných a výstupných veličín, ich medzné hodnoty, atď. Ak rozšírime tieto dáta o funkcie príslušnosti všetkých vstupných a výstupných fuzzy množín (ako bude vysvetlené neskôr), potom všetky tieto informácie o procese sa v báze znalostí označujú ako bázy dát.

b) Kvantitatívne formulované skúsenosti aj so slovne definovanými stratégiami riadenie, pomocou ktorých je možno realizovať riadenie, čiže generovať akčnú veličinu. Takto skúsenosťami získané stratégie riadenia označujeme ako bázu pravidiel [1].

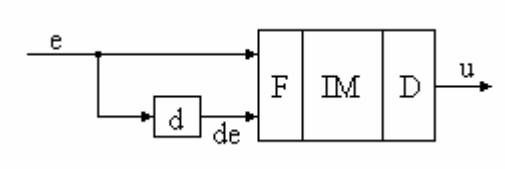


Obr. 2.14 Štruktúra fuzzy regulátora

Štruktúra fuzzy regulátorov je na obr. 2.14 a jeho ústredný člen tvoria tri základné bloky: fuzzifikácia F, interferencia I a blok defuzzifikácie D.

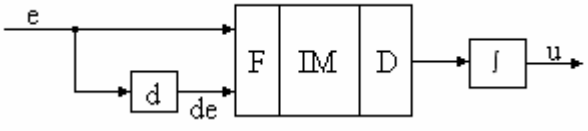
Fuzzy – PID regulátory [11]

Fuzzy PD polohový



If e is A and de is B then u is C

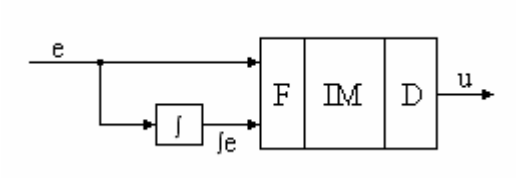
Fuzzy PI rýchlostný



If e is A and de is B then du is C

$u(t) = \int du(t)dt$ resp. $u(k) = u(k-1) + du(k)$

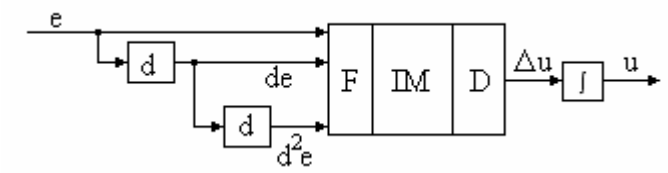
Fuzzy PI polohový



If e is A and ie is B then u is C

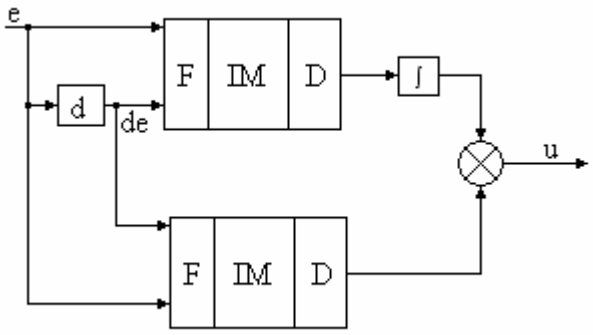
$\int e = \int e(t)dt$ resp. $\sum e$

Fuzzy PID regulátor s 3-D bázou pravidiel



If e is A and de is B and d^2e is C then Δu is D

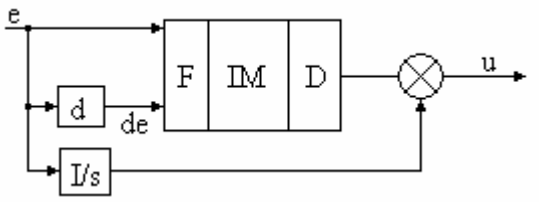
Fuzzy PI + fuzzy PD regulátor



If e is A and de is B then u_1 is E

If e is C end de is D then u_2 is F

Fuzzy PD + klasický I regulátor



Návrh fuzzy – PID regulátorov [11]

Postup návrhu fuzzy regulátora:

1. definovanie vstupných a výstupných lingvistických premenných
2. definovanie univerzálnych lingvistických premenných
3. tvorba bázy pravidiel
4. vyladenie všetkých parametrov (bázy pravidiel a funkcií príslušnosti)

Vplyv zmeny tvaru fuzzy množín:

- zhustenie funkcií príslušnosti e okolo 0 obyčajne zrýchľuje regulačný proces
- zhustenie funkcií príslušnosti de okolo 0 obyčajne pôsobí opačne
- nakláňanie alebo posúvanie f. p. má lokálny alebo smerový charakter, možno tým dosiahnuť prispôsobenie sa nelinearitám alebo špecifickému spávaniu sa systému

Návrh parametrov fuzzy regulátora:

- vypozorovanie činnosti operátora
- automatizované generovanie FR (genetické algoritmy, umelé neurónové siete ...)
- prepočet z iného existujúceho typu regulátora
- experimentálne – pokus/omyl

Neexistuje univerzálna metóda, platná pre všetky typy aplikácií. Je potrebne uplatniť skúsenosti, expertné znalosti, intuíciu ...

Použitie fuzzy regulátorov má význam:

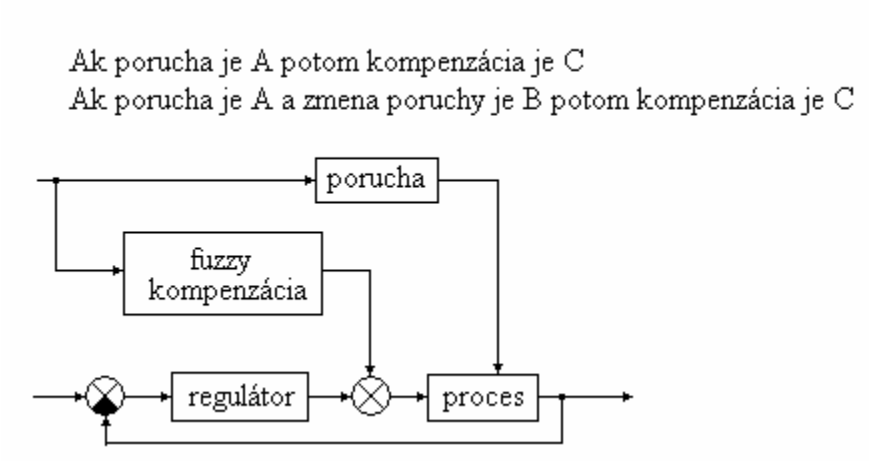
- keď sa nedajú použiť konvenčné (jednoduchšie) algoritmy, alebo kde tieto nevedia zabezpečiť požadované ciele
- keď nepoznáme model objektu riadenia ale jeho správanie vieme opísať lingvisticky
- keď nemáme k dispozícii presné merania ale len približne odhady o stavoch procesu

Nevýhody fuzzy regulátorov:

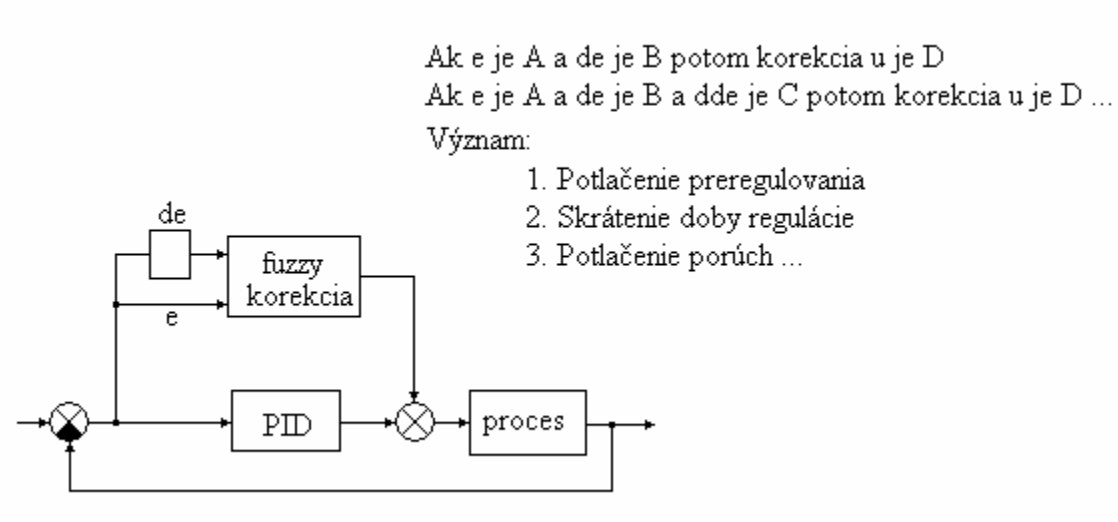
- majú oveľa väčší počet parametrov, ich optimálizácia je zložitý proces
- sú výpočtovo zložitejšie a kladú väčšie nároky na hardware a software
- k ich návrhu nie je k dispozícii univerzálne platná metodika. Sú potrebné skúsenosti s riadením daného procesu a časovo náročné experimentovanie

Fuzzy systém v inej funkcii v regulačnom obvode ako spätnoväzbový regulátor:

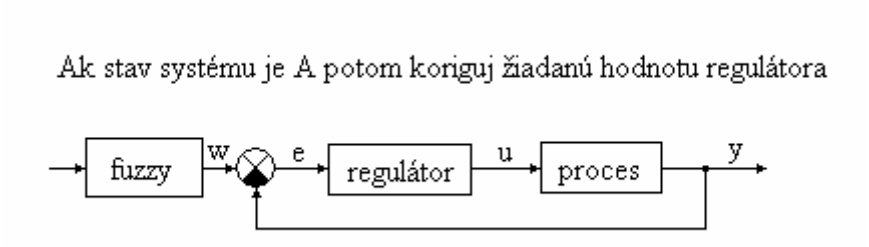
- kompenzácia poruchových veličín (obr. 2.15)
- korekcia akčného zásahu regulátora (obr. 2.16)
- dopredný regulátor (obr. 2.17)
- Iné



Obr. 2.15 Kompenzácia poruchovej veličiny



Obr. 2.16 Korekcia akčného zásahu (klasického) regulátora

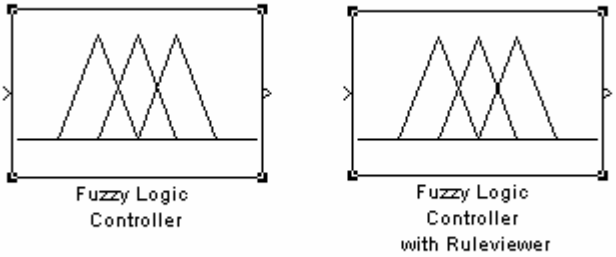


Obr. 2.17 Dopredný regulátor realizovaný fuzzy logikou

2.5 Fuzzy logika a riadenie v MATLAB/SIMULINK ©

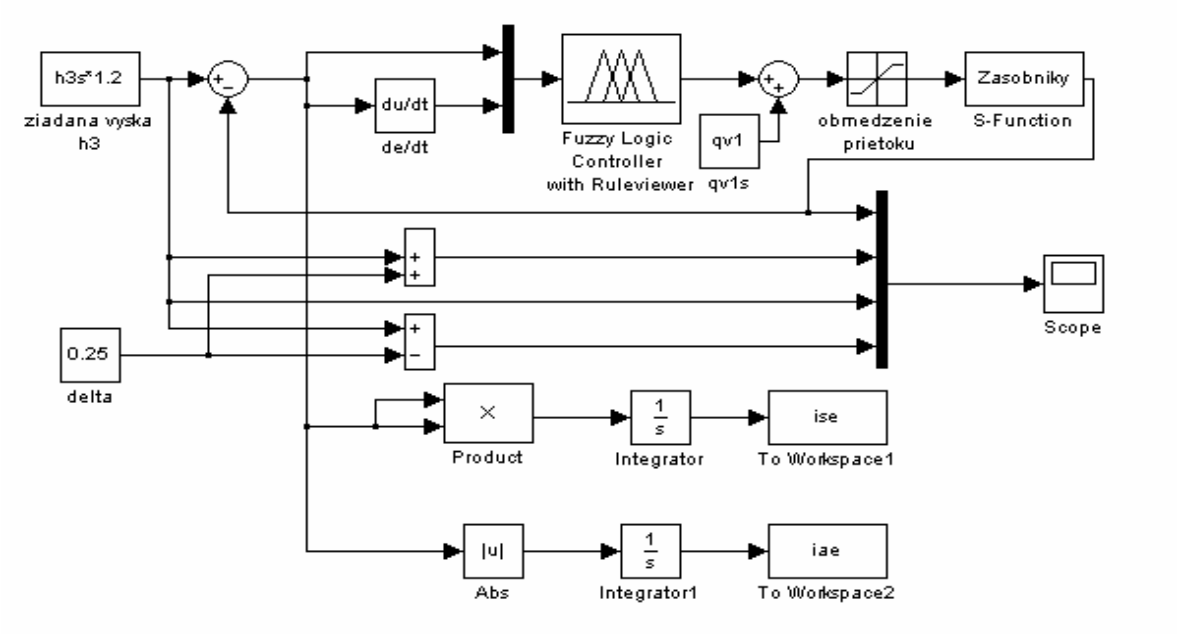
Pre reguláciu v reálnom čase, meranie a simulácie sa požíva prostredie MATLAB/SIMULINK ©, ktorý obsahuje v menu Fuzzy Logic Toolboxu fuzzy regulátory (obr. 2.18):

a) Fuzzy logic controller (fuzzy regulátor)
b) Fuzzy logic controller with Ruleviewer (fuzzy regulátor s grafickým zobrazovaním procesu inferencie)



Obr. 2.18 Ikony fuzzy regulátorov pre MATLAB/SIMULINK ©

Fuzzy regulátor sa v MATLAB/SIMULINK-u © prepojuje bežným spôsobom, viď obr. 2.19. Základnú štruktúru fuzzy regulátorov, ktorú tvoria bloky fuzzifikácie, defuzzifikácie a interferencie, zastupuje v MATLAB/SIMULINK-u © blok „Fuzzy logic controller“ („Fuzzy logic controller with Ruleviewer“).

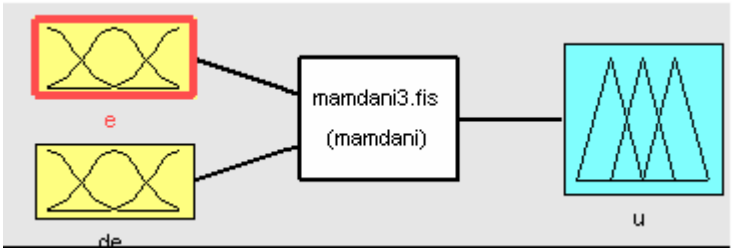


Obr. 2.19 Základná štruktúra spätnoväzbového obvodu s fuzzy regulátorom

Fuzzy regulátor využíva ďalšie možné informácie o procese (báze pravidiel a dát) aj s informáciami a skúsenosťami obsluhy. Obsahom návrhu fuzzy regulátorov je potom získanie týchto informácií a ich využitie v rámci návrhu fuzzy regulátorov.

2.5.1 Návrh štruktúry a vlastností fuzzy regulátorov

Vlastnú návrhársku prácu fuzzy regulátorov je potom možné robiť pomocou interaktívneho grafického prostredia Graphical User Interface (GUI) alebo pomocou príkazového riadku. Projektovanie vyžaduje definovať vstupné a výstupné premenné, ich rozsahy, funkcie príslušnosti a ich parametre, zadávanie inferenčných a rozhodovacích pravidiel, nastavenie metód fuzzifikácie a defuzzifikácie. Pre tieto požiadavky môžeme vyjadriť štruktúru fuzzy regulátora blokovo podľa obr. 2.20. Túto štruktúru budeme označovať ako Fuzzy Inference System FIS – Inferenčný systém fuzzy. Užívateľské grafické prostredie GUI obsahuje nástroje pre vytvorenie, editáciu a zobrazovanie fuzzy inferenčného systému (FIS).



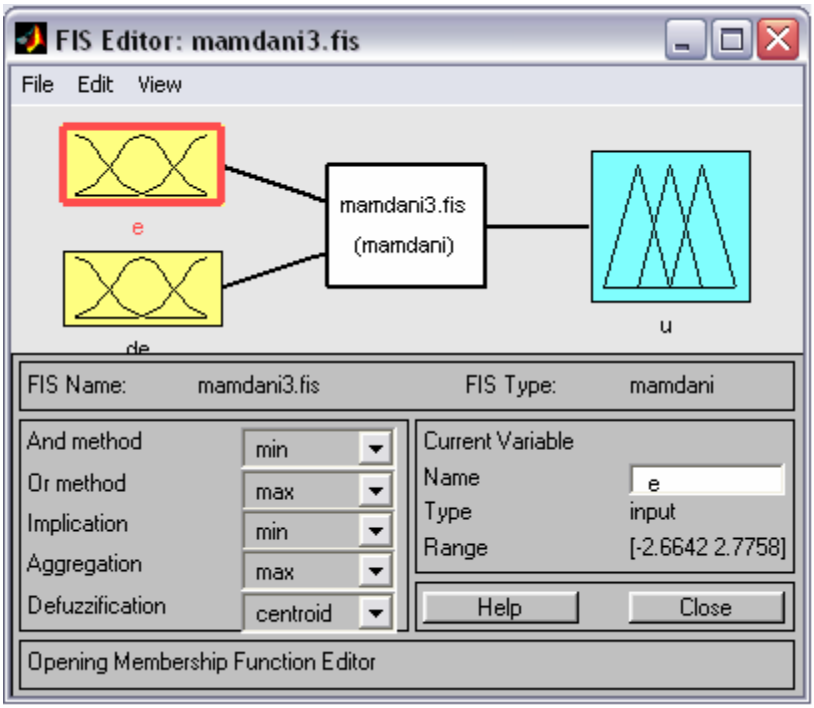
Obr. 2.20 Fuzzy inference Systém FIS – inferenčný systém fuzzy

FIS tvoria 3 editory: FIS Editor (editor inferenčného systému fuzzy regulátorov), viď obr. 2.21, Membership Function Editor (editor funkcií príslušnosti), viď obr. 2.22 a Rule Editor (editor rozhodovacích pravidiel), viď obr. 2.24 a dve zobrazovanie – Rule a Surface Viewer (grafické zobrazovanie procesu inferencie, viď obr. 2.25 a plochy ohraničujúci priestor generovaných akčných zásahov, viď obr. 2.26).

FIS Editor sa pri vytváraní nového fuzzy inferenčného systému aktivizuje príkazom `FUZZY.` Aktivácia už existujúceho fuzzy inferenčného systému sa aktivizuje príkazom `fuzzy meno.fis.` Týmto príkazom sa vyvolá už existujúci fuzzy inferenčný systém definovaný menom súboru.

FIS Editor

Hlavné menu viď obr. 2.21 obsahuje roletové menu File, Edit, View, ktoré umožňujú ukladanie a volanie súborov a editáciu fuzzy systémov pomocou nástrojov GUI. V ponuke Edit je možné pridaním alebo ubraním určiť počet vstupov a výstupov. V grafickom okne sú v princípe zobrazované tromi ikonami – blokmi: vstupné premenné, typ inference FIS, výstupné premenné. Ak sú spojovacie linky medzi blokmi vyznačené čiarkovanou čiarou, potom nie sú jednotlivé bloky parametrizované alebo parametrizácia nie je správne ukončená a tento blok nie je možné zapojiť a spustiť.

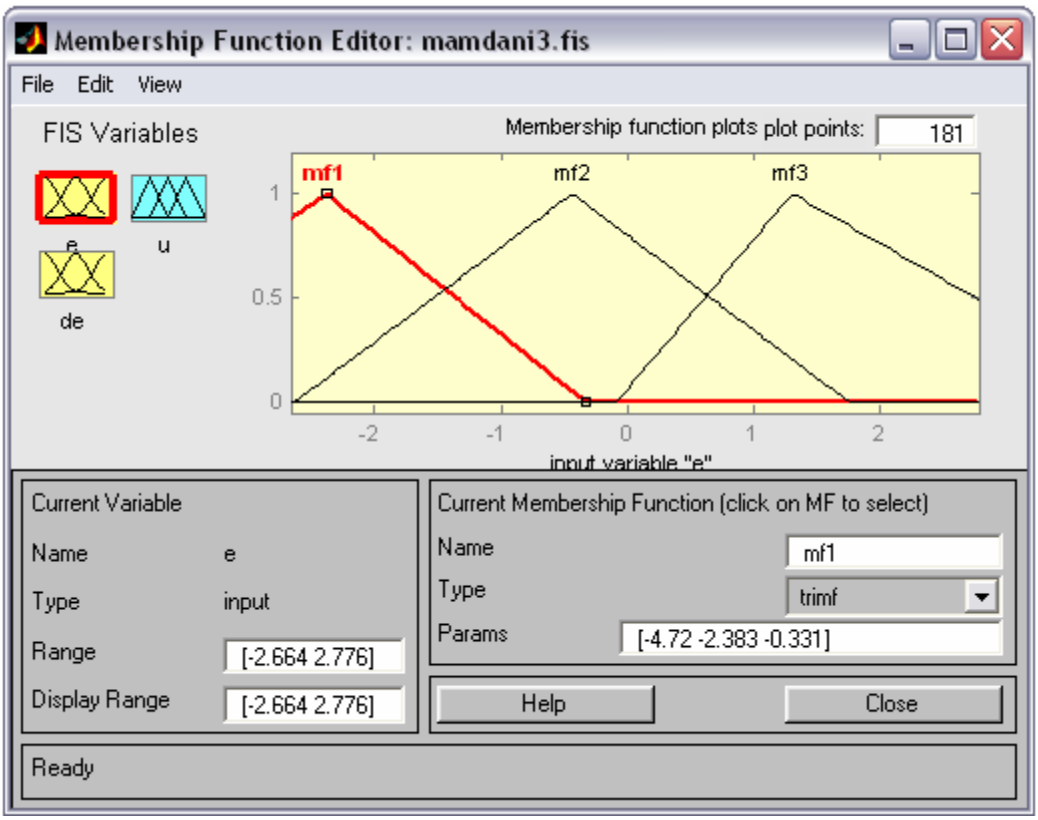


Obr. 2.21 FIS Editor

Dvojitým kliknutím na vybranú vstupnú premennú reprezentovanú obrázkom je možné prejsť do Membership Function Editor (editoru funkcií príslušnosti). Dvojitým kliknutím na typ inferencie reprezentované obrázkom je možné prejsť do Rule Editoru (editoru rozhodovacích pravidiel). Dvojitým kliknutím na vybranú výstupnú premennú reprezentovanú obrázkom je možné prejsť do Membership Function Editor (editoru funkcií príslušnosti). V ľavej časti okna je možné zadávať príslušné parametre metódam AND, OR a Implikácií a parametrov agregačnej defuzzifikačnej metóde. V pravej časti je možné editovať mená vstupných a výstupných premenných. Zobrazené sú tiež rozsahy premenných a typ.

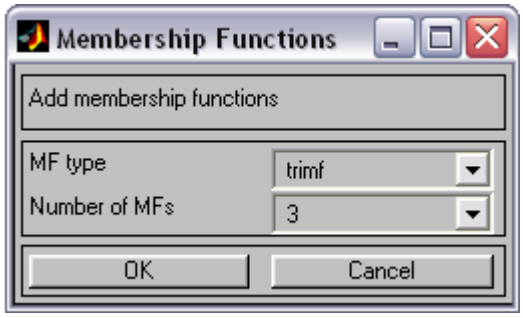
Membership Function Editor

Spustí sa buď vo FIS editoru dvojitým kliknutím na ikonu výstupu alebo vstupu alebo cez roletkové okno Membership Function Editor.



Obr. 2.22 Membership Function Editor

Ak vyberieme na obr. 2.22 vstupnú veličinu kliknutím na jej ikonu v ľavom rohu, vybraná ikona po obvode sčervená a v grafickom okne „Membership function plots“ sa zobrazia všetky jej funkcie príslušnosti s nastavenými parametrami a s menami ich premenných – termov. V okne „Current Variable“ je uvedené meno, typ, rozsah a rozsah displeja označených heslami: Name, Typ, Range, Display Range. V tomto okne môžeme zadávať potrebné rozsahy. Je vhodné najskôr u zvolenej premennej nastaviť jej rozsahy. Potom kliknutím v rolovaciom menu Edit na príkaz Add FMs sa zobrazí okno „Add membership functions“ podľa obr. 2.23. V rolovaciom menu hesla MF type sa volí typ funkcie príslušnosti vstupnej alebo výstupnej premennej (lingvistickej) z množiny (trimf, tramf, gbellmf, gausmf, gaus2mf, pimf, dsigmf, psigmf).

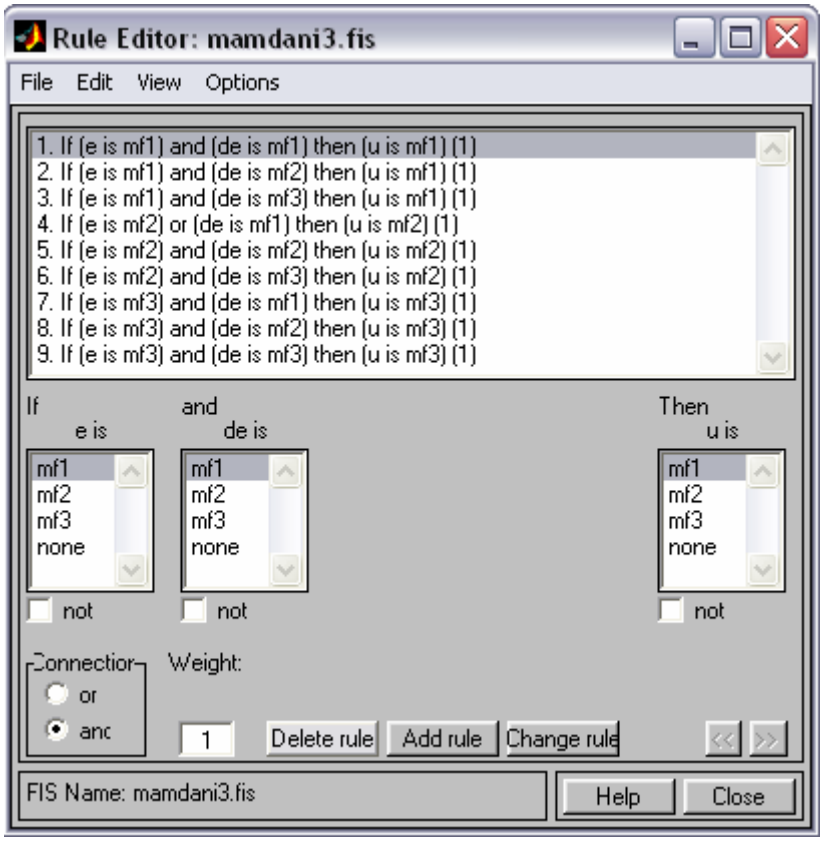


Obr. 2.23 Okno „Add membership functions“

V rolovaciom menu hesla Number of MFs sa volí počet termov – hodnôt (lingvistických) vstupných alebo výstupných premenných. Ak klikneme na jednu z funkcií príslušnosti, zmení farbu na červenú a je možné ju zmeniť aj jej meno, tvar či číselné parametre. Tieto zmeny prevedieme v dielčom okne „Current Membership Function“. Meno, typ a parametre vybranej funkcie príslušnosti sú zobrazené v poliach hesiel: Name, Typ, Params, na ktorých je tiež možné prepísaním mena a novým nastavením typu a parametrov funkcie príslušnosti urobiť požadované zmeny.

Rule Editor

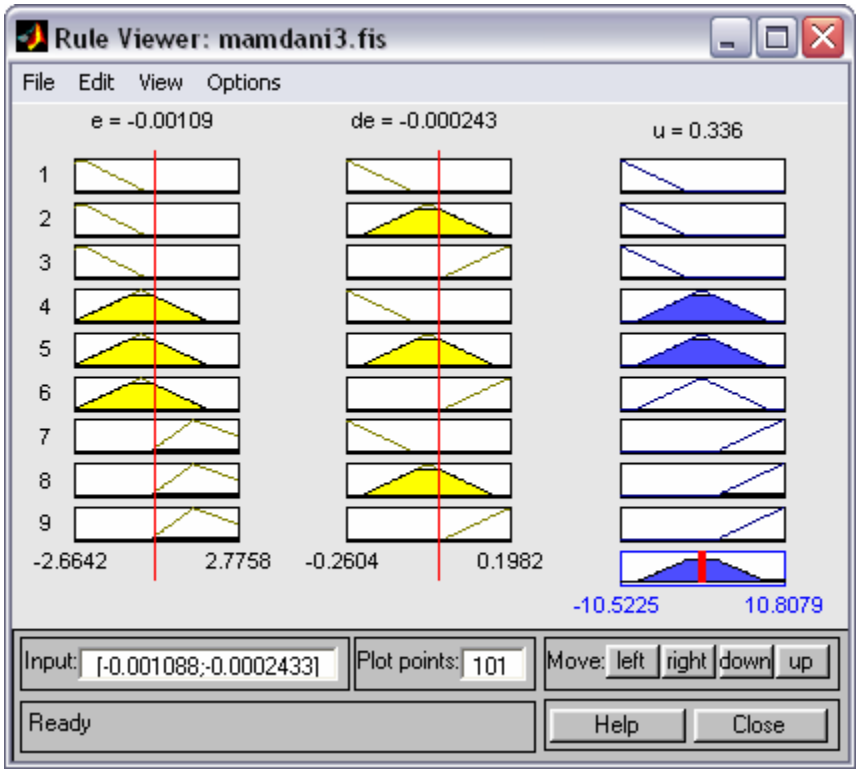
Spustí sa vo FIS editoru cez roletové okno View-Edit Rules. Obsahuje editačné a zobrazovacie pole, viď obr. 2.24. V tomto poli je možné priamo editovať ručne alebo použitím tlačítok: Delete rule – maže pravidlo, Add rule – pridáva pravidlo, Change rule – mení pravidlo. Vlastné pravidlo je možné zostaviť pomocou roletových menu vstupných a výstupných lingvistických premenných (e(t), de(t) a u(t)), aj so zadávaním váh. Rule Editor ponúka roletové menu pre vstupné a výstupné premenné, kde každú položku tvorí meno lingvistickej premennej – termy. Termy sa dajú spojovať operátormi AND alebo OR. Jednotlivé termy môžu vystupovať v rozhodovacích pravidlách i v negácii, čo prevedieme na príslušný operátor NOT.



Obr. 2.24 Rules Editor

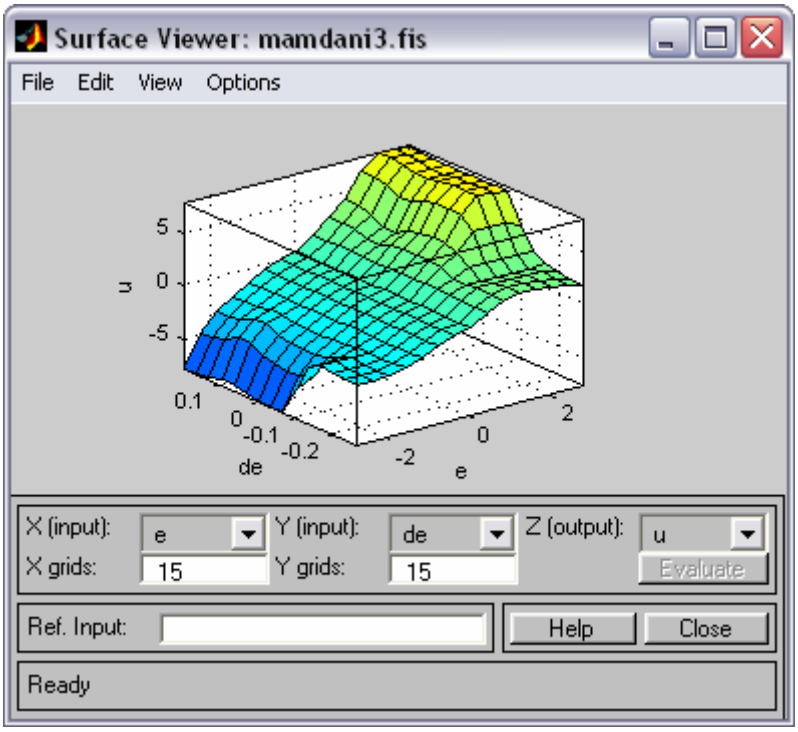
Rule Viewer, Surface Viewer

Rule Viewer (grafické zobrazovanie procesu inferencie), viď obr. 2.25 sa aktivizuje pomocou roletových menu View výberom Rule Viewer. Obsahuje ako všetky pravidlá, tak aj tvary funkcií príslušnosti vstupov a výstupov a ich inferencií.



Obr. 2.25 Rule Viewer

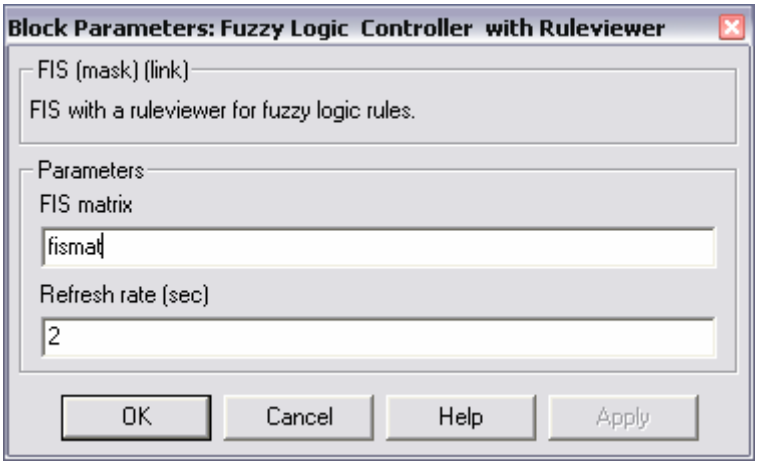
Surface Viewer (grafické zobrazovanie procesu inferencie), obr. 2.26, sa aktivizuje pomocou roletových menu View výberom Surface Viewer. Zobrazuje priestor hodnôt výstupnej veličiny v závislosti na vstupných premenných. Pre fuzzy reguláciu je uvažovaná spravidla regulačná odchýlka a jej derivácie.



Obr. 2.26 Surface Viewer

FIS Matice

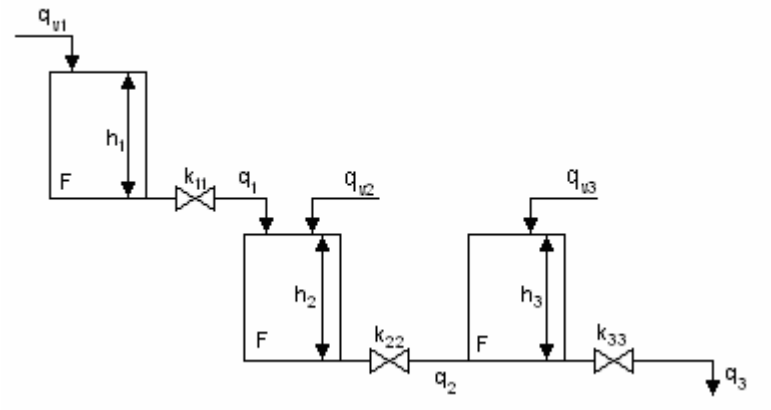
Aby bolo možné spustiť simuláciu v simulinku, je treba Fuzzy Inference System uložiť pod jeho menom do pracovného prostredia Matlabu (workspace). Informácie uložené vo Fuzzy Inference System sú uložené ako matice. Táto matica je označovaná ako FIS Matrix (FIS Matica). Uloženie do pracovného prostredia sa prevedie na hlavnej lište v rolovacom menu File príkazom Save to workspace. Dvojitým kliknutím na ikonu fuzzy regulátora v MATLAB/SIMULINK-u © sa zobrazí okno „Block Parameters: Fuzzy Logic Controller“ a je možné zadať aktuálne meno FIS – Matice, viď obr. 2.27. Znamená to teda, že pre každú voľbu bázy dát a rozhodovacích pravidiel sa vytvorí odpovedajúca FIS Matica. MATLAB/SIMULINK © umožňuje vybranému fuzzy regulátoru v schéme priradiť zvolenú FIS maticu, čím sú vlastnosti regulátora definované. V danej programovej schéme sa dá potom už iba meniť váhy na vstupu a výstupu [1].



Obr. 2.27 Okno „Block Parameters: Fuzzy Logic Controller“

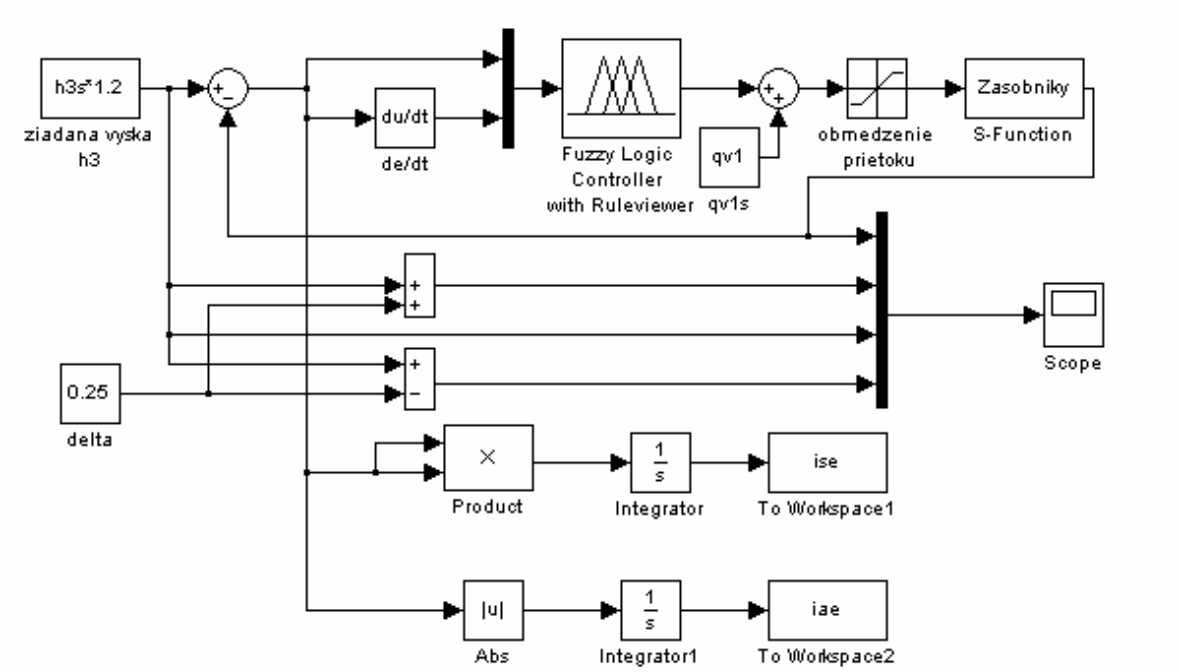
2.5.2 Riadenie troch zásobníkov kvapaliny s interakciou medzi 2. a 3.

Máme systém troch zásobníkov kvapaliny s interakciou medzi druhým a tretím ako je zobrazené na obr. 2.28.



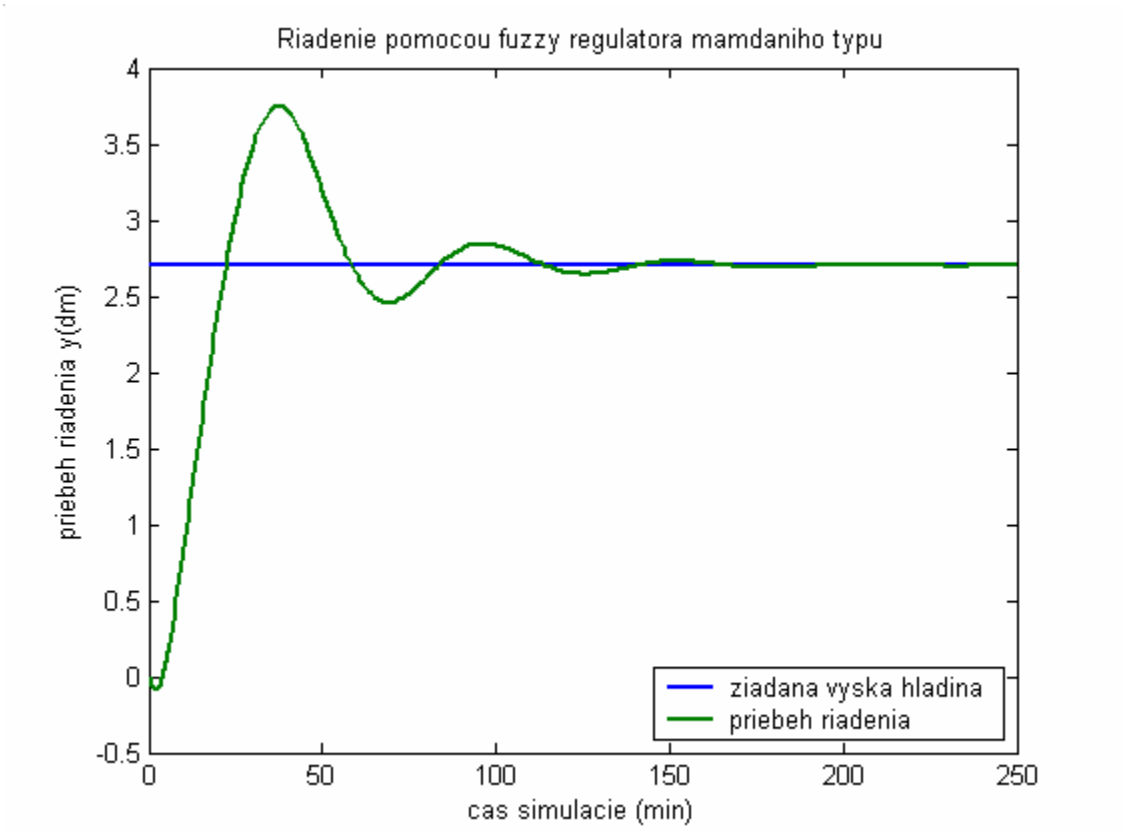
Obr. 2.28 Zásobníky kvapaliny

Riadime fuzzy regulátorom Mandaniho typu s tromi trojuholníkovými funkciami príslušnosti pre každý vstup a výstup podľa nasledujúcej schémy, ktorá je zobrazená na obr. 2.29. Súbor **fuzzy.m** vytvorený v prostredí MATLAB © na riadenie zásobníkov je uvedený v prílohe 1.



Obr. 2.29 Schéma na riadenie zásobníkov kvapaliny s fuzzy regulátorom

Priebeh riadenia fuzzy regulátorom Mamdaniho typu s 3 funkciami príslušností je zobrazený na obr. 2.30.



Obr. 2.30 Priebeh riadenia fuzzy regulátorom

Fuzzy príkazy použité na riadenie zásobníkov kvapaliny v súbore fuzzy.m

- newfis – vytvorenie fis matice (názov fis matice) pr. `p=newfis('mamdani3.fis')` uložené do premennej p
- addvar – pridaj vstup alebo výstup (názov premennej, vstup (výstup), názov vstupu (výstupu), vstupný rozsah) pr. `p=addvar(p,'input','e',[-2.6642 2.7758]);`
- addmf – pridaj funkciu príslušnosti (názov premennej, pre vstup (výstup), číslo vstupu (výstupu), názov funkcie príslušnosti, typ funkcie príslušnosti, parametre funkcie príslušnosti) pr. `p=addmf(p,'input',1,'mf1','trimf',[-4.72 -2.38336507936508 -0.331]);`
- plotmf – vykreslenie vstupov alebo výstupov (názov premennej, vstup (výstup), číslo vstupu (výstupu)) pr. `plotmf(p,'input',1)`

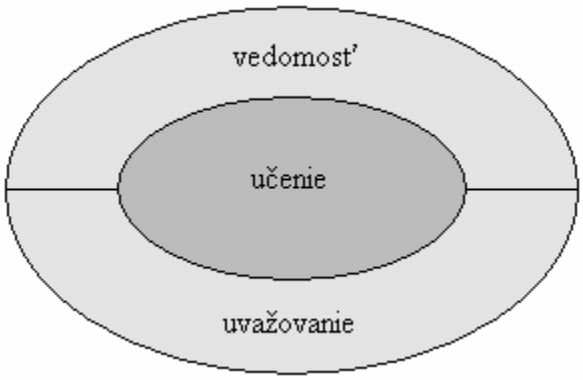
- rulelist – zadanie bázy pravidiel (vstupy | výstupy | váha | spojenie(1 = and, 2 = or)) pr.
```
rulelist=[1 1 1 1 1;1 2 1 1 1;1 3 1 1 1;
          2 1 2 1 2;2 2 2 1 1;2 3 2 1 1;
          3 1 3 1 1;3 2 3 1 1;3 3 3 1 1];
```
- addrule – pridanie bázy pravidiel (názov premennej, názov bázy pravidiel) pr. `fismat=addrule(p,rulelist)`
- showrule – výpis bázy pravidiel (názov premennej) pr. `showrule(p)`
- showfis – výpis parametrov fis matice (názov premennej) pr. `showfis(p)`

3. Umelé neurónové siete

Nový pohľad na paralelné výpočty a na paralelné systémy v súčasnosti prináša najnovší trend, ktorý zaujal mnohých vedcov a odborníkov, a síce quasiencefalické výpočty, v ktorých ide o masívny paralelizmus a majú za cieľ modelovať chovanie sa nervovej sústavy u živočíchov a hlavne modelovať chovanie sa ľudského mozgu. Ľudský mozog obsahuje 10^{11} – 10^{14} neurónov uložených v šedej kôre mozgovej a synapsie sú realizované v rozsahu asi 10^4 na jeden neurón. Vytvorenie umelého ľudského mozgu so všetkými jeho schopnosťami je vec veľmi ťažko riešiteľná aj z hľadiska kvantity neurónov aj z hľadiska spôsobu ich prepojenia. Je tu však možnosť simulovať aspoň niektoré funkcie ľudského myslenia a implementovať ich. Novovytvárané modely sa o túto situáciu pokúšajú, a preto dostali názov neurónové siete.

Každý systém požadovaný za systém umelej inteligencie, by mal spĺňať 3 nasledujúce požiadavky:

- vedieť uložiť znalosti
- aplikovať znalosti pre riešenie problému – uvažovanie
- získavať nové znalosti počas experimentov – učenie

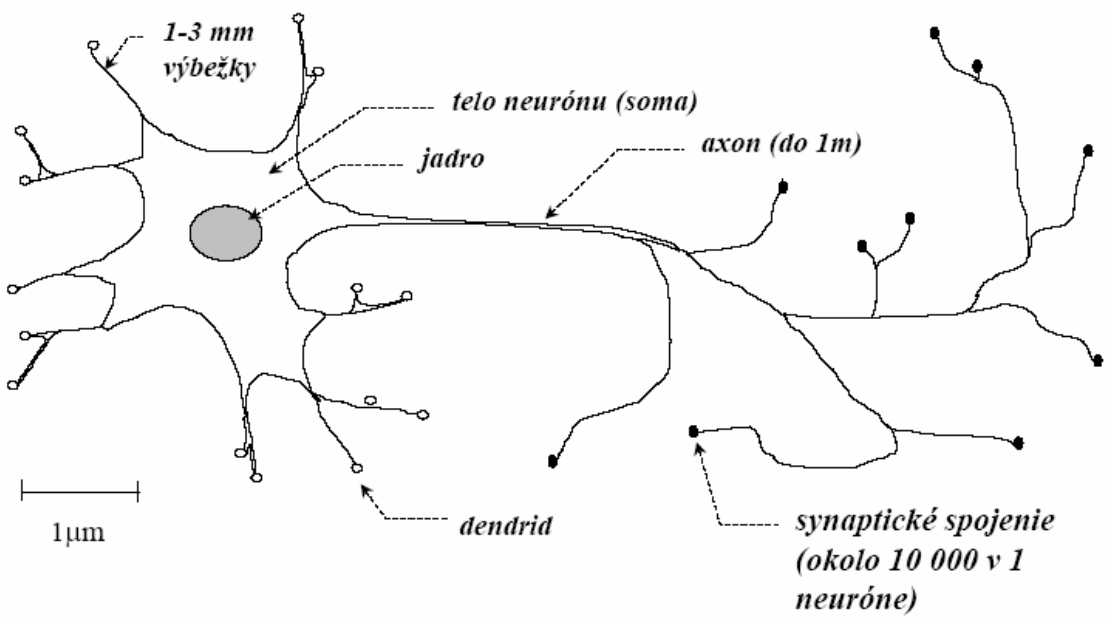


Obr. 3.1 Hlavné komponenty všeobecného systému umelej inteligencie

3.1 Čo je to neurónová sieť

Neurónová sieť je masívne paralelný procesor, ktorý má sklon k uchovaniu experimentálnych znalostí a ich ďalšieho využívania. Napodobňuje ľudský mozog v dvoch aspektoch:

- poznatky sú zbierané v neurónových sieťach počas učenia,
- medzineurónové spojenia (synaptické váhy) sú využívané na ukladanie znalostí.



obr. 3.2 biologický neurón

Je zrejmé, že inšpirácia ku vzniku neurónových sietí prišla z biologických systémov. Hrubo povedané ide o simuláciu mozgu. Na prvý dojem vysoko abstraktná disciplína nachádza množstvo aplikácii v praxi a stáva sa prostriedkom pre riešenie problémov v širokom spektre odborných oblastí. Jednou z veľmi významných vlastností neurónových sietí je, že svojím spôsobom je tzv. univerzálnym aproximátorom funkcií. Môže sa nám stať, že máme systém, ktorého popis je mimoriadne náročný alebo je systém natoľko zložitý, že jeho popis je skoro nemožný. Máme však dáta, ktoré do systému vstupujú, a k nim odpovedajúce výstupy. V takejto situácii, môžeme použiť vhodnú neurónovú sieť a pokúsiť sa naučiť ju chovať sa ako sledovaný systém pomocou

trénovacích údajov (spomínaných vstupov a výstupov). Toto je veľmi dôležitý moment, ktorý determinuje aj aplikačné uplatnenie neurónových sietí v praxi.

Pri štúdiu neurónových sietí môžeme rozlišovať tri oblasti:

- **Teória neurónových sietí** – matematický rozbor činností neurónových sietí, problémy neurónových sietí ako dynamického systému vo všeobecnosti, teoretické rozbory návrhu topológie neurónových sietí a pod. Je treba upozorniť, že matematický model pre popis chovania sa neurónových sietí je dosť náročný.
- **Simulácia neurónových sietí** – ide o simuláciu neurónových sietí pomocou počítačových systémov.
- **Implementácia neurónových sietí** – ide o implementáciu naučenej neurónovej siete do hardwarovej formy. Takéto systémy existujú a objavujú sa čoraz častejšie mikročipy neurónových sietí.

Všetky tri oblasti úzko súvisia, ale v súčasnej situácii možnosti využitia výpočtovej techniky prežívajú plný rozvoj.

Na záver tejto časti je treba zdôrazniť dôležitosť paralelizmu systémov neurónových sietí. Základným elementom neurónových sietí je neurón. V porovnaní s ľudským neurónom vieme pomocou počítačov nasimulovať omnoho rýchlejší neurón ako je neurón ľudský. Problémom je však množstvo ľudských neurónov a množstvo spojení v mozgu. Podľa výskumov má ľudský mozog okolo 10^{11} až 10^{14} neurónov a počet prepojení na každý jeden neurón predstavuje počet 10^3 až 10^4. Takúto masívne paralelnú neurónovú sieť nemáme ešte dlhší čas šancu nasimulovať. Takže paradoxne, aj keď vieme dosiahnuť rýchlejší procesný element – neurón, nevieme dosiahnuť taký masívny paralelizmus, ktorý v konečnom dôsledku určuje silu celej neurónovej siete [10].

3.2 Stručný prehľad histórie neurónových sietí

- 1943 začína éra teórie neurónových sietí pod vedením amerických vedcov McCullocha a Pittsa. Dr. McCulloch bol psychiater a neuroanatóm, kým dr. Pitts bol matematik a celá aktivita, bola sústredená na Univerzite v Chicagu, kde v spomínanom roku 1943 títo dvaja páni prvýkrát definovali binárny neurón. Je

zaujímavé, že John von Neumann, pri konštrukcii svojho prvého počítača ENIAC v roku 1946 bol do určitej miery inšpirovaný aj spomínanou prácou.

- 1948 Wiener vo svojej knihe Kybernetika naznačuje určité koncepty neurónových sietí.
- 1949 Hebb vo svojej knihe The Organization of Behavior (Organizácia správania) prvýkrát explicitne spomína pojem učenie a jeho vzťah k synaptickým váham a ich modifikácii.
- 1952 dr. Ashby napísal knihu Design of a Brian: The Oigin of Adaptive Behavior (Konštrukcia mozgu: pôvod adaptívneho správania). Táto publikácia mala zásadný význam pre rozvoj neurónových sietí.
- 1954 dr. Minsky napísal svoju Ph.D. dizertáciu na tému Neurónové siete a neskôr v roku 1961 napísal zásadný článok Step Toward Artificial Intelligence.
- 1956 páni Rochester, Holland, Habit a Duda sa prvýkrát pokúsili o počítačovú simuláciu neurónových sietí.
- 1958 prichádza dr. Rosenblat s novým prístupom k rozpoznávaniu pomocou tzv. perceptrónu a neskôr prichádza so svojou konvergenčnou teóriou perceptrónu, ktorá predstavuje počiatky neurodynamiky.
- 1960 prichádza s Widrow a Hoff a tzv. Adaline – Adaptive linear element a o 2 roky nato prichádza Widrow s tzv. Madaline – Multiple adaptive element. Tieto príspevky do značnej miery posunuli teoretickú bázu neurónových sietí dopredu, hoci nedostatok výpočtovej techniky vytváral ohromné zábrany ďalšiemu rozvoju.
- v roku 1965 mala dôležitý význam publikácia dr. Nilsona Learning Machines (Učiace sa stroje).
- v roku 1967 dr. Cowan predstavuje svoju „sigmoidálnu“ aktivačnú funkciu
- v roku 1968 d. Gossberg predstavuje svoj adaptívny model neurónu a používa nelineárne diferenciálne rovnice na jeho popis so zámerom ich použitia pre tzv. short term memory (krátko trvajúce pamäte).
- v roku 1969 dr. Minsky a d. Papert popisujú činnosť viacvrstvového perceptrónu.
- obdobie 1970-80 nazývame obdobím útlmu (Decade of Dormacy). Dôvodom, boli nedostatočné výpočtové kapacity vrátane pamäťových možností. Určité práce v teoretickej oblasti boli urobené ale nie s takou dynamikou ako predtým.

- 1975 d. Little a Shaw popisujú pravdepodobnostný model neurónu.
- v roku 1980 prichádza Grossberg s rozvojom tzv. Competitive learning, ktorá po rozpracovaní a modifikácii zakladá novú triedu neurónových sietí založenej na tzv. Adaptive resonance theory.
- v roku 1982 dr. Hopfield použil termín energie neurónových sietí pre pochopenie rekurentných sietí. Postupne rozvojom vzniká aj trieda tzv. Hopfieldových sietí. O rok neskôr v podstate formuloval princípy simulácie pamäte resp. uchovania informácie v dynamickom systéme.
- dr. Kirkpatrick a jeho kolegovia popisujú procedúru tzv. simulovaného ochladzovania (simulated annealing). Toto inšpirovalo v roku 1985 dr. Hintona a kol. k návrhu stochastickej učiacej procedúry pre tzv. Boltzmanov stroj.
- v tom istom roku (1983) prišli páni Barto a kol. s tzv. reinforcement learning a jeho aplikáciou v oblasti riadenia technologických systémov.
- v roku 1986 prišli dr. Rumelhart a kol. s metódou učenia spätným šírením chyby (Backpropagation of Error). Táto metóda pre svoju relatívnu jednoduchosť je jednou z najrozšírenejších metód učenia neurónových sietí.
- v roku 1988 prišli páni Broomhead Lowe s procedúrou Radial Basis Functions, pre dopredné siete, ktorá má korene v teórii potenciálnych funkcií, ktoré využili Duda a Hart v roku 1973 pre rozpoznávanie [10].

3.3 Typy úloh riešiteľných pomocou neurónových sietí

Oblasti využitia neurónových sietí:

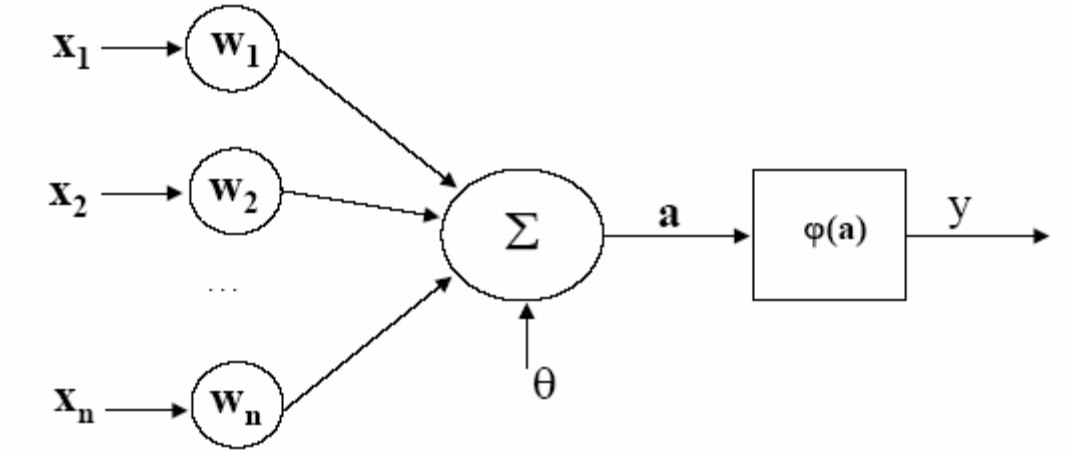
- problémy aproximácie funkcií
- klasifikácia do tried, klasifikácia situácií
- riešenie predikčných problémov
- problémy riadenia procesov
- transformácia signálov
- asociačné problémy, simulácia pamäte

3.4 Základné pojmy neurónových sietí

Neurónová sieť je masívne paralelný procesor. Skladá sa z neurónov (jednotiek) a prepojení (hrán).

Prepojenie je orientovaná hrana, ktorá je ohodnotená váhou w_{ij}, kde i je neurón z ktorého hrana vychádza a j je neurón do ktorého hrana smeruje.

Neurón je základným prvkom neurónovej siete. Realizuje matematické operácie, ktoré v neurónovej sieti prebiehajú [9].

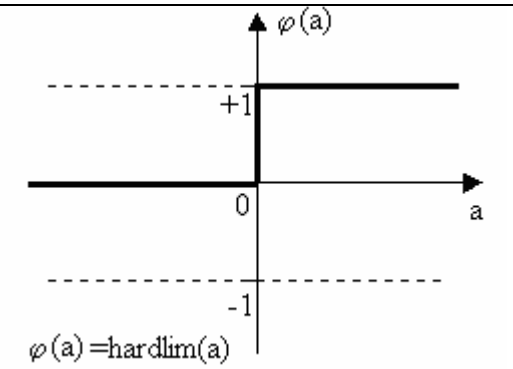
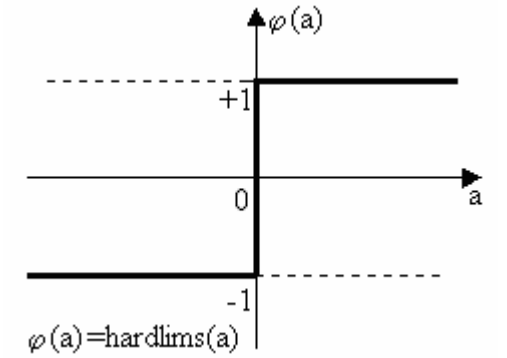
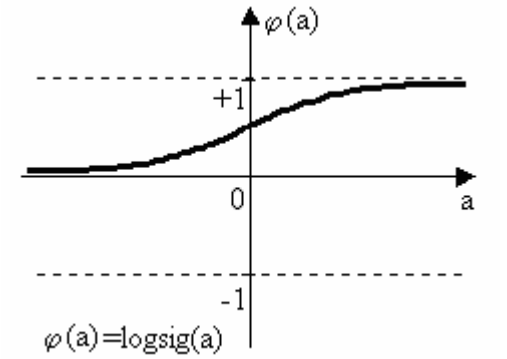
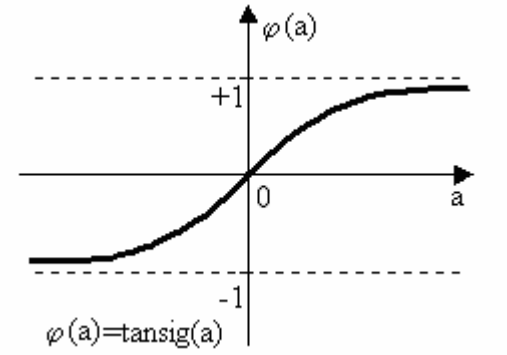


Obr. 3.3 Neurón

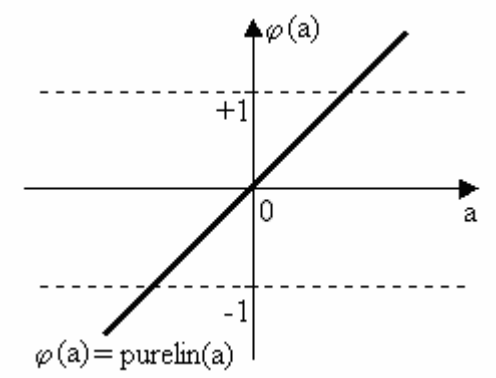
Skladá sa z nasledujúcich častí:

- x_i - vstupy neurónu
- w_i - váhy synaptických spojení
- θ_i - prah neurónu, je externým vstupom do neurónu
- a - vnútorná aktivita neurónu
- φ - aktivačná funkcia neurónu
- y - výstup neurónu

Typy aktivačných funkcií a príkaz na použitie tejto funkcie v neurónových sieťach v Matlabe:

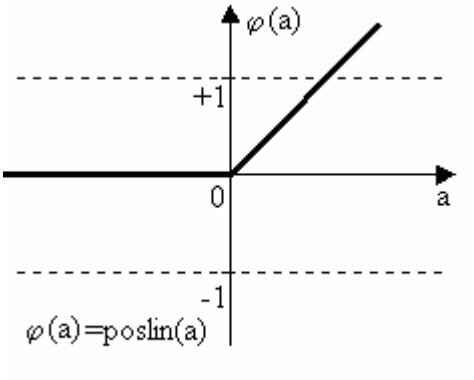
• **silno obmedzená aktivačná funkcia** – obmedzuje výstupné hodnoty neurónu na dve výstupné hodnoty. $\varphi(a)=\begin{cases}-1 & pre \quad a<0\\ 1 & pre \quad a\geq 0\end{cases}$	
• **symetrická silno obmedzená aktivačná funkcia** – obmedzuje výstupné hodnoty neurónu na dve výstupné hodnoty. $\varphi(a)=\begin{cases}-1 & pre \quad a<0\\ 1 & pre \quad a\geq 0\end{cases}$	
• **logsigmoidálna aktivačná funkcia** – veľmi často sa využíva v mnohovrstvových perceptrónových sieťach, a to z dôvodu, že funkcia je diferencovateľná. $\varphi(a)=\frac{1}{1+e^{-\beta a}}$	
• **tansigmoidálna aktivačná funkcia** – na rozdiel od logsigmoidálnej funkcie má rozsah výstupných hodnôt $\langle -1,1\rangle$. $\varphi(a)=tgh\left(\frac{a}{2}\right)=\frac{1-e^{-a}}{1+e^{-a}}$	

- **lineárna aktivačná funkcia** – má lineárny vzťah medzi vstupnou a výstupnou hodnotou neurónu. Používa sa väčšinou v poslednej vrstve neurónovej siete. $\varphi(a) = a$



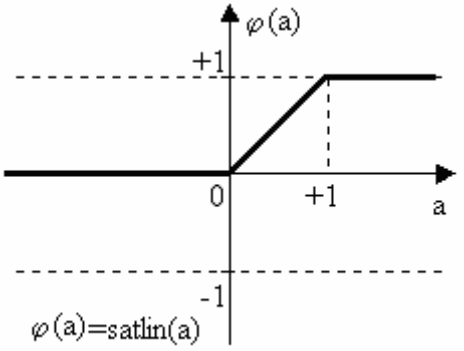
- **pozitívne lineárna aktivačná funkcia** – je obdobou lineárnej funkcie, ale výstupné hodnoty neurónu môžu nadobúdať iba kladných hodnôt.

$$\varphi(a) = \begin{cases} a & pre \quad a \geq 0 \\ 0 & pre \quad a < 0 \end{cases}$$



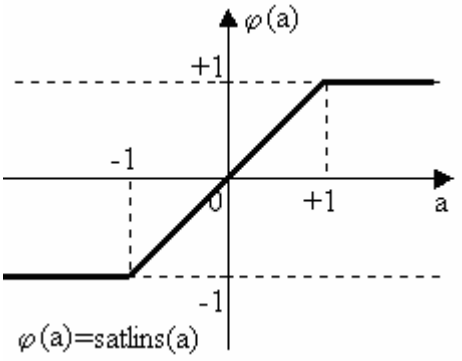
- **saturačná lineárna aktivačná funkcia** – obmedzuje veľkosť vstupného signálu v rozsahu

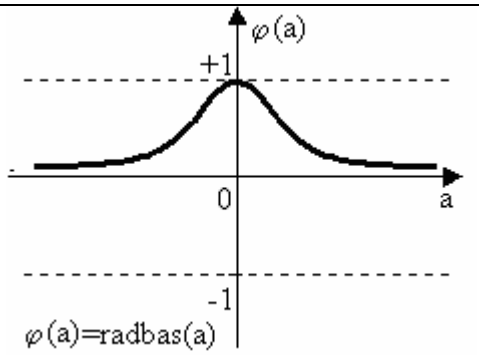
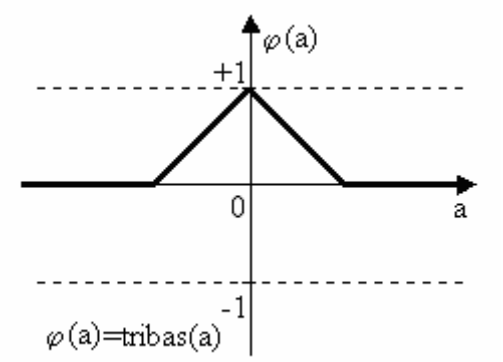
$$\langle 0,1 \rangle . \varphi(a) = \begin{cases} 0 & pre \quad a < 0 \\ a & pre \quad a \in \langle 0,1 \rangle \\ 1 & pre \quad a > 1 \end{cases}$$



- **symetrická saturačná lineárna aktivačná funkcia** – obdoba saturačnej lineárnej aktivačnej funkcie.

$$\varphi(a) = \begin{cases} 0 & pre \quad a < -1 \\ a & pre \quad a \in \langle -1,1 \rangle \\ 1 & pre \quad a > 1 \end{cases}$$

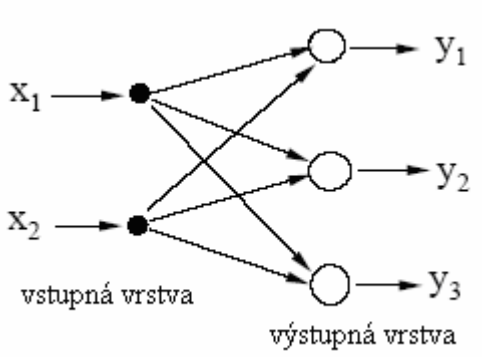


• **aktivačná funkcia radiálneho základu** – je určená pre radiálne neurónové siete.	
• **aktivačná funkcia trojuholníkového základu** – prevedie zhodnotenie vstupných hodnôt len v rozsahu $\langle -1,1 \rangle$.	

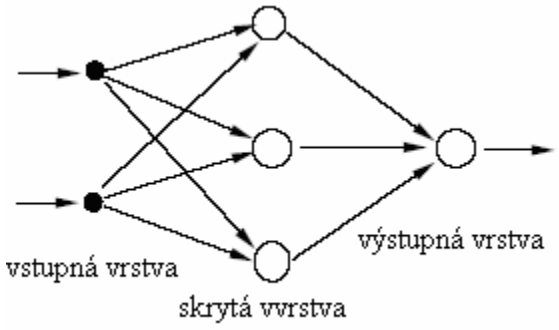
3.4.1 Rozdelenie neurónových sietí podľa architektúry:

Vrstvové štruktúry [11]

- **Jednovrstvové (resp. dvojvrstvové) siete** – úplné prepojenie, obyčajne perceptrónové siete so skokovými aktivačnými funkciami za účelom klasifikácie do tried (napr. sieť štruktúry 2-3 na obr.3.4)
- **Viacvrstvové siete** – obsahujú aspoň jednu skrytú vrstvu. Na obr. 3.5 je trojvrstvová sieť štruktúry 2-3-1. Vstupná vrstva iba distribuuje vstupné signály, výkonnú funkciu majú len skrytá a výstupná vrstva.



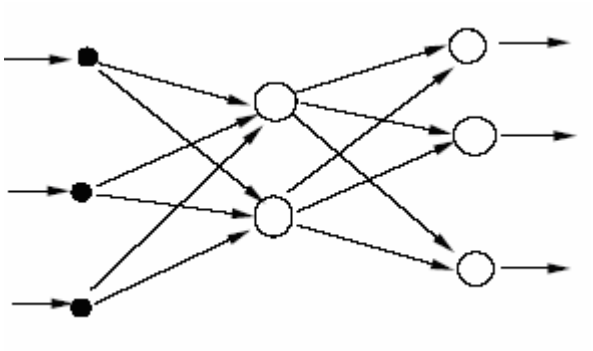
obr. 3.4 Dvojvrstvová sieť



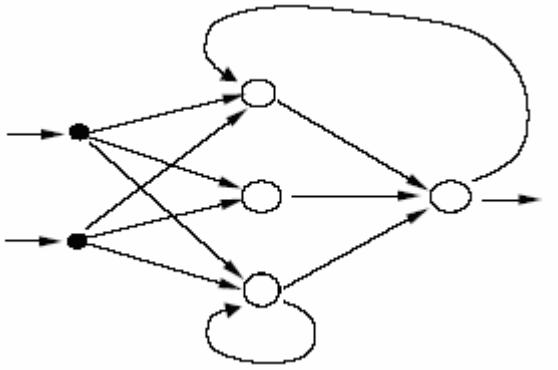
obr . 3.5 Viacvrstvová sieť

Dopredné a rekurentné vrstvové štruktúry [11]**:**

- **Dopredné siete** – obsahujú iba väzby v smere od vstupov k výstupom. Na obr. 3.6 je dopredná sieť veľkosti 3-2-3.
- **Rekurentné** siete – sú podobné ako dopredné, ale obsahujú aspoň jednu spätnú väzbu v rámci tej istej vrstvy alebo v rámci rôznych vrstiev

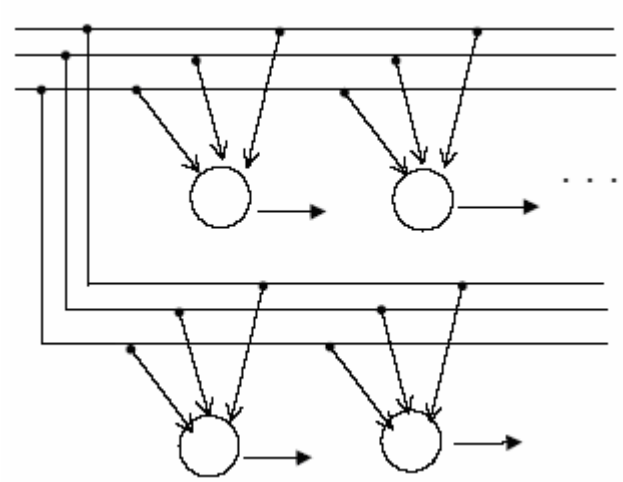


obr. 3.6 Dopredná neurónová sieť

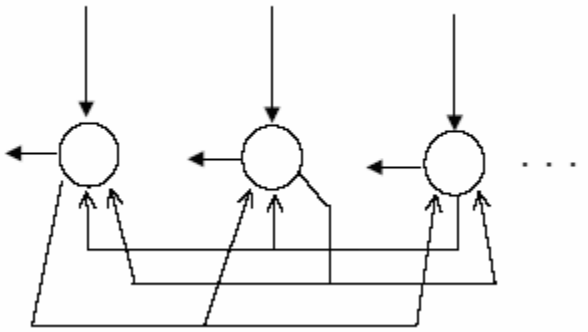


obr. 3.7 Rekurentná neurónová sieť

Iné štruktúry



obr. 3.8 Kohonenova sieť – mriežková štruktúra



obr. 3.9 Hopfieldova sieť

Učenie (trénovanie) neurónových sietí je proces, kedy sa váhy menia. T.j. ak vytvoríme maticu všetkých váh W, potom: $\frac{\partial W}{\partial t} \neq 0$. Zmena váh prebieha na základe nejakých pravidiel, ktoré sú pravidlá učenia [11].

Spôsoby učenia neurónových sietí:

- podľa miery vplyvu (nadradeného) učiteľa
  - *učenie s učiteľom* – danému vstupu zodpovedá známy výstup (odozva). Parametre siete sa koridujú, aby sa správanie siete približovalo reálnym výstupom modelovaného systému (od učiteľa).
  - *učenie známkované* – učiteľ namiesto správnej odpovede iba ohodnotí adekvátnosť správania sa siete známkou.
  - *samoorganizácia* – sieť sa modifikuje sama reakciou na vstupné dáta bez akéhokoľvek učiteľa
- podľa typu učenia
  - *učenie korigujúce chybu* – napr. back-propagation
  - *súťažné učenie*
  - *Hebovo učenie*
  - *iné*
- podľa prístupu k dátam
  - *skupinový prístup (off-line)* – učiace dáta sú pred začiatkom učenia už dané a aktualizácia parametrov neurónovej siete prebieha vo všetkých iteráciách na základe známych požadovaných vstupno-výstupných hodnôt (napr. back-propagation, Newtonová metóda, Gauss-Newtonova metóda a iné).
  - *postupný prístup (on-line)* – učiace dáta sú pri hľadaní optimálnych parametrov neurónovej siete dodávané postupne, v každom iteračnom kroku sú dostupné iba dáta aktuálne v danom časovom okamžiku. Využíva sa v prípade, že učiace dáta sú závislé na aktuálnom dynamickom správaní modelovaného systému, ktorého vlastnosti sa v čase menia (tzv. rekurzívne metódy)

3.5 Modely neurónových sietí

Kombinácia topológie, typu učenia a učiaceho algoritmu určuje model neurónovej siete. Pre účely dolovania údajov sú najčastejšie používanie siete so spätným šírením a Kohonenove mapy [7].

Sieť so spätným šírením – využíva topológiu doprednej siete, učenie s učiteľom a algoritmus spätného šírenia chýb. Tento typ siete je univerzálne použiteľný. Je dostatočne robustný, avšak výpočtovo náročný počas tréningového procesu. Sieť s jednou skrytou vrstvou dokáže modelovať ľubovoľnú spojitú funkciu s požadovanou presnosťou za predpokladu dostatočného počtu skrytých neurónov. Najväčšou výhodou tohto typu siete je, že je jednoduchá na pochopenie aj implementáciu a rieši široké spektrum problémov.

RBF siete (Radial Basis Function) – sú dopredné siete trénované s učiteľom. Typická konfigurácia siete je jedna skrytá vrstva a množina aktivačných funkcií. Napriek podobnosti so sieťami so spätným šírením, majú RBF siete niekoľko výhod. Proces trénovania je väčšinou kratší. Sú menej citlivé na neustálené vstupy. Je to spôsobené správaním sa skrytej vrstvy. Ako aktivačná funkcia je v nich použitá Gaussovská funkcia alebo nejaká iná, jej podobná funkcia. Aktivita neurónu potom vlastne zodpovedá podobnosti vstupnej hodnoty a váh. Výsledkom toho je, že RBF siete sú vynikajúce v rozpoznávaní vzorov.

ART siete (Adaptive Resonance Theory) – ART siete sú rekurentné siete používané najmä vtedy, keď údaje potrebujeme triediť do zhlukov. Podľa predloženého vstupného vzoru sa z výstupnej vrstvy vyberie jeden víťazný element. Obsahujú spätnú väzbu, cez ktorú sa do vstupu zahrnie očakávaný nasledujúci vzor. Ak aktuálny vstupný vzor nezodpovedá očakávanému, víťazný neurón je odpojený a hľadá sa iný, ktorý vstupu zodpovedá najlepšie. Tento proces sa opakuje dovtedy, pokým vstup nezodpovedá (v istej tolerancii) očakávanému vstupu víťazného neurónu. Ak neexistuje víťazný neurón, vytvorí sa nový, ktorý bude očakávať práve aktuálny vzor.

3.5.1 Kľúčové problémy pri výbere modelu siete

Pomocou nasledovných kritérií je výber toho správneho modelu pre danú problémovú oblasť veľmi jednoduchý [7].

- Výber potrebných funkcií
  - vytváranie zhlukov, skupín
  - klasifikácia
  - modelovanie
  - aproximácia funkcií
- zváženie vstupných údajov
  - binárne údaje
  - reálne čísla
- množstvo tréningových údajov

Proces trénovania siete: Počas trénovania siete môže nastať situácia, kedy sieť nebude schopná sa na prvý krát naučiť správne reprezentovať vstupy, resp. nebude spĺňať požiadavky na presnosť. Závažnými sa v tomto prípade stávajú rozhodnutia o správnom postupe učenia, tréningovej množine a v neposlednom rade aj kritérium akceptácie. Ak niektorý z týchto parametrov bol nevhodne zvolený, sieť nemusí byť schopná naučiť sa to, čo sa ju snažíme naučiť. Ako však zistíme, že sieť sa učí nesprávne? Ako prvý príznak je čas učenia. Pokiaľ sa chyba dlho drží v okolí jednej hodnoty alebo osciluje, pravdepodobne to znamená, že sa sieť dostala do lokálneho minima v priestore tréningových údajov. V tomto prípade máme na dva hlavné postupy. Môžeme pridať náhodný šum k váham neurónov, alebo nastaviť váhy úplne náhodne. V druhom prípade v podstate začíname s trénovaním od začiatku.

Výber správneho modelu: Pokiaľ sme si istí, že máme vybraný správny model neurónovej siete, tak v takom prípade je dosť možné, že pridaním ďalších skrytých neurónov sa problém vyrieši. Pokiaľ v skrytej vrstve nie je dostatok neurónov, sieť nie je schopná naučiť sa zložité nelineárne funkcie.

Reprezentácia údajov: V niektorých prípadoch je chyba vo vstupných údajoch. Kľúčový parameter nemusí byť správne zakódovaný alebo jeho hodnota nie je vhodne

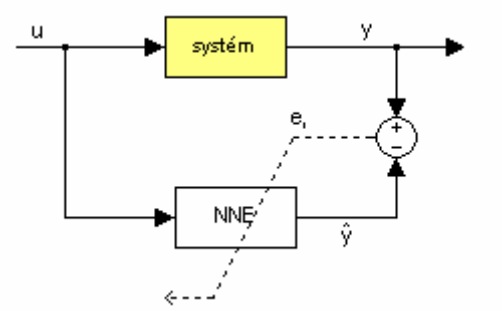
škálovaná. V takom prípade nie je sieť schopná naučiť sa dôležitosť tohto parametra. V horšom prípade v tréningovej množine úplne chýba dôležitý parameter. Táto chyba sa ťažko odhaľuje bez patričnej znalosti problémovej oblasti. Zatiaľ čo by sme sa snažili nastavovať parametre siete, odborník v oblasti by nám mohol poradiť, ktorý parameter vo vstupných údajoch chýba.

Architektúry modelu: V niektorých prípadoch napriek správnemu nastaveniu sieť jednoducho nechce konvergovať. Môže to byť spôsobené prílišnou komplexnosťou problému, aká nie je vhodná pre zvolenú architektúru. Pridávaním ďalších skrytých vrstiev sa výpočtové možnosti siete rozširujú. Každé nové prepojenie je jednou voľnou premennou, ktorú môžeme nastavovať. Preto je vhodnejšie pri trénovaní začínať s komplexnejšou architektúrou siete a postupne jej zložitosť zmenšovať dovtedy, kým je sieť ešte schopná riešiť problém. Problém v tomto prípade môže nastať ak je sieť príliš zložitá. Vtedy môže nastať stav, keď sa sieť perfektne naučí reprezentovať dáta s tréningovej množiny, avšak jej schopnosť zovšeobecňovania sa stratí. Úlohou siete pri trénovaní je rozpoznať vzťahy medzi údajmi a nie sa naučiť presnú odpoveď pre daný vstup.

Vyvarovanie sa pretrénovaniu: Dôležitým bodom pri trénovaní je vedieť, kedy prestať. V tomto prípade neplatí, že viac je lepšie. Pri opakovanom predkladaní rovnakých vzorov sa sieť začne učiť len predkladané vzory a nastaví si váhy tak, aby výpočet presne zodpovedal požadovaným výstupom. Vtedy sieť nedokáže správne zovšeobecňovať na nové vzory a jej výpočty sú nesprávne. Aby sme sa vyhli pretrénovaniu, je vhodné tréningovú množinu rozdeliť na dve časti. Jednou sa bude sieť trénovať a druhou menšou sa bude testovať, či dokáže správne zovšeobecňovať. Testovacie cykly sa vložia medzi trénovacie a trénuje sa dovtedy, kým sieť ešte dáva správne výsledky aj na testovacej množine.

3.6 Modelovanie systémov neurónovými sieťami

Modelovanie riadeného systému je dôležitou oblasťou v uplatnení neurónových sietí v riadení. Spôsob ktorým sa neurónová sieť snaží naučiť riadeného systému môže byť porovnávaný z identifikáciou systému. Informácie získané z riadeného systému sú použité na vytvorenie neurónového modelu. Vytvorením neurónových modelov, získame vhodný model systému, ktorý môžeme použiť v mnohých riadiacich štruktúrach. Neurónový model systému sa v literatúre často označuje skratkou NNE, čo znamená Neural Network Emulator. Na obr. 3.10 môžeme vidieť zapojenie pre vytvorenie neurónového modelu pre reálny systém.



Obr. 3.10 Vytváranie neurónového modelu

Vstup do systému aj neurónovej siete je rovnaký. Výstup siete sa porovná s výstupom systému a na základe chyby týchto dvoch výstupov sa nastavia nové parametre neurónovej siete. Kvalita neurónového modelu je závislá na kvalite a kvantite vstupno-výstupných dát, návrhom architektúry neurónovej siete, použitým pravidlom pre učenia a počtom trénovacích epoch.

Vstupno-výstupné dáta, musia tvoriť vhodnú množinu. Neurónová sieť identifikuje model na základe predložených hodnôt. Dostatočne odlišné vstupné hodnoty vyslané na vstup systému môžu poskytnúť dostatočne odlišné výstupné hodnoty systému. Nemali by sme zabúdať ani na pokrytie celého pracovného rozsahu vstupno-výstupných vzoriek. Odlišnosť dát je veľmi žiadaná, nakoľko čo najväčšia odlišnosť vstupno-výstupných

hodnôt nám dokáže vytvoriť veľmi dobrú trénovaciu množinu. Takto je pravdepodobnosť získania dobrého neurónového modelu oveľa vyššia.

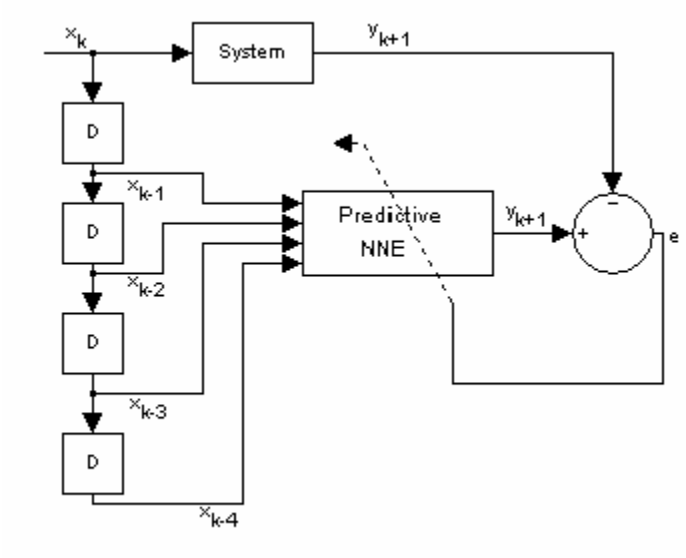
Neurónová sieť modeluje systém funkciou (3.1).

$$y_k = f\left(u_{k-1}, u_{k-2}, u_{k-3}, y_{k-1}, y_{k-2}, y_{k-3}\right) \tag{3.1}$$

Návrh siete udáva, čo môže daná sieť vykonávať. Ak sú dáta nelineárne oddeliteľné, musí sieť obsahovať skrytú vrstvu a používať nelineárne aktivačné funkcie (logsig, tansig ...). Obyčajne oddeliteľnosť dát nie je dôležitá, ak návrhár zvolí skrytú vrstvu ktorá dokáže spracovať ako aj lineárne tak aj nelineárne oddeliteľné dáta [8].

3.6.1 Vytváranie prediktívnych modelov

Prediktívny model, má schopnosť predikcie - predpovedania budúcich stavov resp. výstupov systému. Budúci stav systému o určitý počet časových krokov záleží od oneskorenia vstupu. Ak chceme predpovedať stav systému 5 krokov do budúcnosti, musí byť vstup neurónovej siete oneskorený o 5 časových krokov ako vstup do systému. Počet možných predvídateľných časových krokov teda závisí od rádu modelu [8].



Obr. 3.11 Predvídanie výstupu systému o jeden krok dopredu

3.7 Riadenie systémov s využitím neurónových sietí

Neurónové siete v oblasti riadenia systémov môžu mať rôzne využitie v závislosti na zvolenej štruktúre riadenia. Ich parametre a štruktúra predstavujú napr. dynamiku regulovanej sústavy alebo dynamiku regulátorov. Z hľadiska regulačného obvodu je dôležité správne zvoliť štruktúru neurónovej siete a jej využitie v regulačnom obvode, čo je otázkou voľby vhodnej štruktúry riadenia.

Prvotným krokom nutným pre vývoj neurónových regulátorov pre neznámy nelineárny systém je jeho identifikácia pomocou neurónových sietí. Následne potom prebieha návrh metódy pre riadenie a tvorba neurónového regulátora.

Riadenie nelineárnych systémov je jednou z hlavných aplikačných oblastí pre neurónové siete. Problém návrhu riadenia je na začiatku smerovaný do jedného z dvoch základných spôsobov riadenia: priama a nepriama metóda riadenia [9].

3.7.1 Priama metóda riadenia

Priamy znamená, že regulátor predstavuje priamo neurónová sieť. Neurónový regulátor je výhodný v prípade riadenia systému, ktorého nároky na realizáciu (napr. z hľadiska reálneho času) neumožňujú zložité riešenie regulačného obvodu. Realizácia je jednoduchá, zatiaľ čo návrh a odladenie sú obtiažne z hľadiska opätovného zavedenia a trénovania siete ak dôjde k modifikácii parametrov systému. Preto vo väčšine prípadov trénovanie neurónových regulátorov smeruje k on-line schémam riadenia.

Skupina metód priameho riadenia je okrem niekoľkých výnimiek modelovo založená v tom zmysle, že model systému je potrebný pre návrh regulátora. Niektoré doposiaľ navrhnuté a využívané metódy priameho riadenia:

- Priame inverzné riadenie (realizované ako trénovanie inverzného modelu),
- Riadenie v vnútorným modelom,
- Dopredné riadenie s inverzným modelom,
- Optimálne riadenie

3.7.2 Nepriama metóda riadenia

Hlavnou myšlienkou je využitie neurónových sietí pre model systému, ktorý má byť riadený. Tento model je potom využitý pre tradičný návrh regulátorov. Model je obyčajne trénovaný skôr, ale regulátor je trénovaný on-line. Z toho je zrejmé, že nepriama metóda je veľmi pružná a teda vhodnejšia pre väčšinu úloh riadenia. Medzi doposiaľ známe metódy nepriameho riadenia radíme:

- Aplikácia okamžitej linearizácie na riadenie (aproximácia umiestnenia pólov, aproximácia minimálneho rozdielu),
- Nelineárne prediktívne riadenie.

3.7.3 Priame inverzné riadenie

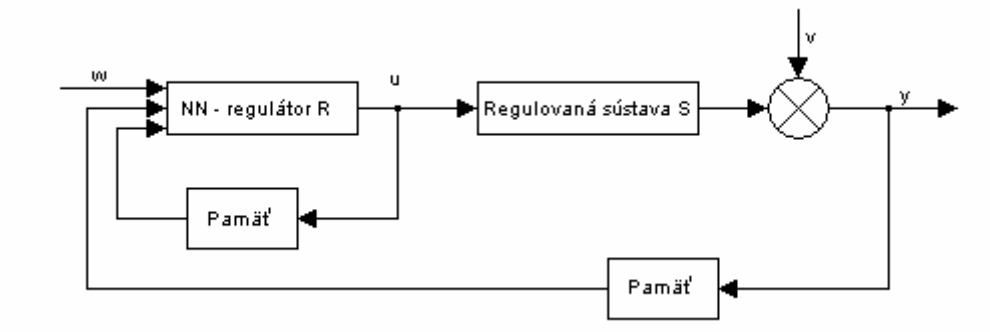
Táto metóda bola navrhnutá ako jedna z prvých metód riadenia neznámych nelineárnych systémov. V stručnosti tu uvádzame základný princíp priameho inverzného riadenia (obr. 3.12). Predpokladajme, že regulovaná sústava je popísaná takto:

$$y(t+1) = g[y(t),...,y(t-n+1),u(t),...,u(t-m)] \tag{3.2}$$

Úpravou vzťahu (3.2) získame vstupný vektor pre požadovanú neurónovú sieť regulátora, ktorá je zostavená zo žiadanej veličiny, aktuálnych a minulých hodnôt výstupnej a akčnej veličiny:

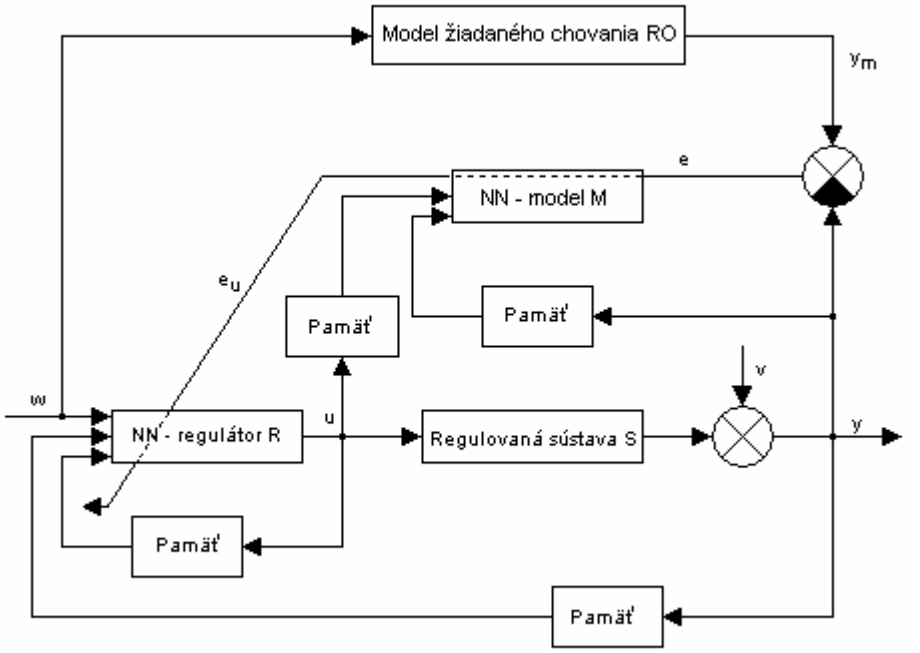
$$\hat{u}(t) = \hat{g}^{-1}[y(t+1), y(t),...,y(t-n+1),u(t),...,u(t-m)] \tag{3.3}$$

Predpokladajme, že máme takúto sieť, ktorá bude pre riadenie systému nahradením výstupnej hodnoty v čase y(t+1) žiadanou hodnotou w(t+1), potom táto neurónová sieť predstavuje inverziu riadeného systému.



Obr. 3.12 Priame inverzné riadenie (obecné trénovanie)

Sú dve techniky pre zostavenie inverzného modelu: off-line metóda známa ako obecné trénovanie a on-line metóda sa nazýva špecializované trénovanie.



Obr. 3.13 Priame inverzné riadenie (špecializované trénovanie)

Vlastnosti priameho inverzného riadenia

Výhody:

- Intuitívne jednoduché
- Jednoduché pre implementáciu
- Neurónový regulátor s použitím špecializovaného trénovania môže byť optimalizovaný pre špecifickú žiadanú trajektóriu
- Špecializované trénovanie je výhodné v prípade zložitých systémov riadenia alebo pre nestacionárne systémy.

Nevýhody:

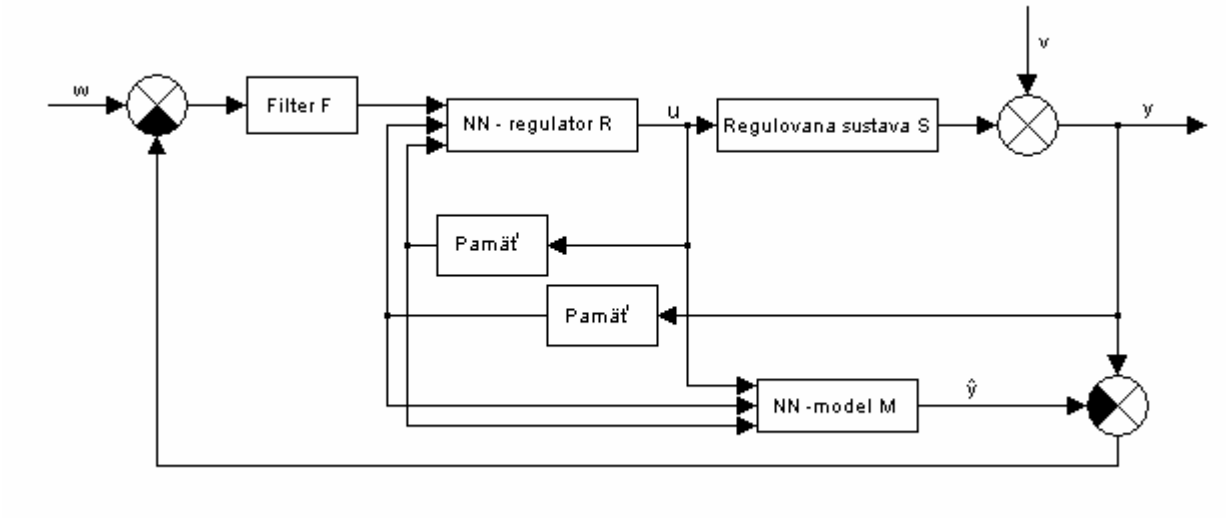
- Nepracuje pre systémy s nestabilnou inverziou, to je žiaľ väčšinou u úloh riadenia, v ktorých je potrebná vysoká frekvencia vzorkovania

- Problémy u systémoch, v ktorých neexistuje ich jedinečný neurónový regulátor (obecné trénovanie inverzných modelov)
- Možné problémy s inverznými modelmi, ktoré nie sú dobre tlmené
- Nedostatok možností nastavenia

3.7.4 Riadenie s vnútorným modelom

Riadenie s vnútorným modelom je metóda riadenia, ktorá je úzko zviazaná s priamym inverzným riadením. Metóda sa vyznačuje veľmi obmedzujúcimi podmienkami pre vlastnosti riadeného systému, čo výrazne obmedzuje platnosť a použitie metódy riadenia s vnútorným modelom. Obmedzujúce podmienky tejto metódy sú v porovnaní s obmedzujúcimi podmienkami priameho inverzného riadenia viac obmedzujúce. Predsa má metóda riadenia s vnútorným modelom niekoľko príjemných vlastností, napr. umožňuje kompenzáciu konštantných porúch.

Vnútorný model regulátora vyžaduje dopredný model rovnako ako model inverzného riadeného systému. Princíp riadenia s vnútorným modelom je znázornený na obr. 3.14.



Obr. 3.14 Riadenie s vnútorným modelom. Regulátor je realizovaný pomocou dvoch neurónových sietí, z ktorých prvá predstavuje model systému (M) a druhá sieť reprezentuje regulátor a je zastúpená inverzným modelom (R).

Filter má vplyv na chovanie uzavretého regulačného obvodu, modelu a tiež na nepriaznivý vplyv porúch na regulačný obvod. Filter musí byť stabilný a mať vhodné zosilnenie aby bolo zaistené sledovanie žiadanej hodnoty.

Vlastnosti riadenia s vnútorným modelom

Riadenie s vnútorným modelom vyžaduje, aby systém rovnako ako jeho inverzný model bol stabilný, a tím vzniká rada úloh, kde táto metóda nemôže byť použitá. Parametrom metódy ja iba filter F. Podobne ako u priameho inverzného riadenia je i pre riadenie s vnútorným modelom obtiažne zostaviť obmedzenie na chovanie akčnej veličiny. V prípade, že dynamika inverzného modelu je kmitavá, akčná veličina má potom kolísavý priebeh s neprimerane veľkými hodnotami.

Väčšina vlastností metódy riadenia s vnútorným modelom je zhodná s metódou priameho inverzného riadenia. Ďalšie špeciálne vlastnosti pre túto metódu sú:

- Odozva riadeného systému na konštantnú poruchu zostáva na požadovanej hodnote bez trvalej regulačnej odchýlky,
- Nutnosť zaistenia stability systému v otvorenom cykle.

3.7.5 Dopredné riadenie s inverzným modelom

Metódu dopredného riadenia s inverzným modelom je vhodné aplikovať vtedy, ak vznikla požiadavka na lepšie sledovanie žiadanej hodnoty, zatiaľ čo spätná väzba spôsobí stabilizáciu systému a potlačenie poruchy.

Akčná veličina = dopredný signál + signál od spätnej väzby

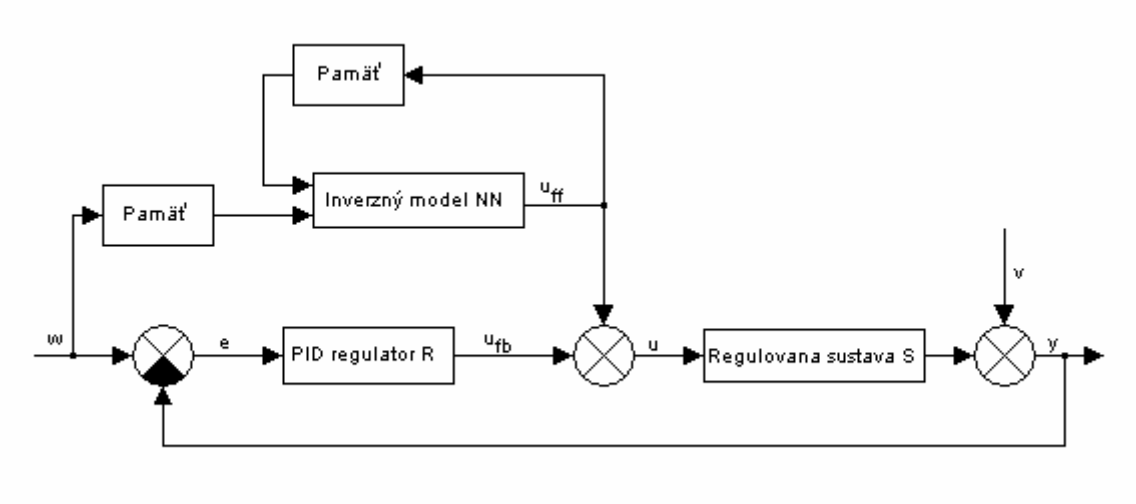
Odbor, v ktorom je rýchle sledovanie žiadanej hodnoty nevyhnutné sa nazýva robotika. Preto sa metóda dopredného riadenia najčastejšie aplikuje pri riadení robotov.

Poznáme dva typy dopredných stratégií: dynamickú a statickú. Metóda dopredného riadenia s inverzným modelom využíva „tradičný“ (P, PI, PD alebo PID) regulátor a tiež doprednú neurónovú sieť.

Existujú rôzne spôsoby pre zavedenie dopredných neurónových regulátorov. Väčšinou sa využívajú dynamické dopredné neurónové siete. Dopredné zložky vstupu

neurónovej siete sú zložené z minulých žiadaných veličín, ktoré nahradzujú minulé regulované veličiny a z minulých akčných veličín.

Princíp metódy dopredného riadenia s inverzným modelom je blokovo znázornený na obr. 3.15. Ak je inverzný model stabilný, potom jeho aplikácia v regulačnom obvode nezmení vlastnosti stability riadeného systému v uzavretom cykle. Avšak môže byť obtiažne vyriešiť či inverzný model bude alebo nebude naozaj stabilný.



Obr. 3.15 Optimalizácia existujúceho systému riadenia vložením syntetického dopredného signálu neurónovej siete, ktorý je odvodený zo žiadanej hodnoty

V prípade, keď nevhodná akčná veličina spôsobí nevhodnú odozvu riadeného systému, je aplikácia tejto metódy veľmi vhodná. Ďalšou užitočnou možnosťou pre dynamické dopredné riadenie je využitie statického dopredného riadenia. Tento prístup rieši možnú stabilitu úlohy dynamického modelu. Stabilné dopredné riadenie nie je vhodné pre nestabilné systémy, keď akčné veličiny sú v stabilnom stave nulové. Aj keď je neurónová sieť v metóde dopredného riadenia užitočná pre optimalizáciu riadiacich systémov, aj tak je nutné aplikovať túto metódu so zvýšenou opatrnosťou. Nepresnosti dopredného riadenia môžu v skutočnosti skôr uškodiť ako skvalitniť proces riadenia.

Vlastnosti dopredného riadenia s inverzným modelom

Metóda je skôr zameraná na zlepšenie schopností sledovať žiadanú hodnotu než na situácie, keď dochádza ku zmenám dynamiky uzavretého regulačného obvodu. Vlastnosť jednoduchej implementácie, postupné zavedenie prídavného signálu a zachovanie

stávajúceho regulačného obvodu smeruje na pozvoľný prechod k systémom riadenia založených na neurónových sieťach.

Výhody:

- Zavedenie metódy je jednoduché
- Metóda môže byť zavedená pozvoľna
- Zlepšenie sledovania žiadanej hodnoty bez narastajúcej citlivosti na šum
- Experiment jednoducho uskutočniteľný, za predpokladu, že tradičný (PID) regulátor je stabilný

Nevýhody:

- Vyžaduje prítomnosť spätno-väzbového regulátora
- Môžu nastať problémy v prípade, pokiaľ je inverzia systému nestabilná alebo pokiaľ výstup získaného inverzného modelu kmitá

3.7.6 Optimálne riadenie

Myšlienkou optimálneho riadenia je navrhnúť regulátor na základe kritéria, keď sledovanie žiadanej hodnoty je ziskom, zatiaľ čo tu je „pokuta“ hodnoty (zastúpená nejakou funkciou) akčnej veličiny. Metóda je zavedená trénovaním neurónovej siete pre poskytnutie akčných veličín.

Zatiaľ čo regulátory popísané v predchádzajúcich častiach boli všetky navrhnuté tak, aby vytvorili uzavretý regulačný obvod, ktorý sa správa lineárne v okamžiku vzorkovania, takto nepracuje optimálny regulátor. Systém v uzavretom obvode sa môže v skutočnosti správať odlišne v rôznych režimoch operačného rozsahu. Úplné testovanie systému v uzavretom obvode je doporučené, aby bolo zaistené uspokojivé správanie regulátorov pre všetky operačné rozsahy.

Hlavným problémom optimálneho riadenia je zrejme vhodná inicializácia regulátorov. Rovnako ako u špeciálneho trénovania, je vhodné inicializovať sieť prístupom off-line, čo sa realizuje ako simulácia modelu systému pre daný reálny systém.

Vlastnosti optimálneho riadenia

Metóda optimálneho riadenia aplikuje regulátor ako rozšírenie špecializovaného trénovania inverzných regulátorov.

Výhody:

- Jednoduché pre nastavenie
- Zavedenie pre veľkú skupinu systémov (v zrovnaní s prísnym inverzným riadením)
- Cielené riadenie
- Vhodné pre návrh regulátorov pre špecifickú žiadanú trajektóriu
- V princípe je to priamo aplikovateľné na systémy meniace sa v čase

Nevýhody:

- Sieť regulátorov je trénovaná on-line
- Sieť regulátorov musí byť pretrénovaná vždy, keď je faktor penalizácie modifikovaný
- Inicializácia siete je obtiažna

3.8 Príklad na aplikáciu neurónovej siete v prostredí MATLAB ©.

Vytvorenie aplikácie v prostredí MATLAB © a popis jednotlivých častí tejto aplikácie [12].

```
%Vstupne data
InputVector = 0:2*pi/16:2*pi;
TargetVector = sin(InputVector);
plot(InputVector,TargetVector,'x');
```

Základom aplikácie neurónových sietí je vstupná vzorová množina. Na tú sa sieť trénuje. V našom prípade bude vzor tvoriť množina bodov sínusoidy. Vstupný vektor siete (InputVector) je tvorený 32 lineárne rozdelenými bodmi intervalu <0,2pi>. Target vektor siete (TargetVector) je tvorený sínusovými hodnotami vstupného vektoru. Pre názornosť sú body vynesené do grafu pomocou príkazu **`plot().`**

```
%Vytvorenie objektu neuronovej siete
PR = minmax(InputVector);
```

```
Layers = [2 1];
TransFcns = {'logsig' 'purelin'};
BTF = 'traingdx';
BLF =  'learngdm';
PF = 'mse';
net = newff(PR,Layers,TransFcns,BTF,BLF,PF);
```

V tejto časti programu dochádza k vytvoreniu objektu neurónovej siete. Existuje niekoľko príkazov, ktoré vytvárajú objekt neurónovej siete. Ak chceme vytvoriť doprednú neurónovú sieť bez spätných väzieb, použijeme príkaz **`newff()`**. Ten vyžaduje niekoľko vstupných premenných:

- **`PR`** – udáva rozsah vstupných hodnôt (minimálna a maximálna hodnota elementov vstupných vektorov)
- **`Layers`** – udáva počty neurónov v jednotlivých vrstvách. Počet vrstiev nie je obmedzený. Najčastejšie sa používajú dve vrstvy.
- **`TransFcns`** – udáva prenosové (aktivačné) funkcie neurónov jednotlivých vrstiev.
- **`BTF`** – udáva názov trénovacej funkcie siete (beachtraining – dávkové učenie).
- **`BLF`** – udáva názov learning funkcie siete (steptraining).
- **`PF`** – udáva názov funkcie pre výpočet chyby pri trénovaní.

Do premennej **`net`** sa potom uložia dáta vytvoreného objektu neurónovej siete. Objekt **`net`** kóduje všetky paradigmaty neurónovej siete. Sú tu uložené informácie o topológii, prenosových funkciách, chybových funkciách, hodnotách váh,, prahov, ... Dátovú štruktúru objektu neurónových sietí zobrazíme zadaním názvu premennej **`net`** a potvrdíme klávesou ENTER.

```
%Inicializacia
net = init(net);
```

Pred vlastným procesom trénovania je sieť potrebné inicializovať. Počas inicializácie dochádza k východziemu nastaveniu váh a prahov neurónovej siete. To, aká inicializačná metóda sa má použiť, je uložené v premenných:

- `net.initFcn`
- `net.initParam`

- `net.layers{i}.initFcn`
- `net.inputWeights{i}.initFcn`
- `net.layerWeights{i,j}.initFcn`

Inicializácia sa urobí implicitne už pri tvorbe neurónovej siete. V našom prípade teda použijeme túto inicializáciu.

```
%Trenovanie siete
[net,tr] = train(net,InputVector,TargetVector);
```

Vlastný proces trénovania sa spúšťa pomocou príkazu **`train()`**. Vstupnými a výstupnými premennými sú:

- **`net`** – názov objektu siete, ktorá sa má trénovať. Naučenú sieť možno uložiť do novej premennej (viď výstupné dáta príkazu **`net()`**).
- **`InputVector`** – vstupný vektor dát vzorovej množiny.
- **`TargetVector`** – vektor požadovaných hodnôt dát vzorovej množiny.
- **`tr`** – [training record] dáta reprezentujúce priebeh trénovacieho procesu.

Metóda, ktorá sa k trénovaniu použije a parametre trénovania, sú opäť uložené v objekte **`net`** v premenných **`net.trainFcn`** a **`net.trainParam`**. O nastavovaní parametrov sa zmienime neskôr. Počas trénovania sa postupne vykresľuje trénovacia chybová krivka. Tá ukazuje, ako postupne klesá chybová funkcia trénovania.

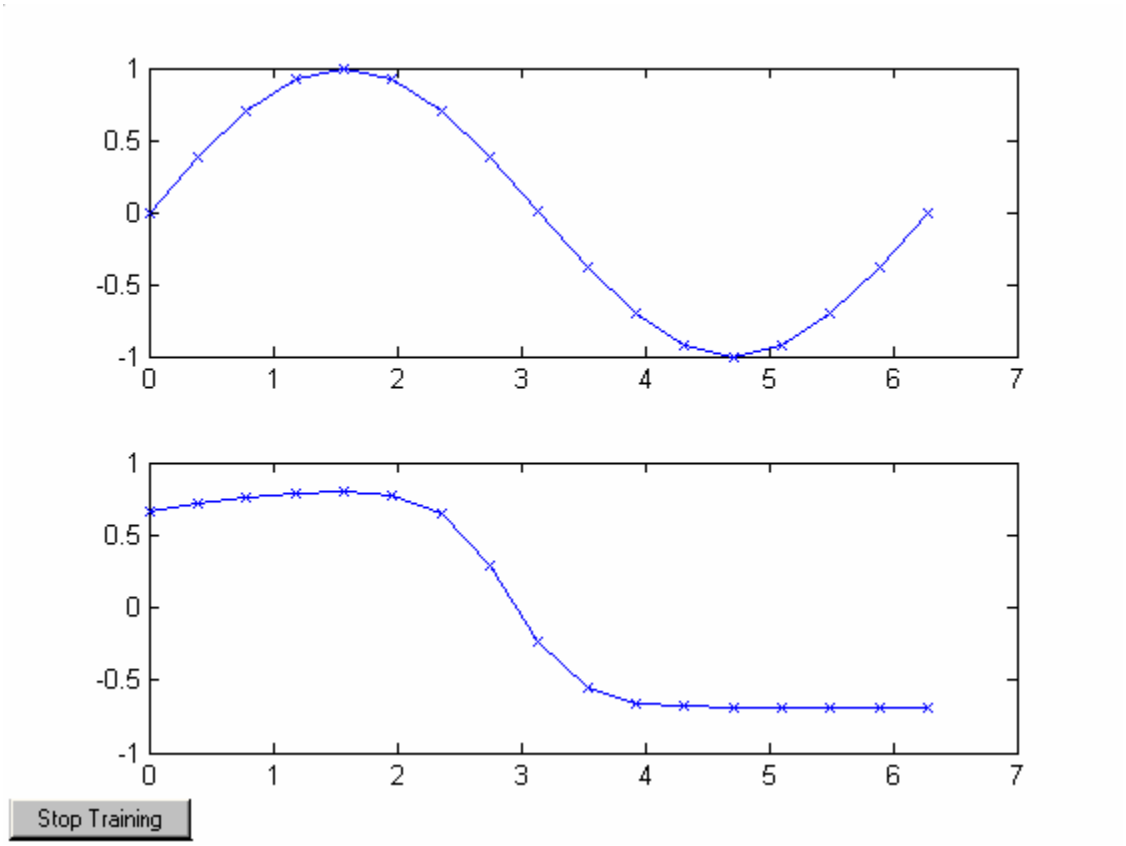
```
%Simulacia
[OutputVector] = sim(net,InputVector);
subplot(2,1,1);
plot(InputVector,TargetVector,'x-');
subplot(2,1,2);
plot(InputVector,OutputVector,'x-');
```

Neurónovú sieť máme natrénovanú. Teraz ostáva overiť, ako úspešne sa natrénovala. K simulácii siete slúži príkaz **`sim()`**. Vstupnými a výstupnými premennými sú:

- **`net`** – meno objektu siete, pomocou ktorej sa má simulácia previesť.
- **`InputVector`** – vstupné dáta simulácie. Ide o definičný obor simulácie. V našom prípade simulujeme presne tie dáta, na ktoré bola sieť naučená (trénovacie dáta).

- **`OutputVector`** – výstupné dáta simulácie. Ide o obor hodnôt simulácie.

Aby sme mohli posúdiť, ako sa neurónová sieť naučila vykreslíme vstupné vzorkové dáta a výstupné simulované dáta do spoločného grafu.



Obr. 3.16 Graf vstupných dát a výstupných simulovaných dát

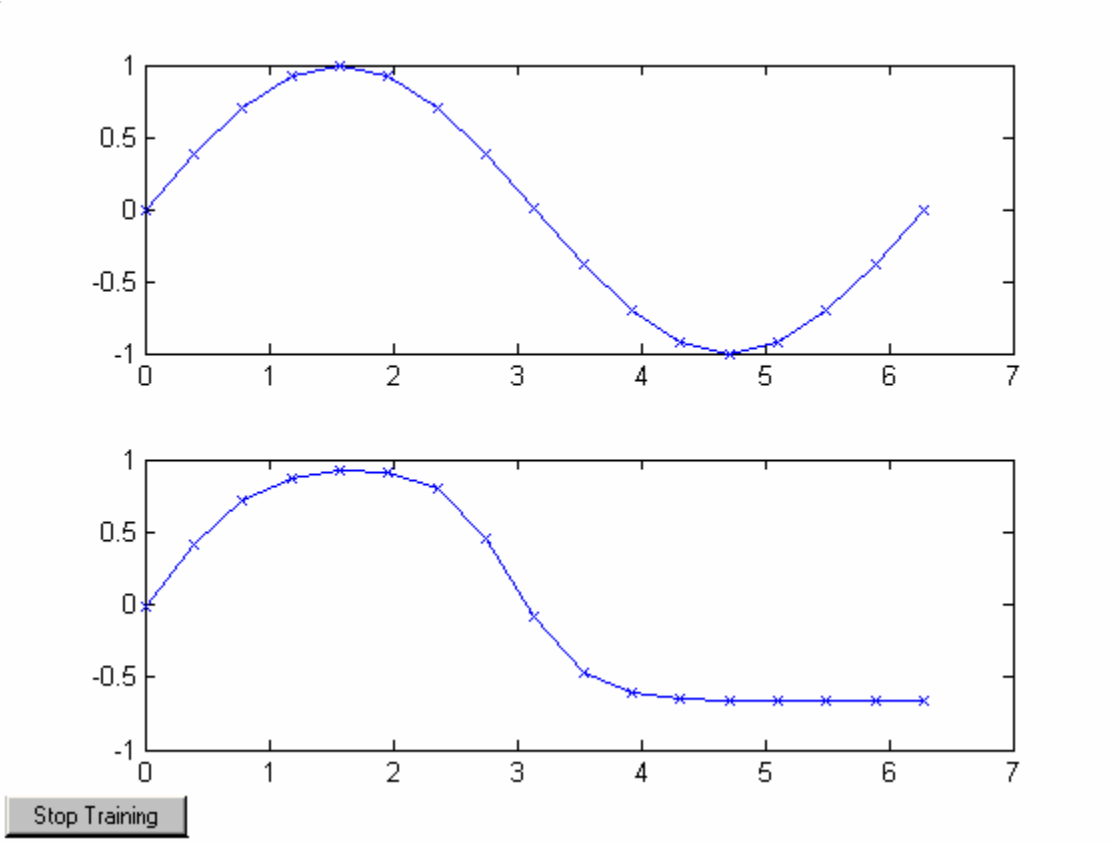
3.8.1 Nastavenie parametrov trénovania

Ako je vidieť z predchádzajúceho obr. 3.16, neurónová sieť sa moc dobre nenaučila. Stále generuje výstupy s veľkou chybou. Problém je v nastavení trénovacích parametrov. Po ukončení trénovania stále ešte klesala chybová funkcia. Zmeníme preto trénovacie parametre. Tie sú uložené v dátovej štruktúre objektu net. Pred príkazom trénovania **`[net,tr] = train(net,InputVector,TargetVector);`** zmeníme niektoré z nasledujúcich parametrov učenia.

- **`net.trainFcn`** – udáva, aká trénovacia funkcia sa má použiť. K dispozícii sú funkcie **`traing`**, **`traingda`**, **`traingdm`**, **`traingdx`**, **`trainlm`**, ... Podrobnosti o trénovasích funkciách sa dajú nájsť v help-e.

- **`net.trainParam.epochs`** – udáva, koľko epoch sa má neurónová sieť naučiť. Východzia hodnota je 100. Hodnotu parametra nastavíme na 500.
- **`net.trainParam.show`** – udáva, ako často sa má chybová funkcia vykresľovať. Východzia hodnota je 25. Po každých 25 epochách učenia nám teda MATLAB © ukáže, ako sa sieť učí (vykreslením chybovej funkcie).
- **`net.trainParam.lr`** – [learning rate] udáva, ako sa majú meniť koeficienty váh a prahov v závislosti na chybovej funkcii. Udáva teda rýchlosť učenia. S týmto parametrom sa pokúsime experimentovať. Ak bude lr malé, neurónová sieť sa bude učiť pomaly. Ak bude lr veľké bude neurónová sieť pri učení oscilovať.
- **`net.trainParam.goal`** – udáva, pri akej hodnote chybovej funkcii sa má trénovanie ukončiť. S týmto parametrom sa pokúsime opäť experimentovať.
- **Všetky nastaviteľné parametre učenia sa vypíšu zadaním príkazu `net.trainParam`.**

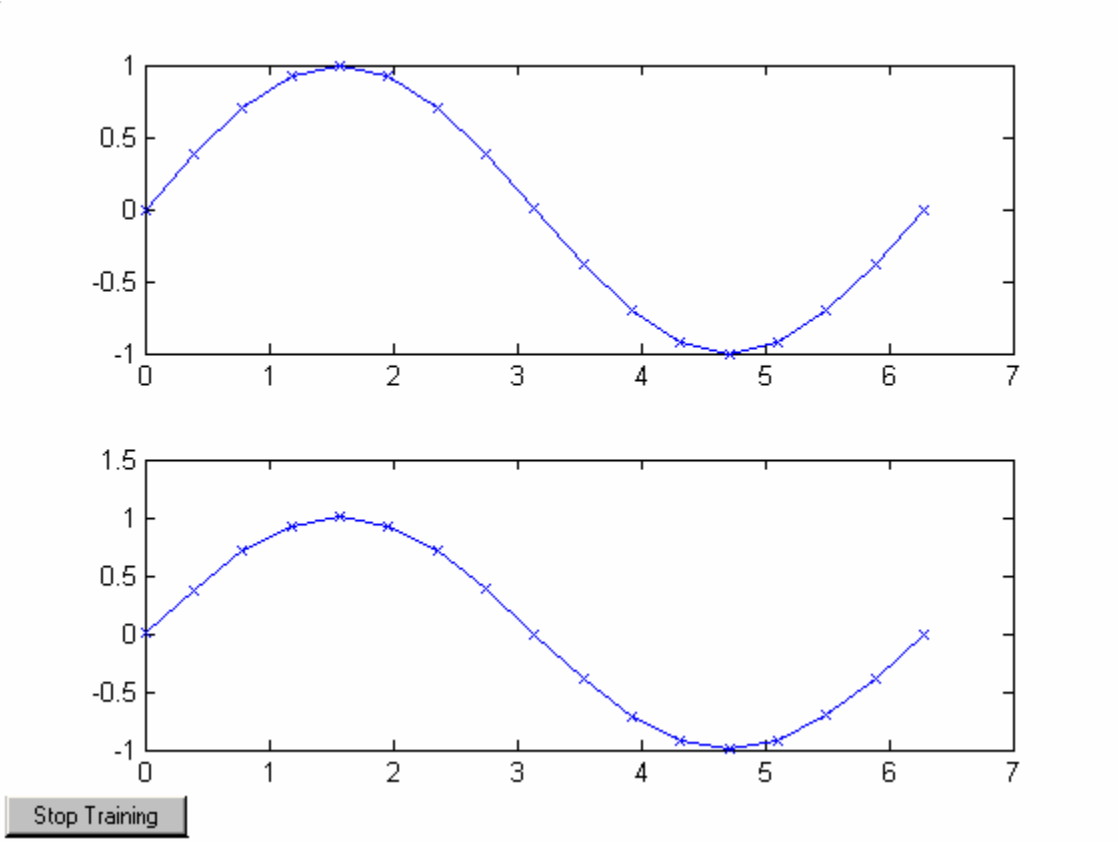
Výsledný priebeh trénovacej funkcie bude vypadať podobne ako na nasledujúcom obr. 3.17. Ako je vidieť, neurónová sieť sa naučila lepšie. Priebeh chybovej funkcie už ďalej neklesá. Aj tak však výstup zo simulácie neodpovedá svojmu vzoru. Je to preto, že je malá sieť (zvolená sieť má málo neurónov). Zmení sa teda veľkosť siete.



Obr. 3.17 Graf vstupných dát a výstupných simulovaných dát

3.8.2 Optimalizácia počtu neurónov siete, generalizácia

Ako sme videli na obr. 3.17, dva neuróny v prvej vrstve a jeden výstupný neurón pre aproximáciu nami zadaných dát nestačí. Zmeníme teda počet neurónov v prvej vrstve. Upravíme premennú takto: **Layers=[20 1]**. Naša neurónová sieť teda bude mať 20 neurónov v prvej vrstve a 1 neurón vo výstupnej vrstve. Upravený skript opäť pustíme. Chybová funkcia klesla omnoho viac. Experimentujeme s koeficientami trénovania tak, aby sme dosiahli čo najlepšiu naučenosť neurónovej siete.

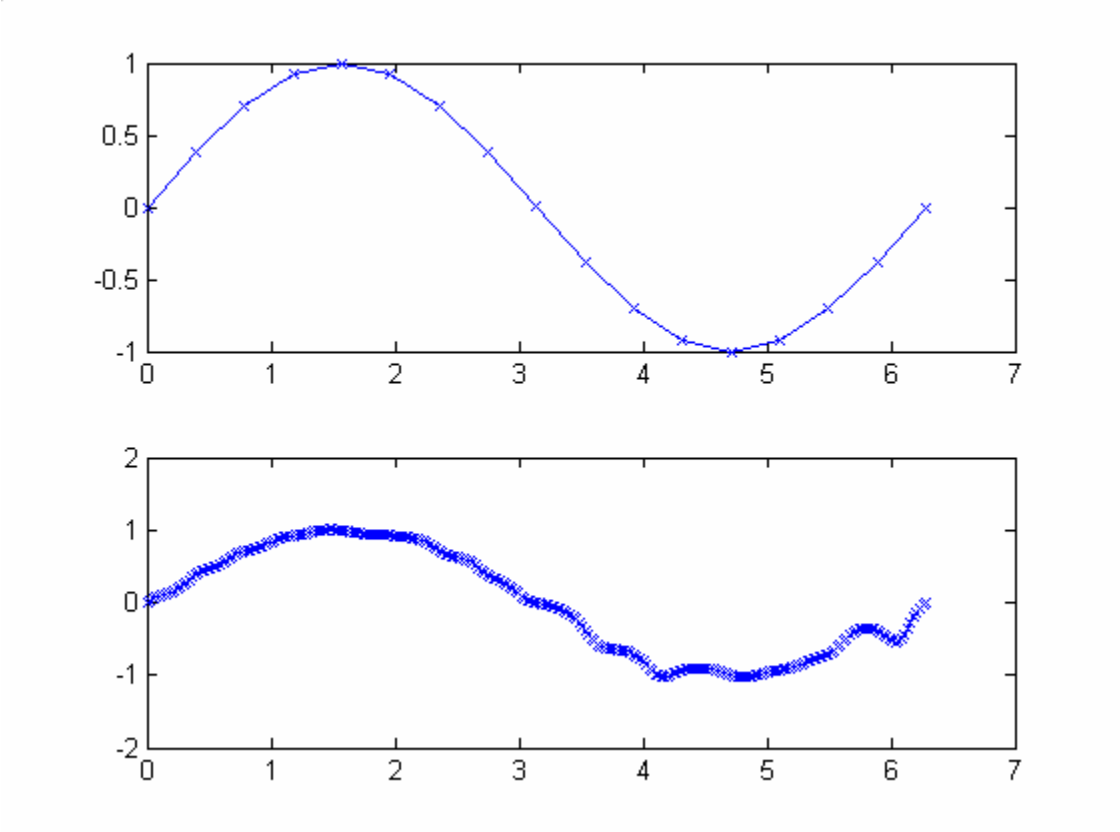


Obr. 3.18 Graf vstupných dát a výstupných simulovaných dát

Z tohoto pokusu vyplýva, že čím viac neurónov v neurónovej siete použijeme, tým lepšie sa neurónová sieť vzorové dáta naučí. Problém nastáva, keď chceme, aby neurónová sieť generovala dáta, na ktoré nebola naučená.

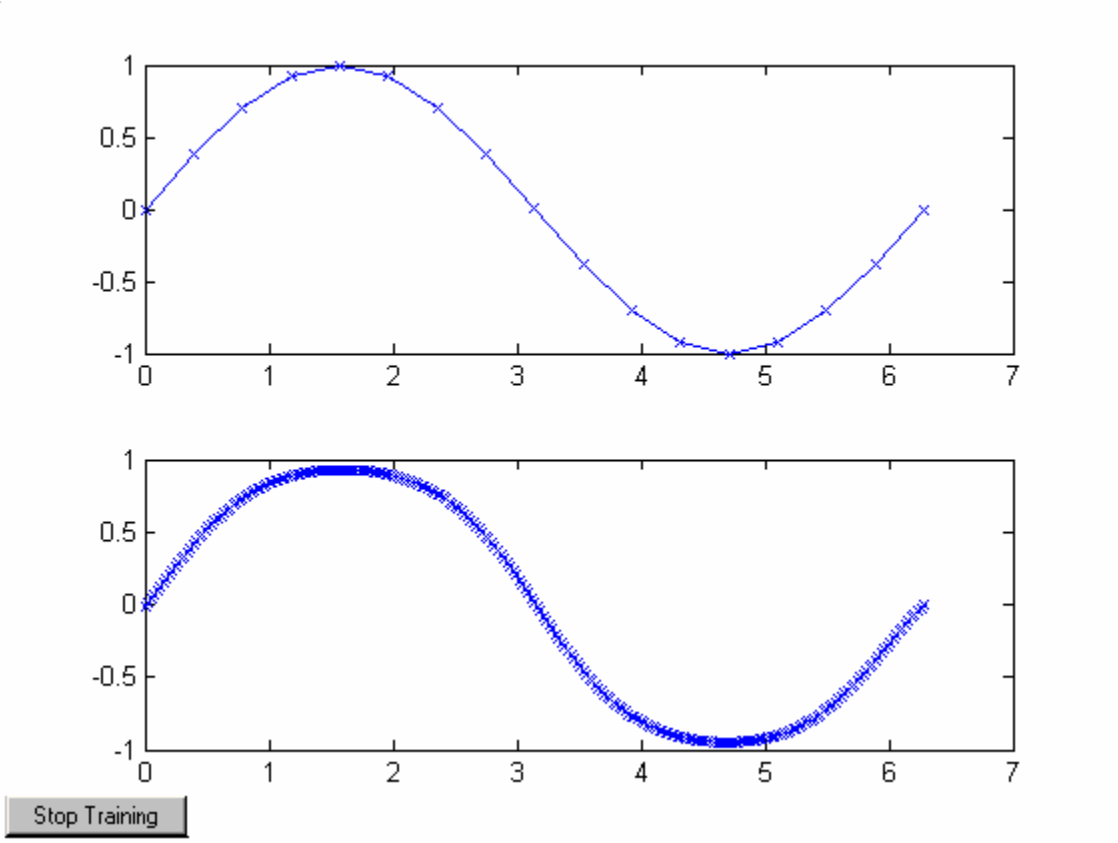
Zmeníme vstupný vektor simulácie. Nebudeme simulovať iba dáta, na ktoré sme neurónovú sieť naučili, ale i tie, ktoré neurónová sieť nepozná (validačné a testovacie dáta). Urobíme to nasledovne:

```
%Simulacia
ExtendedInputVector = 0:2*pi/256:2*pi;
[OutputVector] = sim(net,ExtendedInputVector);
subplot(2,1,1);
plot(InputVector,TargetVector,'x-');
subplot(2,1,2);
plot(ExtendedInputVector,OutputVector,'x-');
```



Obr. 3.19 Graf vstupných dát a výstupných simulovaných dát

Ako je vidieť na obr. 3.19, v bodoch, ktoré neboli súčasťou trénovacej databázy odpovedá neurónová sieť chybne. Svedčí to o tom, že je neurónová sieť predimenzovaná. 20 neurónov v prvej vrstve je príliš veľa. Pre jednu periódu sínusoidy by mala stačiť neurónová sieť o konfigurácii 1-3-1. Vyplýva to z podstaty neurónu. Jeden neurón je schopný generovať iba jeden zlom funkcie. Keďže funkcia sínus má tieto zlomy tri, sú potrebné tri neuróny.



Obr. 3.20 Graf vstupných dát a výstupných simulovaných dát

3.9 Príklad na modelovanie neurónovej siete v prostredí MATLAB ©

Cely súbor vytvorení v prostredí MATLAB © **neuron.m** na modelovanie neurónovej siete je v prílohe 2. Ako vstupné dáta sme použili dáta pri simulácii zásobníkov kvapaliny z príkladu pre fuzzy riadenie.

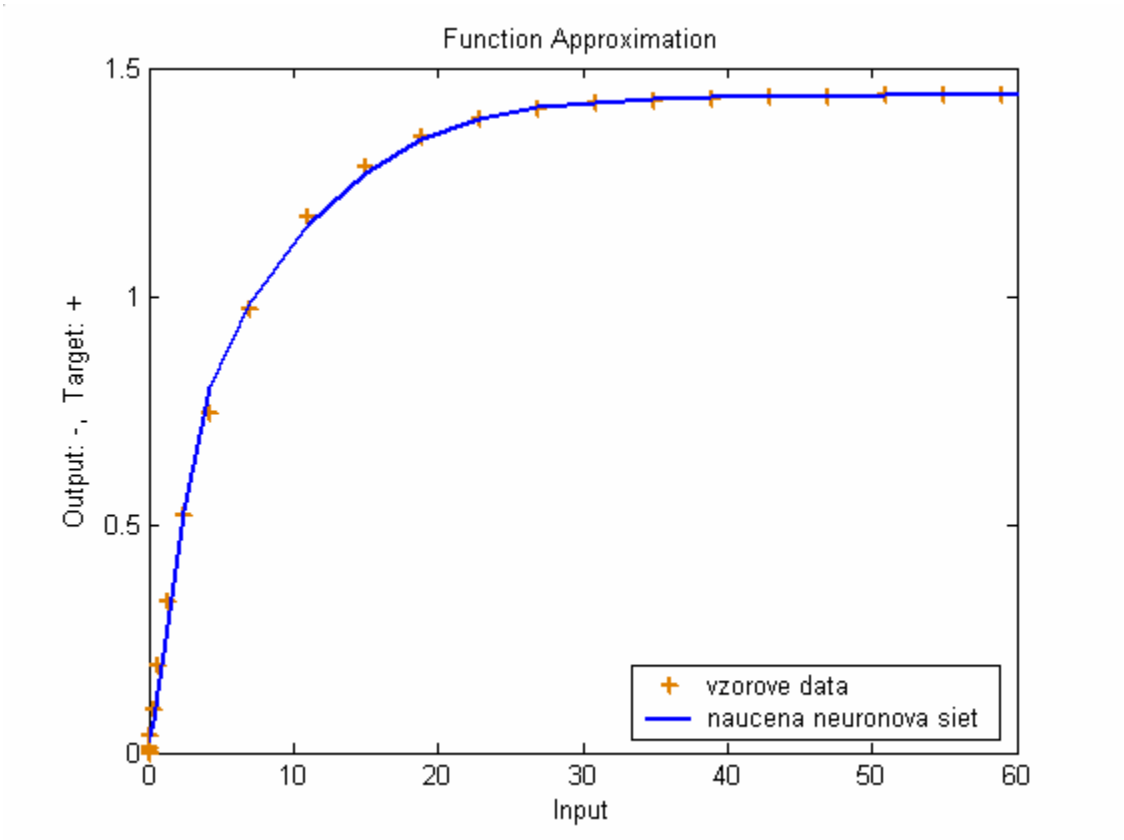
```
A = data(:,1);
u=A';
B = data(:,2);
y=B';
plot(u,y,'x');
```

Najprv sme si neurónovú sieť inicializovali. Použili sme dvojvrstvovú sieť so 4 neurónmi vo vnútornej vrstve. Ako výstupná funkcia bola použitá logsigmoidálna funkcia vo vnútornej vrstve a lineárna funkcia vo výstupnej vrstve.

```
[W1,b1,W2,b2] = initff(u,4,'logsig',y,'purelin');
```

Potom sme neurónovú sieť trénovali algoritmom backpropagation, kde sme si do premennej tp definovali parametre učenia.

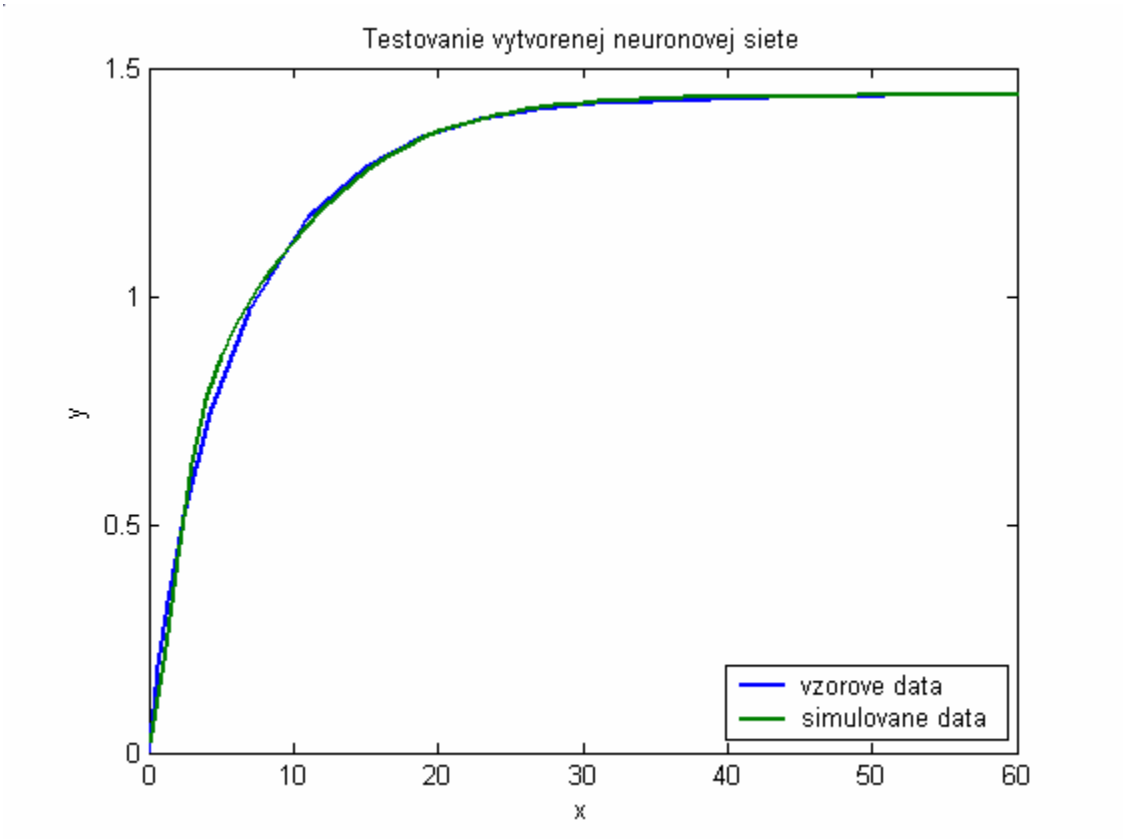
```
df = 50;   % frekvence zobrazování kroku
me = 5000; % maximální pocet kroku ucení
eg = 0.001; % cílová kvadratická chyba
lr = 0.01; % ucící konstanta

tp = [df me eg lr];

[W1,b1,W2,b2,ep,tr] = ...
trainbp(W1,b1,'logsig',W2,b2,'purelin',u,y,tp);
```



Obr. 3.21 Vzorové dáta a naučené dáta

Nakoniec sme neurónovú sieť otestovali nami zadanými hodnotami.

```
x=0:200
y = simuff(u,W1,b1,'logsig',W2,b2,'purelin');
```



Obr. 3.22 Vzorové a simulované dáta

4. Záver

Práca vytvára moduly pre e-learningový výučbový systém pre predmet inteligentné systémy riadenia v oblasti fuzzy riadenia a umelých neurónových sietí. Študíjny materiál umožňuje osvojiť si základné informácie o spomínaných inteligentných systémoch riadenia a pomáha s ich využitím v praktických aplikáciach.

Na úvod fuzzy riadenia je dôležité sa oboznámiť s teóriou fuzzy množín a lingvistickými premennými. V práci sú popísané typy funkcií príslušnosti a operácie s fuzzy množinami, ktoré sú pre lepšie pochopenie doplnené obrázkami. V kapitole 2.4 je popísaný fuzzy regulátor, jeho časti a typy. Jeho jednotlivé časti sú fuzzifikácia, inferenčný mechanizmus, defuzzifikácia a báza pravidiel, ktorá je rozhodovacím mechanizmom fuzzy regulátora. Na obrázkoch 2.15 až 2.17 sú naznačené aj iné možnosti využitia fuzzy systému pri riadení. Ďalšia kapitola opisuje návrh regulátora pomocou interaktívneho grafického prostredia v programe MATLAB ©, ktorý obsahuje nástroje pre vytvorenie, editáciu a zobrazovanie fuzzy inferenčného systému, a pomocou príkazov, ako je ukázané na príklade riadenia zásobníkov kvapaliny.

V úvode umelých neurónových sietí je popísané, čo vlastne neurónové siete sú, ich história a v akých úlohách sa môžu použiť. Ďalej je opísaná stavba neurónu, ako základného prvku umelých neurónových sietí, typy aktivačných funkcií, základné rozdelenie a spôsoby učenia neurónových sietí. V kapitole 3.5 sú uvedené niektoré modely neurónových sietí a zásadné problémy, ktoré môžu nastať pri výbere modelu. V ďalšej kapitole je popísané modelovanie systémov a následne v kapitole 3.7 riadenie systémov s využitím umelých neurónových sietí. V tejto kapitole sú popísané rôzne metódy riadenia s jeho vlastnosťami. V práci sú uvedené 2 príklady na aplikáciu neurónovej siete, a to v kapitole 3.8 a 3.9. V prvom príklade je ukázaná jedna z aplikácií neurónových sietí, jej vytvorenie, trénovanie a následne jej simuláciu. Druhý príklad slúži na modelovanie neurónových sietí na základe vstupných a výstupných dát systému. Ako systém boli použité zásobníky kvapaliny, ktoré sa vyskytli aj v príklade fuzzy riadenia.

Fuzzy riadenie sa osvedčilo v mnohých aplikáciách v oblasti riadenia systémov. Príkladom je riadenie cementárskych pecí, výmenníkov tepla, zásobníkov kvapaliny

a mnoho ďalších. K jednej z najdôležitejších aplikácií patrí doprava. Fuzzy riadenie je snáď najlepšie spracovaná časť prostriedkov umelej inteligencie. Použitie neurónových sietí je praktické, keď nevieme opísať proces a máme o ňom iba vstupné a výstupné informácie. Neurónové siete môžeme prirodzeným spôsobom použiť napr. k rozpoznávaniu naskenovaných, napísaných resp. tlačených znakov. Ďalšou možnosťou použitia je riadenie zložitejších zariadení v dynamicky sa meniacich podmienkach. Efektívnym využitím je použitie pri riešení problémov v oblastiach klasifikácie, riadenia procesov, predikcie, aproximácie, spracovanie signálov a pod.

5. Literatúra

[1]. Modrlák O., 2004. Fuzzy řízení a regulace. [online]. Liberec: FM TU. Dostupné na internete: http://www.fm.vslib.cz/~krtsub/fm/tr2/tar2_fuz.pdf

[2]. Rydval. S., 2005. Základy fuzzy logiky, [online]. Dostupné na internete: http://www.rydval.cz/phprs/view.php?cisloclanku=2005061701

[3]. Šoltys J.**,** 1996. Použitie fuzzy neurónovej siete typu NARA na riadenie vozíka s balancujúcou tyčou. [online]. Košice: FEI TU. Dostupné na internete: http://www.ai-cit.sk/source/publications/thesis/master_thesis/1996/soltys/html/

[4]. Gaľová L., Surgent S., Esej z predmetu Umelá Inteligencia. [online]. Košice: FEI TU. Dostupné na internete: frtk.fri.utc.sk/~cechovic/**Fuzzy**_logic.doc

[5]. Žitniak J., Reprezentácia neurčitých poznatkov a Fuzzy množiny. Bratislava: FEI STU

[6]. Navara M., Fuzzy řizení. [online]. Praha: FE ČVUT. Dostupné na internete: http://cmp.felk.cvut.cz/~navara/m6f/fcontrol.pdf

[7]. Vevurka M., 2004, Dolovanie údajov a neurónové siete, [online].Bratislava: FIIT STU. Dostupné na internete: http://www2.fiit.stuba.sk/~kapustik/ZS/Clanky0405/**vevurka**/datamining.html

[8]. Bucko V., 2004, Neurónové siete v teórii riadenia. Bratislava: FEI STU

[9]. Škutová J., 2004, Neuronové sítě v řízení systémů, [online]. Ostrava FS VSB. Dostupné na internete: http://www.fs.vsb.cz/books/NeuronoveSite/

[10]. Sinčák P., Andrejková G., 1996, Neurónové siete I (Inžiniersky prístup). [online]. Košice: FEI TU, PF UPJS Dostupné na internete: http://www.ai-cit.sk/source/publications/books/NS1/html/all.html

[11]. Sekaj I., ‚Fuzzy a neurónové systémy. [online]. Bratislava: FEI STU. Dostupné na internete: http://www.kasr.elf.stuba.sk/predmety/fnr/

[12]. Tučková J., 2005, Aplikace neuronové sítě v prostředí MATLAB ©. [online]. Praha: FE ČVUT. Dostupné na internete: http://amber.feld.cvut.cz/ssc/CV/nnetDemo/INDEX_05.HTM

Príloha 1

Súbor **fuzzy.m** na riadenie zásobníkov kvapaliny pomocou fuzzy regulátora

```
clc
clear

% parametre sustavy
k11=1.25;
k22=2;
k33=1.5;
F=4;
qv1=1.5;
qv2=0.5;
qv3=0.25;

% vypocet ustalenych vysok hladin
h1s=(qv1/k11)^2;
h3s=((qv1+qv2+qv3)/k33)^2;
h2s=((qv1+qv2)/k22)^2+h3s;
h=[h1s,h2s,h3s]

% vypocet konstant prietokomera
K1=(0.5*k11)/sqrt(h1s);
K2=(0.5*k22)/sqrt(h2s-h3s);
K3=(0.5*k33)/sqrt(h3s);

% vypocet zosilneni
Z1=1/K1;Z2=1/K2;Z3=1/K3;
Z=[Z1 Z2 Z3]

% vypocet casovych konstant
T1=F/K1;T2=F/K2;T3=F/K3;
T=[T1 T2 T3]

% matice stavoveho opisu
A=[-K1/F 0 0; K1/F -K2/F K2/F;0 K2/F -(K2+K3)/F]
B=[1/F 0 0;0 1/F 0;0 0 1/F]
C=eye(size(A))
D=zeros(size(B))

% prenosy
[cit1,men1]=ss2tf(A,B,C,D,1)
[cit2,men2]=ss2tf(A,B,C,D,2)

% ustalene vysky hladin
open_system('ustalene_hladiny')
sim('ustalene_hladiny',200);
figure(1)
plot(data(:,1),data(:,2),'b',data(:,1),data(:,3),'g',data(:,1),data(:,4
),'r')
xlabel('cas simulacie (min)')
ylabel('vyska hladiny')
title('Porovnanie prechodových charakteristík')
```

```
legend('ustalena vyska hladiny v 1. zasobniku','ustalena vyska hladiny
v 2. zasobniku','ustalena vyska hladiny v 3. zasobniku',4)

bdclose('ustalene_hladiny')

figure(1)
plot(data(:,1),data(:,4))
xlabel('cas simulacie (min)')
ylabel('vyska hladiny h_3(dm)')


%%%%%%% MAMDANI 3MF %%%%%%%

p=newfis('mamdani3.fis')
showfis(p)

% pridanie vstupu e
p=addvar(p,'input','e',[-2.6642 2.7758]);   % [vstupny rozsah]
% pridanie funkcie prislusnosti pre vstup e
p=addmf(p,'input',1,'mf1','trimf',[-4.72 -2.38336507936508 -0.331]);
p=addmf(p,'input',1,'mf2','trimf', [-2.636 -0.44 1.751]);
p=addmf(p,'input',1,'mf3','trimf', [-0.0944 1.3 4.2]);
figure(2)
plotmf(p,'input',1)

% pridanie vstupu de
p=addvar(p,'input','de',[-0.2604 0.1982]) ;  % [vstupny rozsah]
% pridanie funkcie prislusnosti pre vstup de
p=addmf(p,'input',2,'mf1','trimf',  [-0.4439 -0.2604 -0.07694] );
p=addmf(p,'input',2,'mf2','trimf', [-0.2145 -0.0311 0.1523]  );
p=addmf(p,'input',2,'mf3','trimf', [0.01358 0.197 0.3804]);
figure(3)
plotmf(p,'input',2)

%pridanie vystupu u
p=addvar(p,'output','u',[-10.5225 10.8079]);   % [vstupny rozsah]
% pridanie funkcii prislusnosti pre vystup u
p=addmf(p,'output',1,'mf1','trimf', [-19.05 -10.52 -1.991]);
p=addmf(p,'output',1,'mf2','trimf',[-8.388 0.1427 8.676] );
p=addmf(p,'output',1,'mf3','trimf',[2.162 10.7 19.23])
figure(4)
plotmf(p,'output',1)


showfis(p)

% pridanie bazy pravidiel
% [vstupy | vystupy | vaha | spojenie]
% spojenie: 1 = and, 2= or
rulelist=[1 1 1 1 1;1 2 1 1 1;1 3 1 1 1;
          2 1 2 1 2;2 2 2 1 1;2 3 2 1 1;
          3 1 3 1 1;3 2 3 1 1;3 3 3 1 1];
fismat=addrule(p,rulelist)
showrule(p)

open_system('schemafuzzyanim')
sim('schemafuzzyanim',250);
```

```
t=fuzzy_riadenie(:,1);
w=fuzzy_riadenie(:,4);
y=fuzzy_riadenie(:,2);
figure(5)
plot(t,w,t,y)
xlabel('cas simulacie (min)')
ylabel('priebeh riadenia y(dm)')
title('Riadenie pomocou fuzzy regulatora mamdaniho typu')
legend('ziadana vyska hladina','priebeh riadenia',4)
bdclose('schemafuzzyanim')
```

Príloha 2

Súbor **neuron.m** na modelovanie neurónovej siete

```
clc
clear

% zadane konstanty
k11=1.25;
k22=2;
k33=1.5;
F=4;
qv1=1.5;
qv2=0.5;
qv3=0.25;

% vypocet ustalenych vysok hladin
h3s=((qv1+qv3)/k33)^2;
h2s=(qv1/k22)^2;
h1s=(qv1/k11)^2+h2s;

hs=[h1s h2s h3s]

% vypocet konstant prietokomera
k1=k11/(2*sqrt(h1s-h2s))
k2=k22/(2*sqrt(h2s))
k3=k33/(2*sqrt(h3s))

% matice stavoveho opisu
a=[-k1/F k1/F 0;k1/F -k1/F-k2/F 0;0 k2/F -k3/F]
b=[1/F 0;0 0;0 1/F]
c=[0 0 1]
d=[0 0;]

% prenosy
[cit1,men1]=ss2tf(a,b,c,d,1)
[cit2,men2]=ss2tf(a,b,c,d,2)


open_system('ustalene_hladiny')
sim('ustalene_hladiny',200)
figure(1)
plot(data(:,1),data(:,2))
legend('prechodova charakteristika',4)
title('PCH nelinearneho modelu')
xlabel('cas (min)'), ylabel('vyska hladiny v stvrtom zasobniku (dm)')

figure(2)
%Vstupne data
A = data(:,1);
u=A';
B = data(:,2);
y=B';
plot(u,y,'x');
```

```
[W1,b1,W2,b2] = initff(u,4,'logsig',y,'purelin');

df = 50;   % frekvence zobrazování kroku
me = 5000; % maximální pocet kroku ucení
eg = 0.001; % cílová kvadratická chyba
lr = 0.01; % ucící konstanta

tp = [df me eg lr];

[W1,b1,W2,b2,ep,tr] = trainbp(W1,b1,'logsig',W2,b2,'purelin',u,y,tp);

x=0:200;
yy = simuff(x,W1,b1,'logsig',W2,b2,'purelin');
figure(3)
plot(u,y,x,yy)
```