SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE
Radlinského 9, 812 37 Bratislava
KATEDRA INFORMATIZÁCIE A RIADENIA PROCESOV

Bc.Lucia PASTOREKOVÁ

DIPLOMOVÁ PRÁCA

RIADENIE SYSTÉMOV NA BÁZE UMELÝCH NEURÓNOVÝCH SIETÍ

Vedúci diplomovej práce
Doc.Ing. Alojz Mészáros, PhD.

Bratislava, 2005

Katedra: **informatizácie a riadenia procesov**

Číslo: 6/KIRP/2005

Vec: **Zadanie diplomovej práce.**

Meno a priezvisko študentky: **Bc. Lucia Pastoreková**

Meno a priezvisko vedúceho diplomovej práce**: Doc. Ing. Alojz Mészáros, CSc.**

Meno a priezvisko konzultanta diplomovej práce**: Ing. Ľubomír Šperka**

Názov diplomovej práce:

Riadenie systémov na báze umelých neurónových sietí

1. Preštudovať problematiku umelých neurónových sietí a ich možné využitie v identifikácii a riadení procesov
2. Metodicky spracovat postup a možnosti aplikácie Neural Network toolboxu v prostredí MATLAB
3. Navrhnúť neurónový regulátor na základe inverzného modelu procesu využitím Neural Network Matlab toolboxu
4. Odskúšať fuzzy nastavenie integračného člena

Termín odovzdania diplomovej práce: **21. mája 2005**

Diplomová práca sa odovzdáva v 3 exemplároch vedúcemu katedry.

Bratislava **1. februára 2005**

Doc. Dr. Ing. Miroslav Fikar
vedúci katedry

Prof. Ing. Dušan Bakoš, DrSc.
dekan

Vyhlásenie :

Vyhlasujem, že som diplomovú prácu vypracovala samostatne podľa pokynov vedúceho a konzultanta diplomovej práce, s použitím literatúry, ktorú uvádzam v osobitnom zozname.

V Bratislave, 20.05.2005 .....

Poďakovanie

Chcem sa poďakovať vedúcemu diplomovej práce Doc.Ing. A. Mészárosovi,PhD., za jeho smerovanie v mojej práci a pripomienky. ktoré mi pomohli lepšie zvládnuť danú problematiku.

Chcem sa tiež poďakovať Ing. Ľ. Šperkovi za jeho čas a odborné rady.

ABSTRAKT

Táto diplomová práca sa zaoberá možnosťami využitia umelých neurónových sietí (UNS) pri identifikácii a riadení systémov. Využitie UNS pri identifikácii systémov je podmienené vytvorením dopredného neurónového modelu systému, kým ich využitie v úlohe spätnoväzbového neurónového regulátora je spojené s vytvorením inverzného modelu riadeného systému. Vizualizácia a trénovanie jednotlivých UNS použitých v práci sa realizovala pomocou Neural Network Toolboxu (NNT) v prostredí MATLAB. Súčasťou práce je aj stručný popis ako pracovať s grafickým užívateľským rozrhaním (GUI) toolboxu.

OBSAH

ÚVOD

Vedecká oblasť Umelých Neurónových Sietí (UNS) vznikla ako pokus modelovať biologický mozog a jeho činnosti. Využitie UNS vychádza z ich základných vlastností, napr. viacvrstvové dopredné neurónové siete, ktoré sa učia zo skúsenosti metódou spätného šírenia chýb, sú schopné klasifikovať nelineárne separovatelné problémy a v spojitej oblasti sú univerzálnymi aproximátormi funkcií [1]. V princípe sú UNS schopné spočítať ľubovoľnú spočítateľnú funkciu.

V praxi sa najviac využívajú na identifikáciu systémov, klasifikáciu systémov do tried, funkčnú aproximáciu, predikciu alebo mapovanie funkcií pri použití množstva trénovacích dát, kde systémy s jasne stanovenými pravidlami zlyhávajú. Výhodou UNS je aj to, že sú tolerantné k neurčitostiam v trénovacích dátach. V oblasti riadenia a identifikácie procesov je výhodné ich použitie u zložitých nelineárnych systémov, akými sú aj systémy a procesy využívané v chemickom priemysle. Ich zložitú dynamiku je možné identifikovať, klasifikovať a predpovedať pomocou použitia UNS na zostavenie empirických dynamických modelov a tieto modely je možné následne použiť ako náhradný model pre kontrolu a/alebo optimalizáciu procesu. Nachádzajú stále širšie uplatnenie v oblasti riadenia, vykazujú dobré regulačné vlastnosti tam kde tradičné metódy návrhu regulačných systémov zlyhávajú.

Využitie UNS na identifikáciu a riadenie systémov je aj jednou z úloh tejto práce. Identifikácia systémov na báze UNS je spojená s vytvorením dopredného modelu systému, využitie neurónových sietí v úlohe spätnoväzbového neurónového regulátora je spojené s vytvorením inverzného modelu systému. Kvalitne natrénovaná sieť predstavujúca dopredný model systému, by mala byť schopná dobre opisovať dynamiku systému aj mimo trénovacie dáta.

Neurónový regulátor predstavujúci inverzný model systému, ak je dobre navrhnutý a natrénovaný, je schopný k žiadanému výstupu zo systému určiť takú hodnotu vstupnej veličiny, ktorá tento výstup vyvolá. Navrhnutý regulátor a jeho robustnosť sú následne testované najprv pri riadení nominálneho systému a to lineárneho systému 2. rádu a nelineárneho systému bioprocesu, následne sú testované pri riadení perturbovaných systémov oboch procesov.

V prípade riadenia kde už natrénovaný regulátor nedokáže zabezpečiť riadenie bez trvalej regulačnej odchýlky, sa skúmajú možnosti rozšírenia (adaptácie) inverzného

neurónového regulátora, tak aby predstavoval presnú inverziu doprednej dynamiky jednotlivých systémov. Takto získaný robustný neurónový regulátor sa adaptuje cez prahový neurón na základe hodnoty trvalej regulačnej odchýlky. Ako prídavný adaptér sa používa jednoduchý integračný člen, vlastnosti získaného regulátora sa otestujú na riadení nelineárneho modelu bioprocesu. Ďalším typom regulátora, ktorého vlastnosti sa testovali na riadení nelineárneho procesu, je fuzzy-neurónový regulátor.

Práca je rozdelená na Teoretickú časť a Experimentálnu časť.

Teoretická časť:

kapitola *Umelé neurónové siete* sa zaoberá stručným opisom základných častí UNS a ich štruktúry. Zavádzajú sa pojmy využívané v ďalších kapitolách.

kapitola *Využitie UNS v identifikácii a riadení systémov* hovorí o možnostiach využitia UNS na identifikáciu systémov, získania ich dopredného modelu a o využití inverzného neurónového modelu systému v úlohe neurónového regulátora.

kapitola *Neural Network Toolbox v prostredí MATLAB* je jednoduchým manuálom k práci s Neural Network Toolboxom v prostredí MATLAB, ktorý poskytuje nástroje pre vytváranie, vizualizáciu, realizáciu a simuláciu rôznych typov UNS.

Experimentálna časť: na príklade lineárneho systému 2.rádu a nelineárneho modelu bioprocesu sa testujú vlastnosti UNS pri identifikácii systémov a riadení nominálneho aj perturbovaného systému oboch modelov, testuje sa robustnosť navrhnutých regulátorov a možnosti ich adaptácie a rozšírenia.

1 TEORETICKÁ ČASŤ

1.1 Umelé neurónové siete

Podľa [2]: je umelá neurónová sieť (UNS) masívne paralelný procesor, ktorý má sklon k uchovávaniu experimentálnych znalostí a ich ďalšieho využívania.
Napodobňuje ľudský mozog v dvoch aspektoch :

- poznatky sú zbierané v neurónovej sieti (NN) počas učenia
- medzineurónové spojenia sú využívané na ukladanie znalostí

UNS sa skladajú z jednotlivých jednoduchých navzájom poprepájaných prvkov (neurónov) pracujúcich súčasne. Tieto prvky a teda samotné UNS boli inšpirované biologickým nervovým systémom, a tak ako aj v prírode, je ich funkcia do značnej miery určená jednotlivými spojeniami medzi neurónmi. Pomocou nastavovania týchto spojení (synaptických váh) a spojení siete s vonkajším okolím (prah neurónu bias) možno sieť trénovať a určovať jej vlastnosti tak, aby vykonávala určité funkcie.

1.1.1 História vývoja UNS

V tejto kapitole venovanej histórii UNS je stručný prehľad udalostí, ktoré najviac ovplyvnili ich rozvoj.

- 1943 Začína éra teórie UNS 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.
- 1948 Wiener vo svojej knihe Kybernetika naznačuje určité koncepty UNS.
- 1949 Hebb vo svojej knihe The Organization of Behavior prvýkrát explicitne spomína pojem učenia a jeho vzťah k synaptickým váham a ich modifikácii.
- 1952 Dr. Ashby napísal knihu Design of a Brain: The Origin of Adaptive Behavior. Táto publikácia mala zásadný význam pre rozvoj UNS.
- 1954 Dr. Minsky napísal svoju Ph.D. dizertáciu na tému Neurónové siete a neskoršie v roku 1961 napísal zásadný článok Step Toward Artificial Intelligence.
- 1956 páni Rochester, Holland, Habit a Duda sa prvýkrát pokúsili o počítačovú simuláciu UNS.

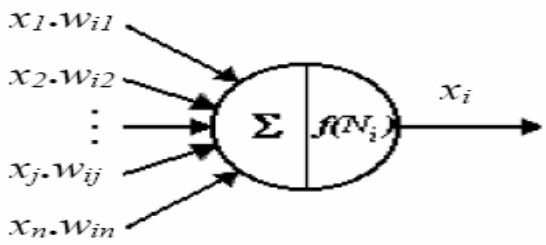
- 1958 Dr. Rosenblat prichádza s novým prístupom k rozpoznávaniu pomocou tzv. perceptrónu a neskôr so svojou konvergenčnou teóriou perceptrónu, ktorá predstavuje počiatky neurodynamiky.
- 1960 Widrow a Hoff prichádzajú s tzv. Adaline – Adptive linear element a o dva roky nato prichádza Widrow s tzv. Madaline – Multiple adaptive element. Tieto príspevky do značnej miery posunuli teoretickú bázu UNS dopredu, hoci nedostatok výpočtovej techniky vytváral ohromné zábrany ďalšiemu rozvoju.
- 1967 Dr. Cowan predstavuje svoju „sigmoidálnu“ aktivačnú funkciu.
- 1968 Dr. Grossberg predstavuje svoj adaptívny model neurónu a používa nelineárne diferenciálne rovnice na jeho popis so zámerom ich použitia pre tzv. short term memory.
- 1969 Dr. Minsky a dr. Papert popisujú činnosť viacvrstvového perceptrónu.
- 1970-80 sa nazýva Decade of Dormacy – obdobie útlmu. Dôvodom boli nedostatočné výpočtové kapacity vrátane pamäťových možností. Určité práce v teoretickej oblasti boli urobené ale nie s takou dynamikou ako predtým.
- 1975 Dr. Little a Shaw popisujú pravdepodobnostný model neurónu.
- 1980 Grossberg prichádza s rozvojom tzv. Competitiv learning, ktorá po rozpracovaní a modifikácii zakladá novú triedu UNS založenej na tzv. Adaptive resonance theory.
- 1982 Dr. Hopfield použil termín energie UNS pre pochopenie rekurentných sietí. Postupne rozvojom vzniká aj trieda tzv. Hopfieldových sietí. O rok neskôr v podstate formuloval princíp pamäte resp. uchovávania informácie v dynamickom systéme.
- 1983 Dr. Kirkpatrick a jeho kolegovia popisujú procedúru tzv. simulovaného ochladzovania. Toto inšpirovalo v roku 1985 dr. Hintona a kol. k návrhu stochastickej učiacej procedúry pre tzv. Bolzmanov stroj. V tom istom roku prišli páni Barto a kol. s tzv. reinforcement learning a jeho aplikáciou v oblasti riadenia technologických systémov.
- 1986 Dr. Rumelhardt s kolektívom prišli s metódou Backpropagation of Error – učenia spätným šírením chyby. Táto metóda pre svoju relatívnu jednoduchosť je jednou z najrozšírenejších metód učenia NN.
- 1987 p. Hecht-Nielsen ako prvý ukázal, že trojvrstvové neurónové siete s dopredným šírením a s dostatočným počtom skrytých neurónov sú schopné aproximovať s požadovanou presnosťou každé spojité zobrazenie.

- 1988 Páni Broomhead a Lowe prišli 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.

1.1.2 Štruktúra neurónu a neurónovej siete

Základným prvkom umelých neurónových sietí je *neurón*. Jeho štruktúra je uvedená na obr.1.1. Pozostáva z týchto základných častí [2]:

- vstup do neurónu
- prah neurónu *(bias)* je hodnota θ_i , ktorá prispieva ku vstupu z externého sveta
- aktivačná funkcia neurónu f , ktorej výsledkom je x_i
- výstupná funkcia neurónu o_i (zvyčajne býva identická s f)
- synaptické váhy w_{ij} *(weights)*, ktoré sú na orientovaných prepojeniach medzi neurónmi (synaptických spojeniach, pričom index i označuje postsynaptický - cieľový a index j predsynaptický - zdrojový neurón)



Obr. 1.1 Štruktúra neurónu

Vstup do i-tého neurónu je väčšinou sumačnou funkciou jednotlivých vstupov prichádzajúcich od predsynaptických neurónov ohodnotených váhovými koeficientmi. K vstupu sa ďalej pripočítava prahový koeficient predstavujúci vstup do neurónu z vonkajšieho sveta. Určuje hodnotu vstupného váženého súčtu, pri ktorom je výstup neurónu najviac citlivý na zmenu tejto sumy (z formálnych dôvodov je niekedy označovaný ako nultý vstup x_0 s hodnotou 1 a s váhou $w_0 = 0$). Pre vstup do i-tého neurónu, ktorý má N predsynaptických neurónov možno teda napísať :

$$N_i = \sum_{j=1}^{N} x_j w_{ij} + \theta_i \qquad (1.1)$$

kde w_{ij} sú váhové koeficienty (synaptické váhy), x_j aktivity neurónov predchádzajúcej vrstvy, θ_i prahový koeficient. Synaptické váhy ovplyvňujú celú sieť tým, že vplývajú na vstupy do jednotlivých neurónov a tým aj na ich stavy.

Aktivitu (výstup) *i*-tého neurónu určuje aktivačná funkcia, ktorá je funkciou vstupu do neurónu :

$$x_i = f(N_i) \tag{1.2}$$

Sú známe aktivačné funkcie rôznych tvarov [2], najbežnejšie používaná je sigmoidálna funkcia v tvare :

$$f(N_i) = \frac{A + B.e^{-N_i}}{1 + e^{-N_i}} \tag{1.3}$$

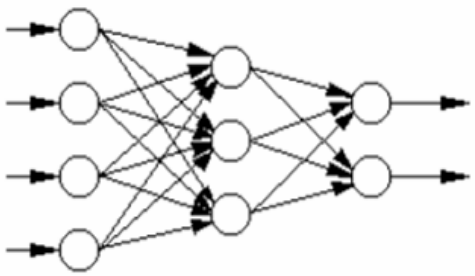
Štruktúru neurónovej siete by sme vo všeobecnosti mohli popísať ľubovoľným orientovaným grafom pomocou vrcholov (neurónov) a orientovaných hrán (prepojenia). Vlastnosti takýchto sietí sa však ťažko analyzujú, preto sa využívajú siete s pravidelnou štruktúrou. Medzi tie patria aj viacvrstvové štruktúry, kde rozlišujeme nasledujúce vrstvy :

- *vstupná vrstva* (input layer); neuróny dostávajú vstup len z vonkajšieho sveta a výstup z nej obvykle pokračuje k ďalším vrstvám (k tzv. skrytým) neurónovej siete. Aktivitu vstupných neurónov predstavuje základná informácia, ktorá vstupuje do siete. Vstupná vrstva slúži len na oddelenie vstupného vektoru ***x*** od ostatných vrstiev siete, v tejto vrstve nedochádza k žiadnej úprave vstupného signálu.
- *skrytá vrstva* (hidden layer); neuróny dostávajú vstup z ostatných neurónov ale aj z externého sveta cez prahové prepojenia a ich výstupy ďalej pokračujú do neurónovej siete. Hodnotu aktivity skrytej vrstvy určujú aktivity vstupných neurónov spolu so synaptickými váhami na spojeniach medzi vstupnou a skrytou vrstvou (resp. pre sieť s viacerými skrytými vrstvami, je to medzi dvoma skrytými vrstvami) a tiež vstupy z externého sveta.
- *výstupná vrstva* (output layer); je podobná skrytej vrstve, ale výstup z nej je do externého sveta. Správanie výstupných neurónov je závislé na aktivitách skrytých neurónov a synaptických váhach na spojeniach skrytej a výstupnej vrstvy, spolu so vstupmi z externého sveta.

Na základe tohoto delenia rozlišujeme aj neuróny na vstupné, skryté a výstupné.

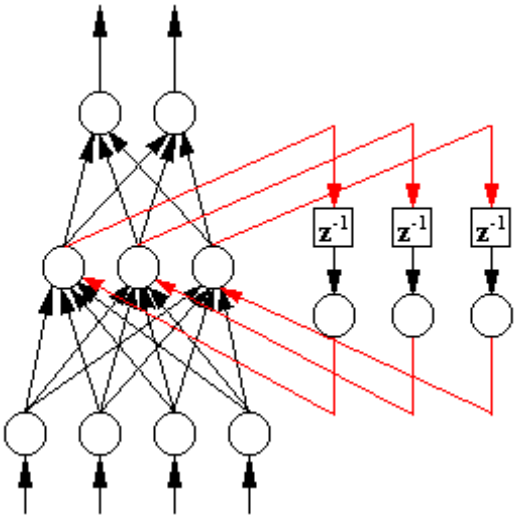
Topológia neurónových sietí sa všeobecne rozdeľuje do dvoch základných skupín:

a) dopredné neurónové siete (feed-forward NN – > FF NN) – v týchto sieťach sa signál (informácia) šíri po orientovaných synaptických prepojeniach len jedným smerom a to od najnižšej vrstvy (vstupu) k najvyššej vrstve (výstupu), teda dopredu. Neuróny v jednej vrstve nie sú navzájom poprepájané.



Obr. 1.2 Príklad doprednej neurónovej siete

b) rekurentné neurónové siete (RC NN) – pri tomto type sietí je rozdelenie vrstiev a neurónov na vstupné resp. výstupné sťažené skutočnosťou, že niektoré neuróny predstavujú vstupné a zároveň výstupné typy (platí aj pre rozdelenie vrstiev). Signál tak môže putovať cez sieť v oboch smeroch. Sú dynamické, ich stav sa mení pokiaľ nedosiahnu bod rovnováhy, následne zostávajú v tomto bode dokiaľ sa nezmení vstup a nie je nutné hľadať nový rovnovážny bod.



Obr. 1.3 Príklad čiastočne rekurentnej neurónovej siete

1.1.3 Trénovanie UNS

Vytvoriť UNS tak, aby mala určité špecifické vlastnosti, znamená zvoliť vhodnú štruktúru siete, a tiež spôsob akým budú jednotlivé vrstvy a neuróny navzájom poprepájané a poznať hodnoty synaptických váh na jednotlivých spojeniach. Tieto spojenia určujú ako sa navzájom jednotlivé neuróny ovplyvňujú a váhy nám hovoria o sile tohoto ovplyvňovania.

Každá sieť je teda nositeľom určitých vedomostí, ktoré sú uložené ako hodnoty synaptických váh. Na zmenu týchto vedomostí, je nutné podrobiť sieť procesu učenia, počas ktorého sa budú meniť parametre UNS (synaptické váhy) na základe určitých pravidiel. Charakter týchto pravidiel určuje typ učenia UNS. Po ukončení učenia bude potom sieť nositeľom nových znalostí získaných počas učenia, uložených do hodnôt synaptických váh. Vo všeobecnosti sa prístupy k učeniu rozdeľujú do dvoch veľkých skupín a to [2]:

- kontrolované učenie (učenie s učiteľom): je podmienené prítomnosťou učiteľa v celom procese učenia. Je teda potrebné UNS v procese učenia ponúknuť vstup do siete a k nemu prislúchajúci výstup. Rozdeľuje sa na :
  - štrukturálne učenie
  - temporálne učenie

  Štrukturálne učenie možno ďalej rozdeliť na autoasociačné (vzorka na vstupe do siete a na výstupe z nej sú rovnaké) a heteroasociačné (učí UNS, že ku vstupu A patrí výstup B, UNS sa teda učí rozpoznávať vstupy a zatrieďovať ich tak ako jej to povie učiteľ).

  Temporálne učenie je forma heteroasociatívneho učenia, ale na vstup musí prísť za čas t sekvencia vstupov a až tejto sekvencii ako celku bude priradený jeden výstup.
- nekontrolované učenie (učenie bez učiteľa): Pri tomto type učenia nepoznáme výstupy, ide o samo-organizujúce siete, lebo sieť sama spracuje a určuje výstup

V roku 1986 Dr.Rumelhardt so spolupracovníkmi navrhli jednoduchý algoritmus adaptácie váhových koeficientov viacvrstvových sietí s dopredným šírením s nelineárnymi neurónmi – metódu nazvanú back-propagation (BP) alebo metóda spätného šírenia chýb. Názov metódy vyplýva zo spôsobu výpočtu parciálnych

derivácií, prebiehajúci rekurentne od najvyššej k najnižšej vrstve (tj. proti smeru šírenia informácie v UNS). Uvedená metóda vznikla zovšeobecnením Widrow-Hoffovho učiaceho pravidla pre viacvrstvové UNS s nelineárnymi prenosovými funkciami, váhy a bias sa menia v smere zápornej zmeny gradientu účelovej funkcie. Cieľom metódy je naučiť sa rozpoznať a opakovať určité zákonitosti v iteračnom procese, pri ktorom sa budú nastavovať také váhy, aby sa minimalizovala zvolená nelineárna účelová funkcia:

$$E = \frac{1}{2}\left(x_0 - \tilde{x}_0\right)^2 \tag{1.4}$$

kde $\tilde{x}_0$ je odhad siete a x_0 skutočná hodnota. Daná metóda je rozpísaná v [1]:

Predpokladajme, že aktivačná funkcia je nelineárna a diferencovateľná. Parciálnu deriváciu účelovej funkcie vzhľadom k váhovým koeficientom môžeme pomocou pravidla o derivovaní zloženej funkcie napísať v tvare:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_i} \cdot \frac{\partial x_i}{\partial w_{ij}} \tag{1.5}$$

pričom parciálnu deriváciu $\frac{\partial x_i}{\partial w_{ij}}$ spočítame z rovnice (1.2) a (1.3):

$$\frac{\partial x_i}{\partial w_{ij}} = f'\left(N_i\right) \cdot \frac{\partial N_i}{\partial w_{ij}} = f'\left(N_i\right) \cdot x_j \tag{1.6}$$

Pri použití sigmoidálnej funkcie v tvare $f(x) = \frac{1}{1+e^{-x}}$ dostaneme :

$$\frac{\partial x_i}{\partial w_{ij}} = f\left(N_i\right) \cdot \left(1 - f\left(N_i\right)\right) \cdot x_j \tag{1.7}$$

Dosadením do (1.5):

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial x_i} \cdot f\left(N_i\right) \cdot \left(1 - f\left(N_i\right)\right) \cdot x_j \tag{1.8}$$

Podobne pre parciálne derivácie účelovej funkcie vzhľadom k prahovým koeficientom platí:

$$\frac{\partial E}{\partial \theta_{ij}} = \frac{\partial E}{\partial x_i} \cdot \frac{\partial x_i}{\partial \theta_{ij}} = \frac{\partial E}{\partial x_i} \cdot f'\left(N_i\right) = \frac{\partial E}{\partial x_i} f\left(N_i\right) \cdot \left(1 - f\left(N_i\right)\right) \cdot x_j \tag{1.9}$$

Porovnaním rovníc (1.8) a (1.9) dostávame :

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \theta_i} \cdot x_j \tag{1.10}$$

Výpočet derivácie $\frac{\partial E}{\partial x_i}$ závisí od toho, či index *i* popisuje výstupný alebo skrytý neurón:

a) výpočet výstupnej vrstvy

$$\frac{\partial E}{\partial x_i} = x_i - \tilde{x}_i \tag{1.11}$$

b) výpočet skrytej vrstvy

$$\frac{\partial E}{\partial x_i} = \sum_k \frac{\partial E}{\partial x_k} \cdot \frac{\partial x_k}{\partial x_i} \tag{1.12}$$

kde sumácia beží cez všetky neuróny, ktoré nasledujú za i-tým neurónom, a teda výraz je nulový, ak by sme ho použili pre výpočet na výstupnej vrstve. Oba výsledky teda môžeme zosumarizovať a bez ohľadu na akej vrstve prebieha výpočet platí:

$$\frac{\partial E}{\partial x_i} = \left(x_i - \tilde{x}_i\right) + \sum_k \frac{\partial E}{\partial x_k} \cdot \frac{\partial x_k}{\partial x_i} \tag{1.13}$$

Parciálnu deriváciu $\frac{\partial x_k}{\partial x_i}$ vypočítame z rovníc (1.2) a (1.3):

$$\frac{\partial x_k}{\partial x_i} = f'\left(N_k \cdot \frac{\partial N_k}{\partial x_i}\right) = f'\left(N_k\right) \cdot w_{ki} \tag{1.14}$$

Dosadíme do (1.3):

$$\frac{\partial E}{\partial \theta_i} = f'\left(N_i\right) \cdot \left[\left(x_i - \tilde{x}_i\right) + \sum_k \frac{\partial E}{\partial \theta_k} f'\left(N_k\right) \cdot w_{ki}\right] \tag{1.15}$$

Úpravou s ohľadom na rovnicu (1.9) získame vzťah:

$$\frac{\partial E}{\partial \theta_i} = f'\left(N_i\right) \cdot \left[\left(x_i - \tilde{x}_i\right) + \sum_k \frac{\partial E}{\partial \theta_k} \cdot w_{ki}\right] \tag{1.16}$$

Výpočet parciálnej derivácie účelovej funkcie vzhľadom k váhovým a prahovým koeficientom sa realizuje následovne:

1. vypočítajú sa parciálne derivácie $\frac{\partial E}{\partial \theta_i}$ pre výstupné neuróny (výstupnú vrstvu označme *n*)

2. z rovnice (1.16) spočítame parciálne derivácie $\frac{\partial E}{\partial \theta_i}$ pre neuróny z predposlednej vrstvy
3. pokračujeme výpočtom $\frac{\partial E}{\partial \theta_i}$ na ďalších vrstvách, pričom musí platiť, že pri výpočte $\frac{\partial E}{\partial \theta_i}$ na j-tej vrstve poznáme hodnoty týchto derivácií z nasledujúcich vrstiev $j+1, j+2, ...n$
4. nakoniec vypočítame parciálne derivácie pre neuróny z druhej vrstvy
5. parciálne derivácie $\frac{\partial E}{\partial w_{ij}}$ pre celú neurónovú sieť určíme jednoducho zo vzťahu (1.10)

Uvedený postup môžeme zovšeobecniť aj pre účelovú funkciu, ktorá obsahuje viac ako jeden pár vstupno-výstupných vektorov $x_I / \tilde{x}_O$. Celkový gradient účelovej funkcie vypočítame ako sumu gradientov pre všetky dvojice $x_I / \tilde{x}_O$:

$$grad \quad E = \sum_{i=1}^{r} grad \quad E^{(i)} \tag{1.17}$$

Adaptačný proces neurónovej siete je realizovaný minimalizáciou účelovej funkcie E vzhľadom k prahovým a váhovým koeficientom. Jedným z najjednoduchších a súčasne najpoužívanejším spôsobom ako realizovať minimalizáciu účelovej funkcie je metóda najprudšieho spádu (steepest descent), v ktorej váhové a prahové koeficienty sú rekurentne obnovované pomocou vzťahov :

$$w_{ij}^{(k+1)} = w_{ij}^{(k)} - \lambda \cdot \frac{\partial E}{\partial w_{ij}} + \mu \cdot \Delta w_{ij}^{(k)} \tag{1.18}$$

$$\theta_j^{(k+1)} = \theta_j^{(k)} - \lambda \cdot \frac{\partial E}{\partial \theta_j} + \mu \cdot \Delta \theta_j^{(k)} \tag{1.19}$$

kde Δw_{ij}^k je zmena váhy w_{ij}^k z predchádzajúceho iteračného kroku. Parameter λ sa volí obvykle z intervalu $\langle 0.01;0.1 \rangle$ a parameter μ z intervalu $\langle 0.5;0.7 \rangle$.

Proces trénovania :

- začína inicializáciou všetkých váhových a prahových koeficientov malým nenulovým číslom. Najčastejšie sú váhové koeficienty generované náhodne, ale používajú sa aj genetické algoritmy alebo metóda simulovaného žíhania, ktorými sa počiatočné hodnoty váh nastavia oveľa rozumnejšie.
- potom sa postupne na vstup do siete predkladá časť alebo celá trénovacia množina, pričom sa zaznamenáva chyba odhadu siete.
- nakoniec sa znova nastavia váhy, napr. spôsobom uvedeným vyššie tak, aby sa znížila chyba odhadu siete.
- tento proces sa opakuje, kým chyba neklesne po hodnotu minimálnej žiadanej chyby.

Jeden takýto cyklus (predloženie trénovacej množiny sieti –> určenie chyby odhadu siete –> obnovenie váhových koeficientov) sa nazýva epocha. Veľkosťou trénovacej množiny sa rozumie počet vzoriek vstupno-výstupných dát, ktoré boli použité pre nastavenie váh a nazýva sa veľkosť epochy.

V praxi je najpoužívanejšia metóda strednej kvadratickej chyby (MSE) na aktiváciu výstupných neurónov. Je ľahko vypočitateľná, jej parciálne derivácie podľa jednotlivých váhových koeficientov sa dajú určiť explicitne a v praxi sa dobre osvedčila. Stredná kvadratická chyba pre jedno predloženie trénovacej vzorky na vstup siete sa určí ako kvadrát odchýlky medzi hodnotou žiadaného výstupu a aktiváciou j-tého výstupného neurónu, a nakoniec sa urobí ich priemer pre všetky výstupné neuróny siete:

$$E_p = \frac{1}{n} \sum_{j=1}^{n} \left(x_{pj} - \tilde{x}_{pj} \right)^2 \tag{1.20}$$

kde n je počet výstupných neurónov, x_{pj} je žiadaný výstup j-tého neurónu a $\tilde{x}_{pj}$ je skutočný výstup neurónu. Ak veľkosť epochy je m, potom pre strednú chybu epochy platí:

$$E = \frac{1}{m} \sum_{p=1}^{m} E_p \tag{1.21}$$

Kritériom pre ukončenie trénovania siete môže byť počet epoch alebo minimálna chyba epochy, ktorá sa má dosiahnuť počas adaptácie siete.

Dobre natrénovaná sieť je následne schopná predikovať správne výstupné hodnoty aj pre neznáme vstupy. Každý nový vstup vedie k takej hodnote výstupu, ktorá je blízka hodnote výstupu prislúchajúcej k vstupu trénovacích dát, ak nový vstup je blízky vstupu z trénovacích dát. Táto vlastnosť umožňuje trénovať sieť na reprezentatívnej vzorke vstupno-výstupných dát, znamená to, že sieť bude dávať dobré výsledky bez toho, aby bola trénovaná na všetkých možných vstupno-výstupných dátach.

1.2 Využitie UNS v identifikácii a riadení systémov

1.2.1 Identifikácia systémov

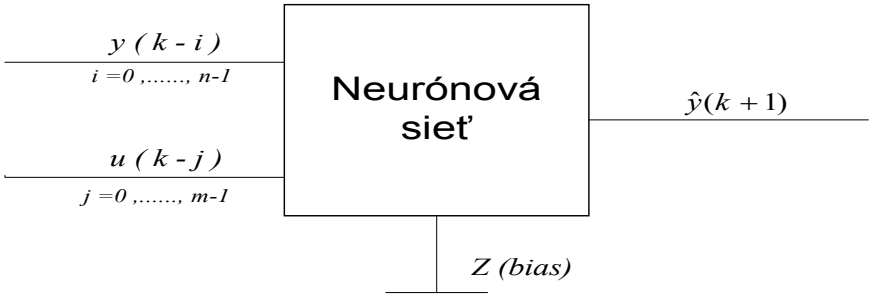
Viacvrstvové neurónové siete s dopredným šírením a dostatočným počtom skrytých neurónov sú univerzálnym aproximátorom a sú teda schopné vystihnúť s požadovanou presnosťou vstupno-výstupné správanie ľubovoľnej spojitej funkcie. Môžu preto byť chápané ako univerzálny prostriedok pre regresnú analýzu funkcií definovaných trénovacou množinou, kde tvar modelovej funkcie bude predurčený architektúrou UNS (topológiou prepojenia neurónov a nastavením váhových a prahových koeficientov na určité hodnoty). Túto vlastnosť je možné využiť pri identifikácii zložitých systémov, kde klasické metódy neuspejú.

Ak teda máme systém, ktorého vstupno-výstupné správanie je možné opísať vzťahom $y = f(u)$, kde y resp. u sú výstupy resp. vstupy systému a funkcia $f(.)$ je neznáma nelineárna funkcia, našou úlohou je zostaviť a natrénovať takú UNS, ktorá bude generovať takú funkciu $g(.)$, aby tá dostatočne dobre aproximovala neznámu funkciu $f(.)$, použitím vstupno-výstupných dát procesu.

Pri identifikácii systémov ide najmä o zachytenie *doprednej dynamiky* identifikovaných systémov, teda o získanie tzv. *dopredného modelu*. Po identifikácii systému pomocou neurónovej siete sa tá správa podľa nelineárnej diskrétnej rovnice tvaru

$$\hat{y}(k+1) = f\left[y(k), y(k-1),..., y(k-n+1), u(k), u(k-1),..., u(k-m+1)\right] \qquad (1.22)$$

kde $u(k)$ a $y(k)$ sú vstup a výstup systému v k-tom kroku výpočtu, n a m sú ich počty a $\hat{y}(k+1)$ je odhad siete do ďalšieho kroku.



Obr. 1.4 Schéma dopredného modelu

Základnými otázkami pri voľbe UNS ako univerzálneho prostriedku na identifikáciu systémov sú [1]:

- *ako zvoliť a zostaviť trénovaciu množinu a testovaciu množinu*

  Pred trénovaním siete je nutné rozdeliť získané údaje na trénovacie a testovacie dáta. Trénovacie dáta by mali popisovať celkové správanie sa systému, ktorý reprezentujú. Ak je sieť vhodne natrénovaná a výstup z nej je dostatočne blízky k nami zadaným výstupom v trénovacej množine, bude druhá mocnina chyby daná rozdielom medzi týmito dvoma výstupmi relatívne malá. Dôležitejšie však je, aby bol tento výstup zo siete vo vzťahu zo všetkými minulými a budúcimi kombináciami vstupov a výstupov. Na otestovanie tejto vlastnosti sa používajú testovacie dáta. Dobrým ukazovateľom kvality a vhodnosti natrénovanej siete je porovnanie priemernej chyby dosiahnutej pri trénovacích dátach s chybou dosiahnutou pri preskúšaní siete na testovacích dátach. Ak je tento rozdiel príliš veľký treba uvažovať nad zmenou trénovacích dát alebo štruktúry siete.

- *návrh architektúry NN*

  V prípade trojvrstvej neurónovej siete, teda typu ktorý bol používaný aj v tejto práci, máme jednu skrytú vrstvu s neznámym počtom skrytých neurónov. Na základe odvodení v [1] možno očakávať, že s rastom počtu skrytých neurónov bude adaptačný proces NN poskytovať stále lepšie výsledky (hodnota účelovej funkcie pre testovaciu množinu sa bude asymptoticky blížiť k nule). Toto tvrdenie je správne, je však tiež známe (a dokázateľné pomocou porovnávania sietí s rozličným počtom skrytých neurónov), že od určitého počtu skrytých neurónov sa predikcia siete pre testovaciu množinu bude zhoršovať. Znamená to, že ďalšie zvyšovanie počtu neurónov je už zbytočné prípadne dokonca nežiaduce.

- *počet iteračných krokov (počet epoch)*

  Hodnota účelovej funkcie definovaná pre objekty z trénovacej množiny v priebehu adaptácie klesá s rastom počtu iteračných krokov. Je však známe, že od určitého počtu iteračných krokov začne rásť hodnota účelovej funkcie pre testovaciu množinu. Znamená to, že predikčná schopnosť siete sa s pokračovaním adaptácie začne zhoršovať (hovoríme vtedy o preučenej sieti).

- *voľba vhodnej adaptačnej (minimalizačnej) funkcie*

  Už spomenutá metóda najprudšieho spádu patrí síce medzi najjednoduchšie a najpoužívanejšie metódy, jej nedostatkom je, že vyžaduje veľký počet (rádovo

tisíc) iteračných krokov a je teda veľmi pomalá. Preto sa venuje pozornosť aj rýchlejším a efektívnejším optimalizačným metódam, pomocou ktorých sa dosiahne rýchlejší proces adaptácie, obvykle za cenu zhoršenia predikčných schopností.

V tejto práci sa na identifikáciu a predikciu systému využívala trojvrstvá dopredná sieť s jednou skrytou vrstvou. Vo všeobecnosti možno jej architektúru s počtom používaných neurónov popísať ako {vstupná vrstva / počet vstupných neurónov (počet vstupov do siete), skrytá vrstva / počet skrytých neurónov, výstupná vrstva / počet výstupných neurónov }. U jednotlivých typov použitých systémov sa daná sieť líšila len počtom neurónov v skrytej vrstve.

Pri identifikácii dopredného modelu lineárneho systému postačovala sieť s 2 neurónmi v skrytej vrstve, jej štruktúru možno skrátene zapísať ako {4,2,1}. Na identifikáciu nelineárneho systému sa počet skrytých neurónov zvýšil na 3, štruktúra siete bola {4,3,1}. Pri identifikácii inverzného modelu sa ukázalo vhodné zvýšiť počet neurónov v skrytej vrstve pre oba prípady o jeden skrytý neurón, teda pre lineárny systém nadobudla sieť štruktúru {4,3,1} a pre nelineárny systém {4,4,1}.

Súbor vstupných údajov do siete sa vytvoril z dvojíc trénovacej množiny, tj. zo vstupov do systému a k nim prislúchajúcich výstupov zo systému získaných simuláciou systému. Veľký význam a pozornosť sa prikladá správnej voľbe týchto trénovacích dát tak, aby dobre popisovali celkové chovanie reprezentovaného systému. Tieto znalosti sa pri učení ukladajú do synaptických váh neurónovej siete, a tým sa predurčuje schopnosť siete správne identifikovať systém a tiež schopnosť predikcie siete na testovacích dátach. Ak by trénovacia množina nebola vhodne zvolená, bude samotná adaptácia parametrov siete nekvalitná.

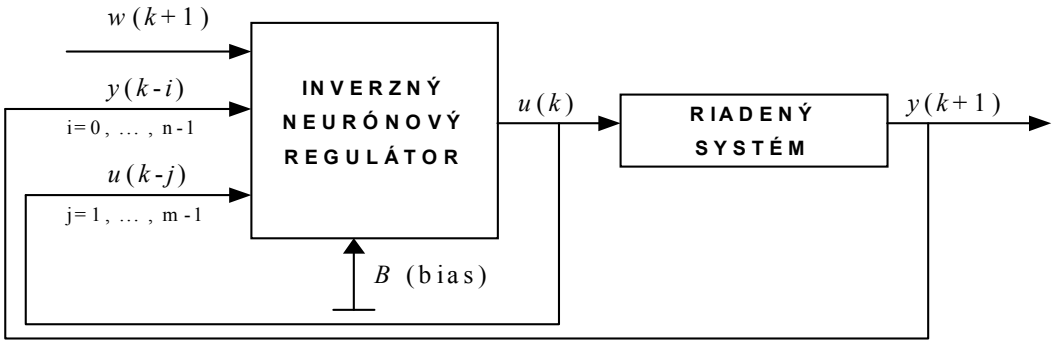
1.2.2 Riadenie na báze UNS

Použitie UNS v roli regulátora je založené na zostavení *inverzného neurónového modelu procesu*, ktorý možno následne použiť ako spätnoväzbový neurónový regulátor.

Aby sa zabezpečila dobrá regulácia, musí byť natrénovaná neurónová sieť presnou inverziou dopredného modelu systému. Pre dopredný model vyjadrený v tvare (1.22), dostávame presnú inverziu modelu vyjadrením $u(k)$ ako akčného zásahu regulátora v tvare

$$u(k) = f^{-1}\left[\hat{y}(k+1), y(k), y(k-1),..., y(k-n+1), u(k-1),..., u(k-m+1)\right] \quad (1.23)$$

Treba si však uvedomiť, že počas trénovania je predikovaná hodnota $\hat{y}(k+1)$ známa zo súboru vstupno-výstupných dát, ak sa však regulátor použije v uzatvorenom systéme URO, je táto hodnota nedostupná. Jedným zo spôsobov jej náhrady je možnosť nahradiť ju žiadanou hodnotou $w(k+1)$, čím získame z inverzného modelu spätnoväzbový neurónový regulátor. Ak teda rovnica (1.23) vyjadruje presnú inverziu modelu systému, získame inverzný neurónový regulátor.



Obr. 1.5 Schéma regulačného obvodu s priamym inverzným neurónovým regulátorom

Natrénovaná UNS predstavujúca inverzný model systému bola použitá ako priamy spätnoväzbový regulátor pre riadenie nominálneho systému. Pri lineárnom systéme sa použila UNS štruktúry {4,3,1} a pri nelineárnom systéme UNS štruktúry {4,4,1}.

1.2.3 Robustné riadenie na báze UNS

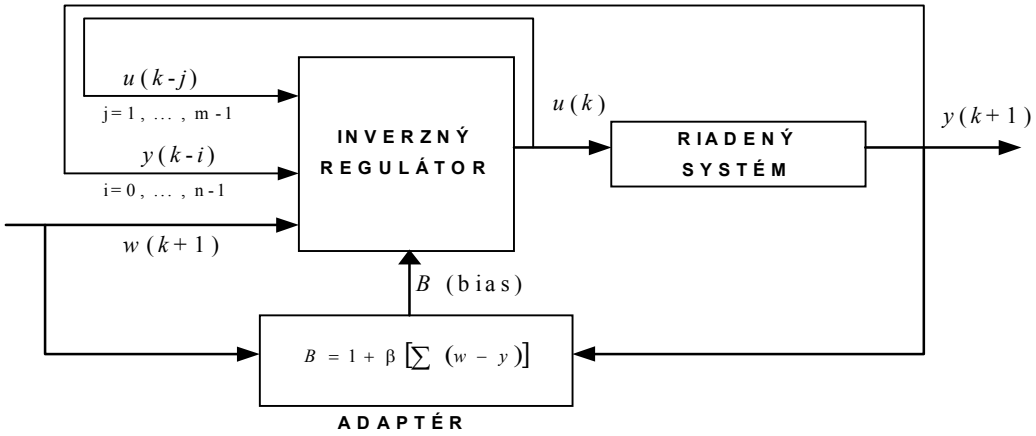
Ako už bolo povedané v kapitole 1.2.2 inverzný neurónový model systému možno použiť vo funkcii regulátora. Pretože sa však takýto model v úlohe spätnoväzbového regulátora správa podobne ako PD regulátor, uspokojivo ureguluje len nominálny systém. V prípade riadenia perturbovaných systémov by takýto regulátor zanechával trvalú regulačnú odchýlku, ktorá je nežiaduca.

Aby nedochádzalo v regulačnom pochode k zanechávaniu TRO, je nutné získaný regulátor upraviť (adaptovať) tak, aby sa získala presná inverzia systému. Výstup z neurónovej siete je možné nastavovať prostredníctvom adaptácie prahových neurónov, ktorých vstup je pri tradičných sieťach jednotkový. Keďže výstupná aktivita i-teho neurónu je definovaná rovnicami (1.1) a (1.2), je možné pomocou úpravy týchto vstupov ovplyvňovať výstup zo siete.

Pri predpoklade, že príčinou odchýliek výstupu zo systému a žiadanej hodnoty je nesprávna hodnota vstupu do prahového neurónu daného inverzného regulátora, možno potom predpokladať, že jeho správnym nastavením možno túto TRO odstrániť. Vhodný signál pre nastavenie vstupu je integrál TRO. Pokiaľ bude existovať TRO, bude sa výstup adaptéra meniť a tým sa bude inverzný neurónový regulátor adaptovať. Ak však TRO dosiahne nulovú hodnotu, adaptácia sa zastaví a neurónový regulátor bude možné pokladať za presnú inverziu dopredného modelu systému. Opísaný adaptér predstavuje *jednoduchý integračný člen*. Jeho rovnicu možno popísať ako:

$$B = 1 + \beta \cdot \sum (w - y) \tag{1.24}$$

kde B je vstup do prahového neurónu, w je žiadaná hodnota, y je výstup zo systému a β je adaptačné zosilnenie. Hodnota β sa teoreticky môže meniť v rozsahu $(-\infty, \infty)$, je však nutné dodržiavať podmienky stability pre konkrétny systém. Nulová hodnota znamená, že adaptácia neprebieha.



Obr. 1.6 Schéma adaptívneho regulačného obvodu s inverzným neurónovým regulátorom

1.2.4 Fuzzy-neurónový regulátor

V [4] bolo ukázané, že prítomnosť integračnej zložky v regulačnom obvode je nevyhnutná,. ale aj napriek tomu môže byť priebeh riadenia neuspokojivý z hľadiska doby regulácie a preregulovania. Tieto negatívne efekty vyplývajú z charakteristiky adaptéra ako jednoduchého integračného člena, ako aj z nesprávneho časovania adaptácie. Preto sa hľadajú iné spôsoby adaptácie.

Ako vhodný spôsob sa ukazuje spojenie regulátorov na báze UNS a fuzzy regulátorov. Podobne ako o UNS možno aj o fuzzy regulátoroch tvrdiť, že sú univerzálnymi aproximátormi ľubovoľnej spojitej funkcie f [9]. Jazykové fuzzy (logické) regulátory (LFLC) sú fuzzy systémy s bázou znalostí, ktoré pomocou fuzzifikácie, vhodného spôsobu odvodzovania záverov (inferencie) a tzv. defuzzifikácie priraďujú hodnotám veličín sledovaných na vstupe hodnoty veličín sledovaných na výstupe, *z matematického hľadiska vytvárajú funkcie aproximujúce neznáme závislosti medzi sledovanými výstupnými a vstupnými veličinami*.

1.2.4.1 Základy teórie fuzzy logiky

Fuzzy [10] slovo pochádzajúce z angličtiny, ktoré znamená nejasný, neostrý, neurčitý. Taká je aj fuzzy logika a také sú aj fuzzy množiny. Teória fuzzy množín a fuzzy logiky vychádza z teórie klasických množín a logiky. Kým klasická logika, ktorá pracuje len s výrokmi, ktoré sú buď pravdivé (*True* s pravdivostnou hodnotu 1)

alebo nepravdivé (*False* s pravdivostnou hodnotou 0), a môže teda nadobúdať len tieto dve hodnoty, fuzzy logika pracuje na celom intervale medzi 0 (*F*) a 1 (*T*).

Klasická logika je spojená s klasickou teóriou množín. Podľa Cantorovej teórie je klasická množina (ostrá množina) súbor ľubovolných prvkov , ktoré môžno považovať za celok. Nech $\boldsymbol{X}$ je základná množina prvkov (nazývaná aj základný priestor či univerzum) a množina A je ľubovoľná podmnožina univerza $\boldsymbol{X}$, potom v klasickom prípade možno o každom prvku $\boldsymbol{x}$ z univerza $\boldsymbol{X}$ jednoznačne rozhodnúť či prvok $\boldsymbol{x}$ patrí do množiny A alebo nie. Platí teda, že pre každý prvok $\boldsymbol{x} \in \boldsymbol{X}$, je výrok „prvok $\boldsymbol{x}$ patrí do množiny A “ buď pravdivý alebo nepravdivý. Žiadna ďalšia možnosť neexistuje, preto možno množiny v klasickej teórii charakterizovať pomocou dvojhodnotovej funkcie, ktorú nazývame indikátor množiny alebo charakteristická funkcia množiny. Pre množinu $A \subseteq \boldsymbol{X}$ je charakteristická funkcia ν definovaná ako

$$v_A: \boldsymbol{X} \to \{0,1\},\ v_A(\boldsymbol{x}) = \begin{cases} 1 & ak\ x \in A \\ 0 & ak\ x \notin A \end{cases} \tag{1.25}$$

Medzi každou podmnožinou A univerza X a jej charakteristickou funkciou je vzájomne jednoznačný vzťah, aj je daná jedna, vieme určiť druhú, z matematického hľadiska ich nemusíme rozlišovať.

Na rozdiel od teórie klasických množín, v teórii fuzzy množín môže ľubovoľný prvok $\boldsymbol{x}$ univerza $\boldsymbol{X}$ patriť do fuzzy podmnožiny F len čiastočne.

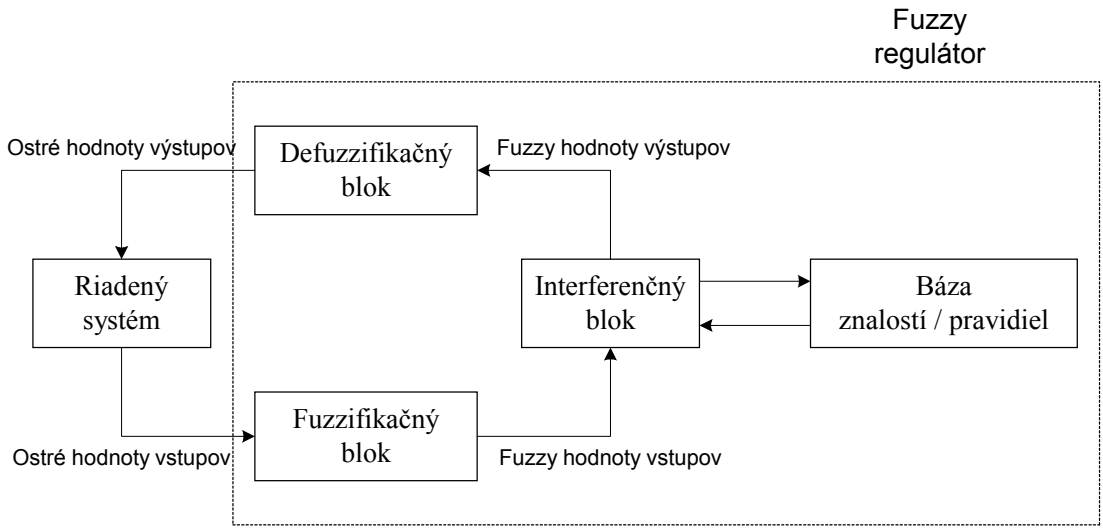
L.A.Zadeh, od ktorého pochádza základná myšlienka matematického modelovania fuzzy pojmov, zaviedol spôsob charakterizácie fuzzy pojmov pomocou funkcií definovaných na univerze $\boldsymbol{X}$, s hodnotami v intervale $\langle 0,1 \rangle$, pričom hodnota funkcie v bode $\boldsymbol{x} \in \boldsymbol{X}$ vyjadruje mieru (stupeň) sledovanej vlastnosti zodpovedajúcej prvku $\boldsymbol{x}$. Tieto funkcie sú zovšeobecnením charakteristickej funkcie ostrých množín. Pretože niektoré pojmy sa dajú dobre modelovať pomocou klasických množín, má zmysel zavádzať pojem fuzzy podmnožiny F uvažovanej základnej množiny. Pre danú podmnožinu F možno definovať *funkciu príslušnosti fuzzy podmnožiny* $\boldsymbol{v_F}: \boldsymbol{X} \to \langle 0,1 \rangle$, ktorá jednotlivým prvkom priraďuje ich mieru sledovanej vlastnosti a číslo $\boldsymbol{v_F}(\boldsymbol{x})$ potom nazývame *stupňom príslušnosti prvku do fuzzy podmnožiny F*. Hodnoty funkcie $\boldsymbol{v_F}(\boldsymbol{x})$ môžme teda interpretovať ako pravdivostné číselné hodnoty jazykových tvrdení z intervalu $\langle 0,1 \rangle$. Pri definovaní fuzzy podmnožín, si treba uvedomiť, že fuzzy

podmnožina je daná, len ak je daná jej funkcia príslušnosti, teda fuzzy podmnožina je objekt, ktorý je určený funkciou príslušnosti a možno písať $F=\{(\boldsymbol{x},\boldsymbol{v}_F(\boldsymbol{x}))\}$ (tzv. singleton)

Ďalším pojmom používaným vo fuzzy logike je *fuzzy odvodzovanie* (približné odvodzovanie). Je to proces, v ktorom sa odvodzujú závery na základe informácií sformulovaných do tzv. fuzzy AK-POTOM pravidiel, ktoré obsahujú vágne pojmy. Pri fuzzy odvodzovaní je dôležitým pojmom *jazyková premenná*. Jazyková premenná je premenná, ktorej hodnoty sú vyjadrené pomocou slov. Je charakterizovaná ako pätica (X,T(X), $\boldsymbol{X}$,G,$\mathcal{M}$), kde X je meno jazykovej premennej; T(X) je množina slovných hodnôt jazykovej premennej; $\boldsymbol{X}$ je univerzum jej hodnôt; G je gramatika, obsahujúca syntaktické pravidlá na vytváranie hodnôt jazykovej premennej; $\mathcal{M}$:T(X)→ $F(\boldsymbol{X})$ je sématicke pravidlo, ktoré každej slovnej hodnote priradí fuzzy podmnožinu univerza $\boldsymbol{X}$, čím sa definuje jej význam.

Poznatky z teórie fuzzy množín a fuzzy logiky možno využiť v už spomenutých LFLC fuzzy regulátoroch. Výhodou fuzzy regulátorov je ich relatívne jednoduchý návrh aj pre zložité systémy, pri ktorých použitie klasických regulátorov nie je možné, ďalšou výhodou je ich veľká robustnosť. Jazykové fuzzy regulátory sú znalostnými systémami, ktoré využívajú znalosti získané od operátora. Na základe skúseností sa dá zostaviť systém fuzzy pravidiel už spomínaného typu fuzzy implikácie AK (fuzzy výrok, predpoklad), POTOM (fuzzy výrok, záver), pomocou jazykových výrazov prirodzeného jazyka, popisujúcich činnosť v základných situáciách. Tento systém riadiacich pravidiel sa nazýva *Báza znalostí* alebo *Báza pravidiel*. Operátorov popis obsahuje vágne pojmy prirodzeného jazyka, čím sa vytvára priestor pre fuzzy množiny na modelovanie jazykových hodnôt sledovaných premenných, fuzzy regulátor potom na základe balíka základných vedomostí z bázy pravidiel a pravidiel fuzzy logiky napodobňuje činnosť operátora.

Zjednodušená schéma fuzzy regulátora s jednotlivými krokmi práce je na obr.1.8.



Obr.1.7 Základná bloková schéma fuzzy regulátora

Fuzzifikácia znamená pretransformovať ostrú hodnotu ***x**** premennej ***x*** na fuzzy množinu. Vstupné ostré číslo sa považuje za fuzzy číslo so špeciálnym typom funkcie príslušnosti už spomínaným singletonom a toto číslo potom patrí do fuzzy podmnožiny *F* s príslušnosťou $\boldsymbol{v}_F(\boldsymbol{x}^*)$.

Interferenčný mechanizmus je proces priraďovania výstupných fuzzy výrokov (záverov) k vstupným fuzzy výrokom (predpokladom) na základe bázy znalostí a bázy pravidiel.

Defuzzifikácia je získanie ostrej hodnoty výstupnej veličiny ***y**** z fuzzy množiny ***y*** ako výsledku interferenčného mechanizmu.

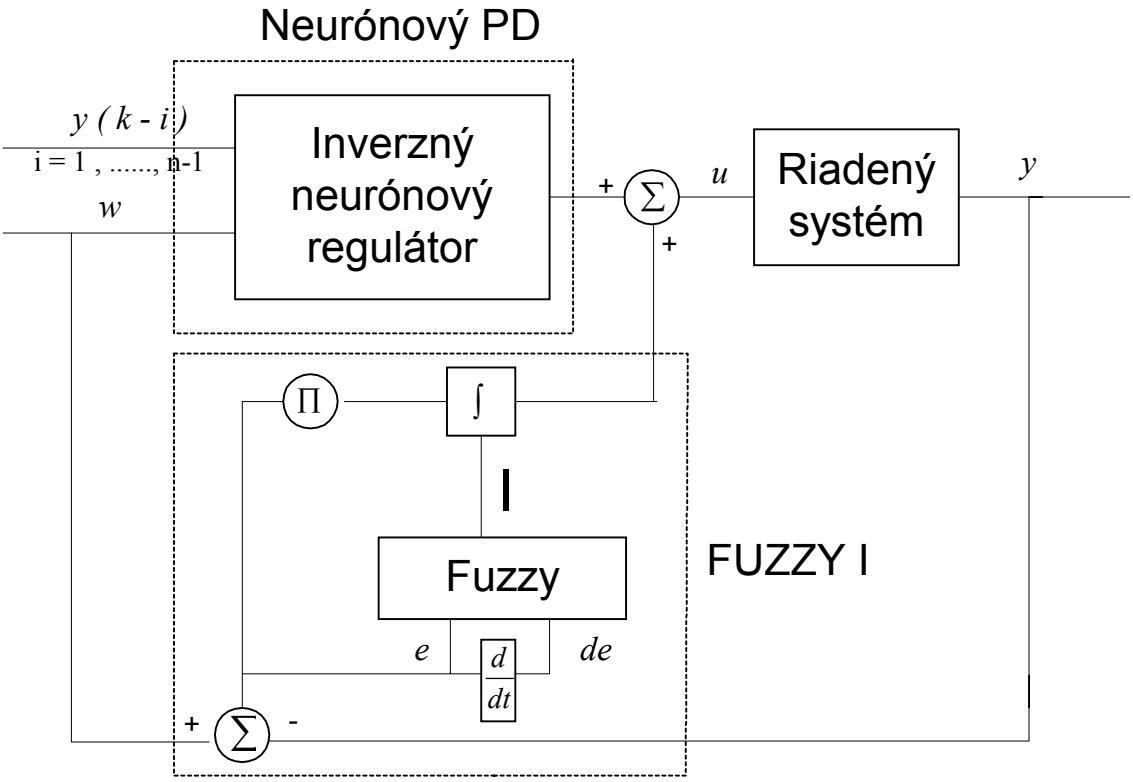
Na obr.1.7 je schéma fuzzy-neurónového regulátora zloženého z inverzného neurónového modelu systému, ktorý si možno predstaviť ako neurónový PD regulátor a fuzzy integračného člena, ktorý nahradí jednoduchý integračný člen. Riadiacu veličinu možno napísať ako

$$u = f_{nn}(w, y) + f_{fuzzy}(e, de) \tag{1.26}$$

kde člen $f_{nn}(w, y)$ je nelineárna funkcia žiadanej hodnoty a výstupu zo systému, ktorá je reprezentovaná inverzným dynamickým neurónovým modelom systému a člen $f_{fuzzy}(e, de)$ je fuzzy integračný člen reprezentovaný nelineárnou funkciou regulačnej odchýlky *e* a jej derivácie *de* v tvare

$$f_{fuzzy}(e, de) = \int_0^t (I(e(t), de(t)).e(t))dt \tag{1.27}$$

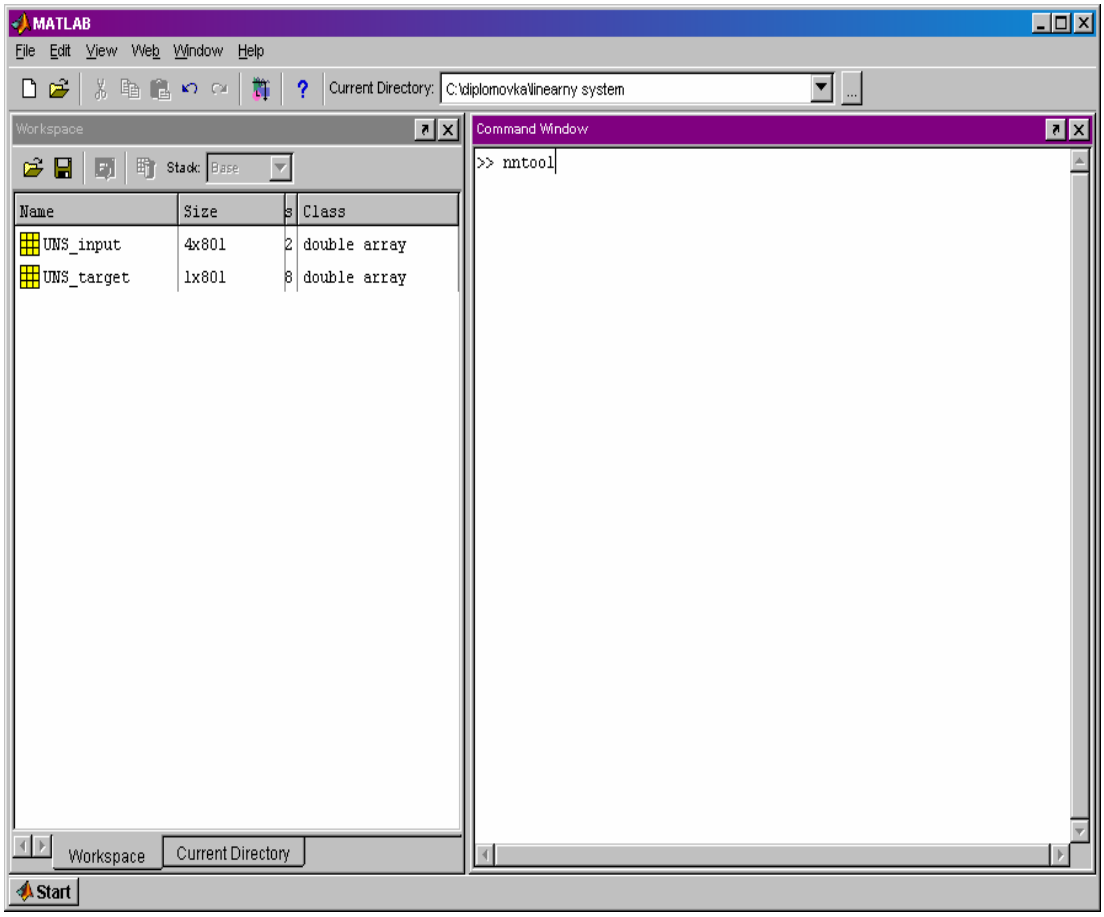
kde člen $I(e(t), de(t))$ zodpovedá výstupu z fuzzy regulátora so vstupmi *e* a *de*.



Obr. 1.8 Fuzzy - neurónový regulátor

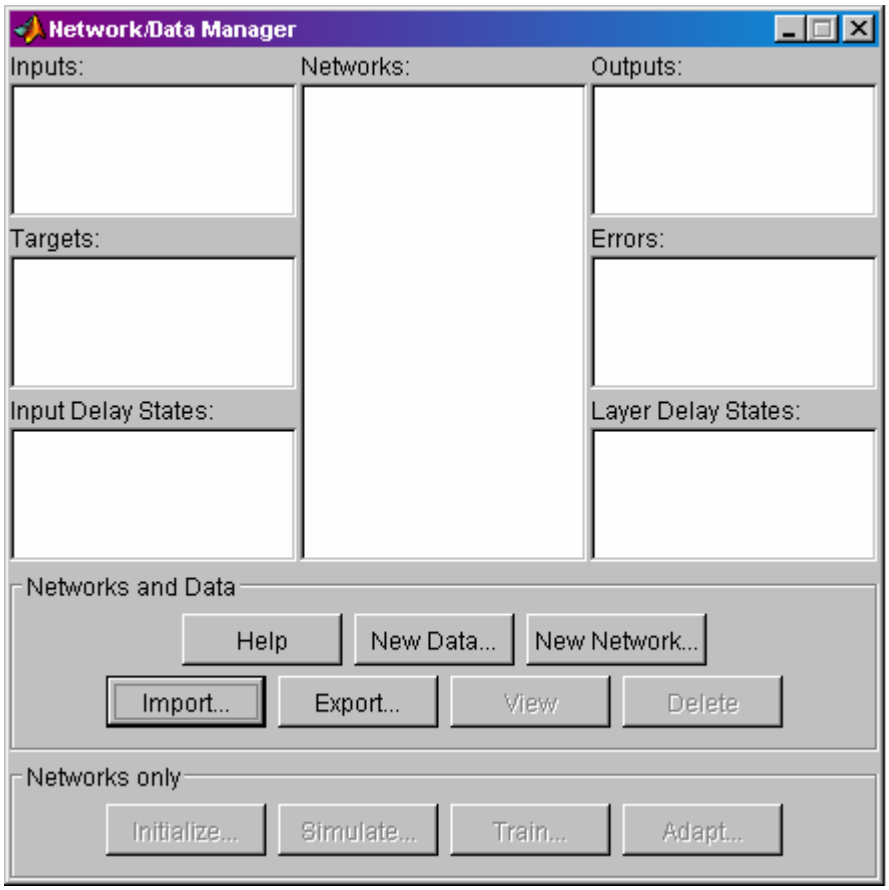
1.3 Neural Network Toolbox v prostredí MATLAB

Pri návrhu a trénovaní UNS sa využíval Neural Network Toolbox (NNT) v prostredí MATLAB. Tento toolbox rozširuje možnosti využívania prostredia MATLAB tým, že poskytuje nástroje pre vytváranie, vizualizáciu, realizáciu a simuláciu rôznych typov UNS. S Toolboxom je možné pracovať priamo v príkazovom riadku prostredia MATLAB pomocou zadefinovaných príkazov, ďalší spôsob je využitie grafického užívateľského prostredia **GUI** (Graphical User Interface). Využívanie GUI je výhodné, pretože nevyžaduje od užívateľa poznať príkazy nutné na používanie toolboxu. Výsledky získané v GUI je možné exportovať do pracovného priestoru MATLABu (okno **Workspace**), rovnako ako je možné importovať potrebné údaje z neho do GUI. Príkaz na spustenie užívateľského prostredia v príkazovom riadku (okne) MATLABu je **nntool** (obr.1.9).



Obr.1.9 Užívateľské prostredie MATLAB

Objaví sa okno **Network/Data Manager** (obr.1.10). Toto okno má vlastnú pracovnú oblasť, pracuje tak mimo príkazového riadku MATLABu. V tomto okne je možné importovať údaje z Workspace, vytvoriť a vidieť sieť, trénovať ju, simulovať a následne exportovať výsledky do Workspace.



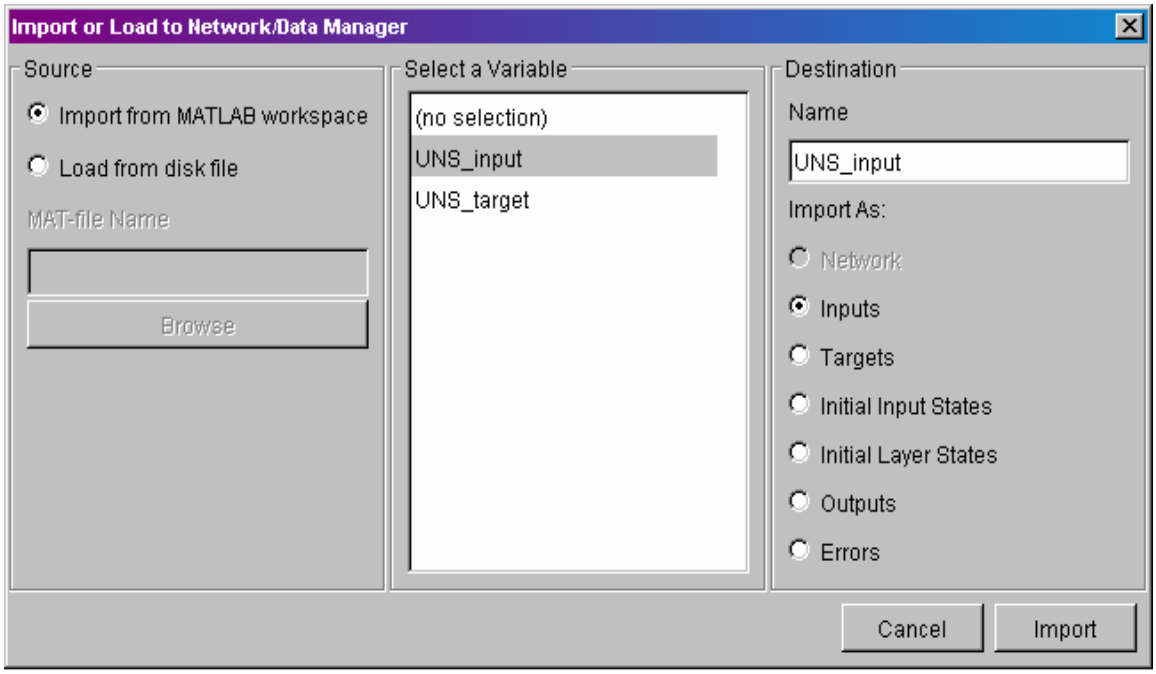
Obr.1.10 Grafické prostredie Neural Network Toolbox

Ďalšie kroky budú viesť k vytvoreniu doprednej siete s jednou skrytou vrstvou štruktúry {4,2,1}, následne použitej na identifikáciu lineárneho systému. Simuláciou modelu systému sa získala trénovacia množina obsahujúca 750 párov vstupno-výstupných údajov opisujúcich systém. Na vstup do systému sa privádzali náhodne generované skokové zmeny, ktoré sa spolu s odozvou systému a tiež s ich časovými posunmi zaznamenávali a získal sa tak súbor údajov obsahujúci : aktuálnu hodnotu vstupu do modelu systému $u(k)$, jeho prvú históriu $u(k-1)$, aktuálny výstup $y(k)$ a jeho prvú históriu $y(k-1)$ a výstup $y(k+1)$ ako odozvu na vstup $u(k)$. Tento súbor sa následne rozdelil na 2 súbory a to na vstup do siete UNS_input a na k nemu zodpovedajúci želaný výstup UNS_target. Vstup do siete tvorila množina hodnôt

prvých 4 premenných (matica rozmeru počet premenných x počet údajov, teda 4x750), výstup obsahoval hodnoty výstupu zo siete $y(k+1)$ (stĺpcový vektor 1x750).

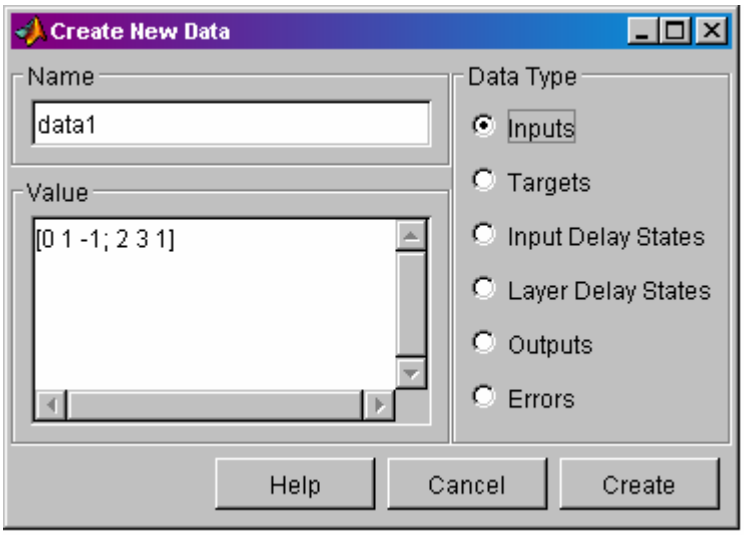
Z Workspace prostredia MATLAB sa tieto údaje importujú do GUI pomocou voľby IMPORT. Dostávame tak okno **Import or Load** to Network/Data Manager (obr.1.11).

Tu je možné vybrať si miesto odkiaľ chceme importovať, označiť žiadanú premennú, zmeniť jej meno a vybrať miesto importovania (Import As). Ak potrebné údaje nie sú v aktuálnom MATLAB/Workspace, možno voľbou Load from Disk zvoliť ľubovoľný MAT-file, z ktorého sa údaje načítajú a ďalší postup je už rovnaký ako pre Workspace.

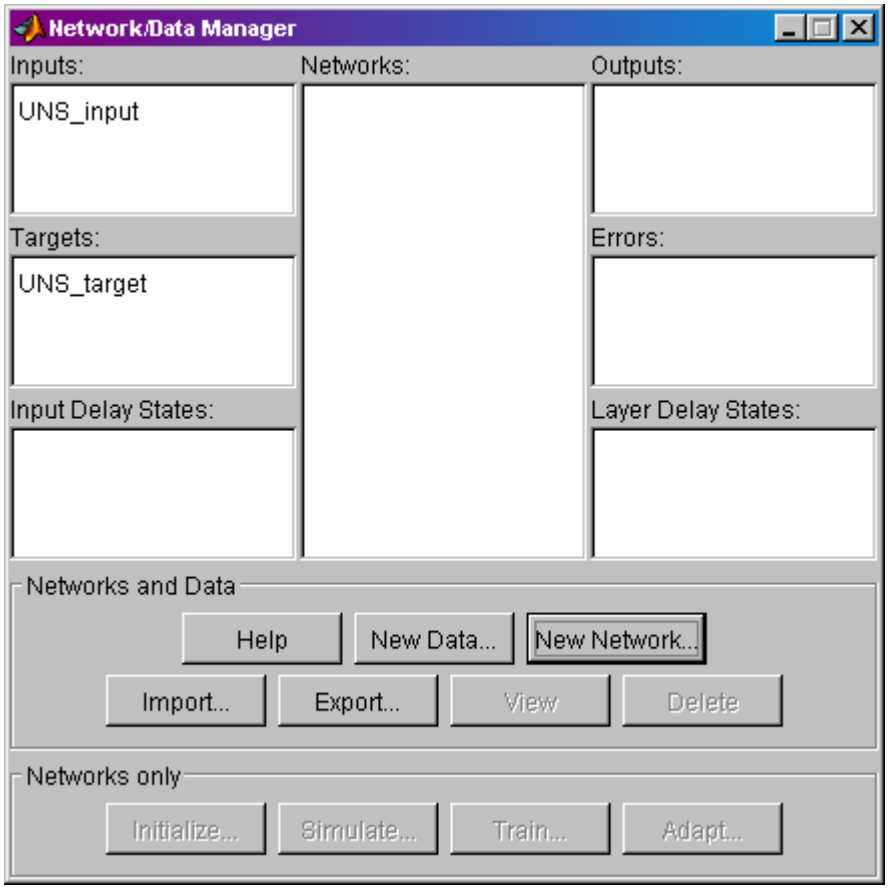


Obr.1.11 Okno pre importovanie dát do NNT

Ďalší spôsob zadávania vstupných údajov je pomocou voľby NEW DATA. V novom okne **Create New Data** (obr.1.12) je následne možné zadávať hodnoty vstupov a výstupov, ľubovoľne si ich pomenovať, zvoliť umiestnenie dát (napr. ako vstup = Inputs) a následne voľbou CREATE vytvoriť súbor dát a importovať ho do **Network/Data Manager**a. Táto možnosť je však vhodná len ak je počet vstupno-výstupných párov údajov malý. Výsledok naimportovania údajov pomocou voľby IMPORT je na obr.1.13.



Obr.1.12 Okno pre vytvorenie nových dát

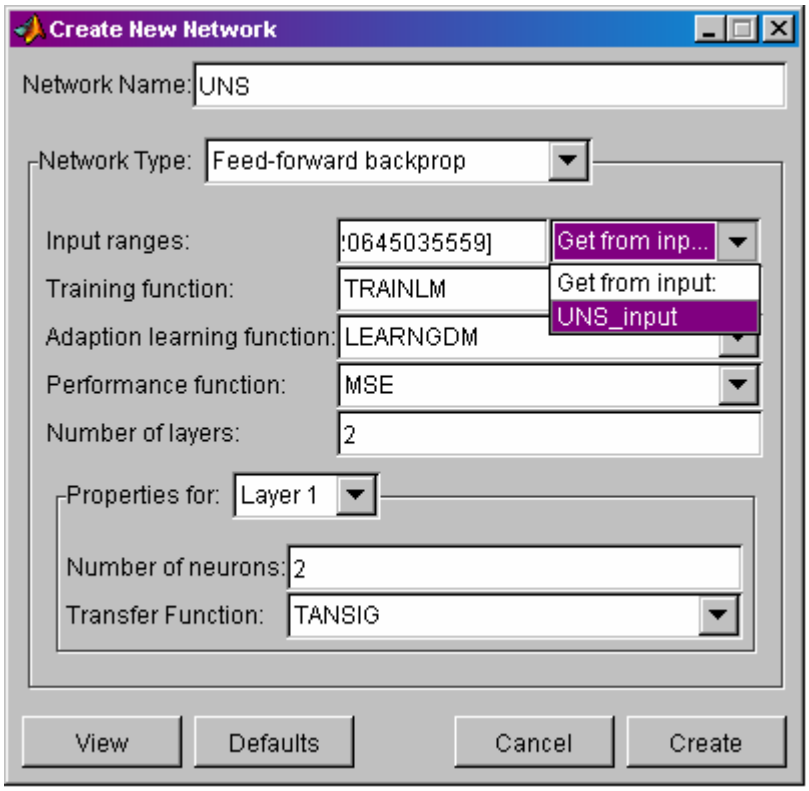


Obr.1.13 Network/Data Manager po naimportovaní vstupno-výstupných dát

Ďalším krokom je vytvorenie siete. Voľbou NEW NETWORK dostávame okno **Create New Network** (obr.1.14). V tomto okne sa definuje štruktúra siete a všetky funkcie ktoré sieť využíva. Ako prvé je možné zadať názov siete. Naša sieť sa volá

UNS. Ďalším krokom je voľba typu siete (*NETWORK TYPE*), v tomto prípade ide o doprednú sieť, ktorá využíva pri adaptácii váhových koeficientov metódu spätného šírenia chýb (*Feed-forward backprop*).

Rozsah vstupných údajov sa zadáva v položke *INPUT RANGES*, kde ho možno zadať zapísaním konkrétnych číselných hodnôt (tu je však nutné poznať formát zadávania a tiež rozsah hodnôt každej premennej). Pre 4 vstupné veličiny je to matica v tvare 4x2, t.j. každý z riadkov zodpovedá jednej premennej. Jednoduchšie je však použiť Get from input a zvoliť si náš vstupný súbor UNS_input (sám si následne vygeneruje hranice hodnôt každej vstupnej veličiny).



Obr.1.14 Okno pre vytvorenie novej siete.

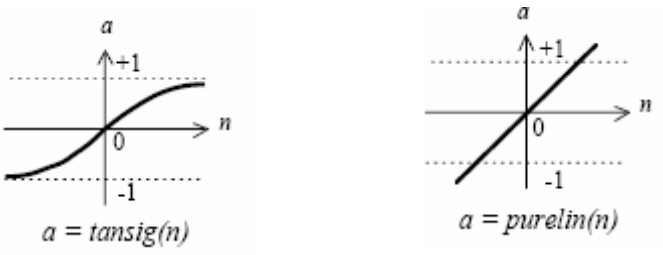
Následne sa volia funkcie, ktoré bude sieť využívať a ktoré jej budú určovať spôsob učenia. Prvá je *TRAINING FUNCTION*, ktorá bude určovať spôsob aktualizácie hodnôt váh a biasov, podľa niektorej z optimalizačných metód. Konkrétne použitá TRAINLM je funkcia využívajúca metódu Levenberg-Marquardtovej optimalizácie.

Ďalšia funkcia je *ADAPTATION LEARNING FUNCTION*, ktorá bude udávať spôsob (metódu) akým sa bude sieť učiť, teda ako budú nastavené a naučené váhy

a bias. Použitá funkcia LEARNGDM je založená na využívaní algoritmu najstrmšieho spádu gradientu účelovej funkcie (GDM = Gradient Descent with Momentum, momentum umožňuje sieti, aby pri učení neprihliadala len na lokálnu hodnotu gradientu, ale tiež na vývoj chyby).

PERFORMANCE FUNCTION udáva spôsob, akým sa bude kontrolovať výsledok učenia siete, na základe zvolenej metódy sa bude počítať chyba trénovania (rozdiel medzi výstupom generovaným sieťou Outputs a nami zadaným výstupom Targets). Použitá MSE (Mean Square Error) je funkcia, ktorá počíta priemernú strednú kvadratickú odchýlku.

Ďalšie parametre zadávané v tomto okne súvisia so štruktúrou siete a s voľbou aktivačnej funkcie pre každú vrstvu. Treba si tu však uvedomiť, že v ponímaní MATLABu vstupná vrstva nie je vrstva, znamená to, že sa nezapočítava do počtu zadávaných vrstiev a ani sa jej nedefinujú žiadne funkcie. Ako prvý parameter sa volí počet vrstiev (*NUMBER OF LAYERS*), ktorý hovorí o počte skrytých vrstiev a výstupnej vrstvy. Následne možno pre každú vrstvu zvlášť zadávať v položke *NUMBER OF NEURONS* počet neurónov vo vrstve a aktivačnú funkciu danej vrstvy *TRANSFER FUNCTION*. Zvolená sieť bude mať jednu skrytú vrstvu s počtom neurónov 2 a aktivačnou funkciou TANSIG. Pre výstupnú vrstvu sa zvolil 1 neurón a aktivačná funkcia PURELIN. *Tansig* je (hyperbolická tangenciálna) sigmoidálna funkcia tvaru $a = \frac{2}{1+e^{-2\cdot n}} - 1$ (obr.1.15a) a *Purelin* lineárna funkcia (obr.1.15b)

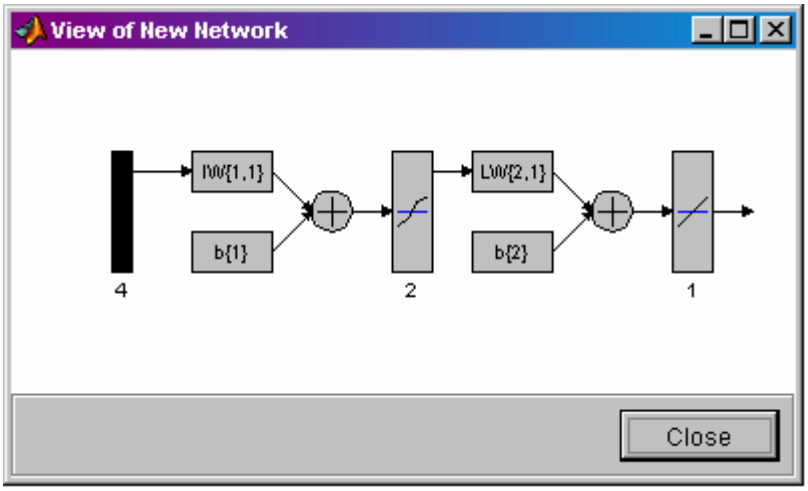


a) b)

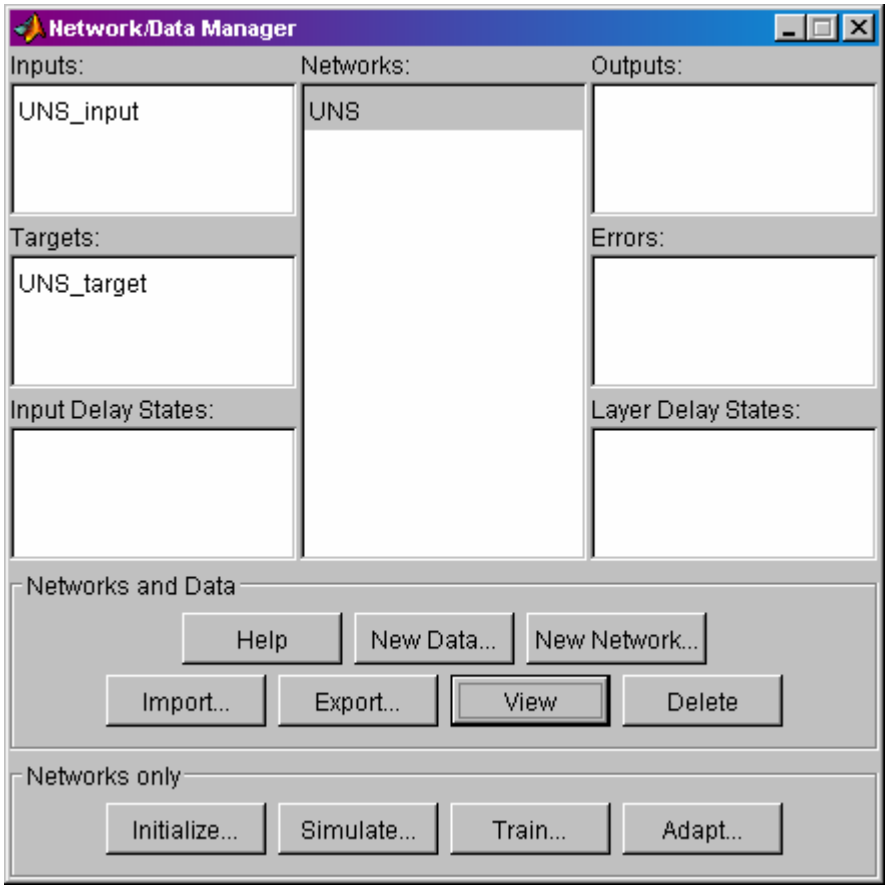
Obr.1.15 Ukážky použitých funkcií

Podrobnejšie sú všetky použité funkcie (a ešte mnoho ďalších) opísané v originálnej dokumentácii k Neural Network Toolboxu ***nnet.pdf*** vydanej ako užívateľská príručka spoločnosťou **The MathWorks, Inc.**

Zadefinovanú sieť si možno ešte pred vytvorením prezrieť po voľbe tlačidla VIEW (obr.1.16) alebo voľbou CREATE vytvoriť sieť. Následne sa nám vytvorená sieť objaví v okne **Network/Data Manager** (obr.1.17)



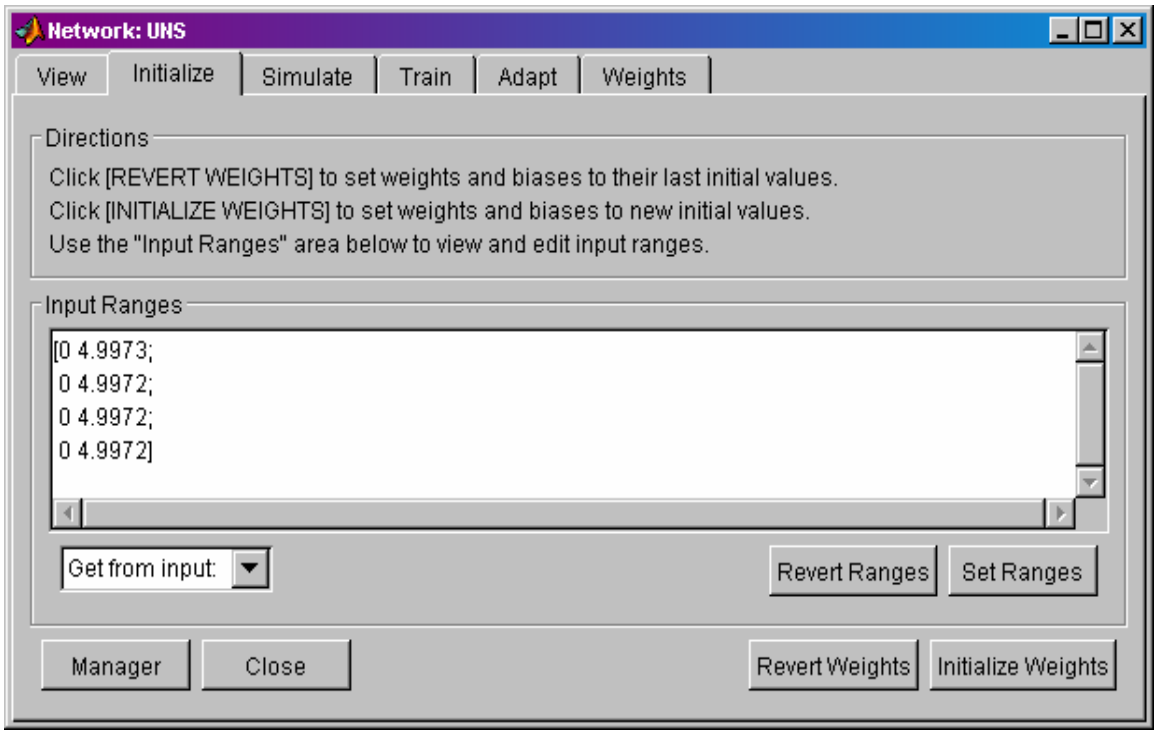
Obr.1.16 Grafické znázornenie neurónovej siete



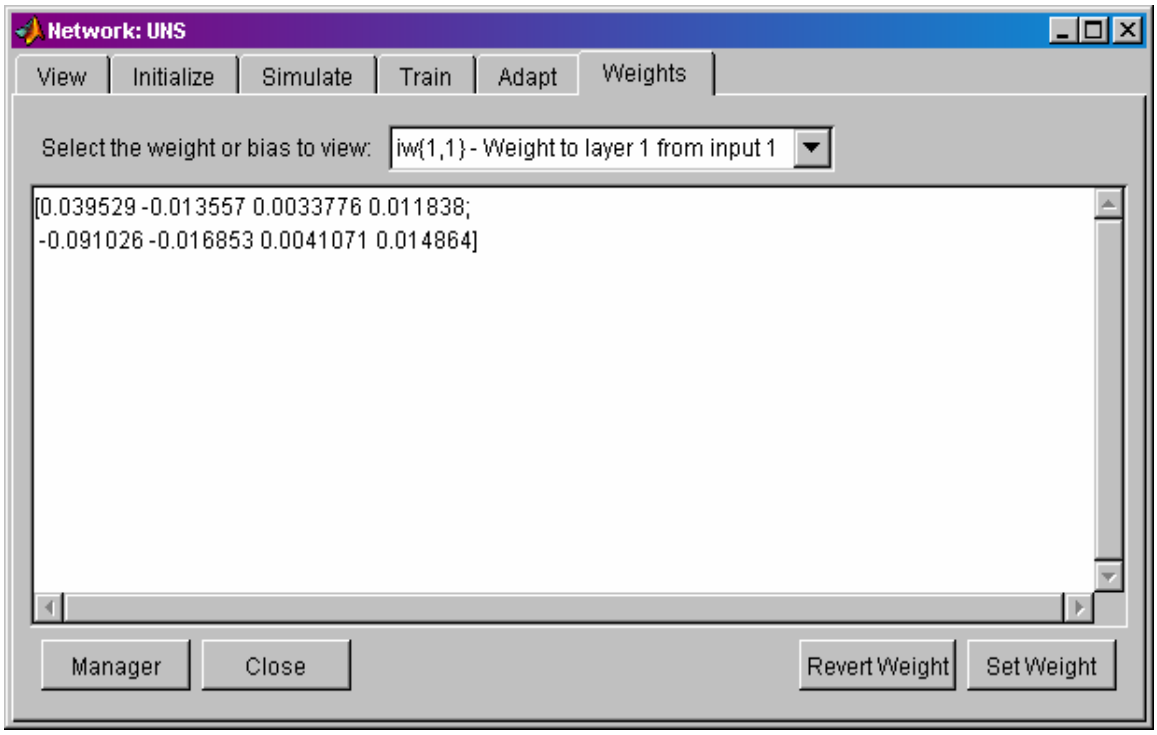
Obr.1.17 Network/Data Manager po vytvorení siete

Po označení siete a voľbe možnosti VIEW získame nové okno **Network:UNS** (názov siete) cez ktoré sa dá ďalej pracovať so sieťou. Okno je zostavené z viacerých záložiek. Záložka ***INITIALIZE*** (obr.1.18) slúži na zobrazenie rozsahu vstupných údajov. Umožňuje zmeniť tento rozsah a to prepísaním zobrazených hodnôt alebo načítaním nových údajov pomocou Get from input, ich nastavenie (uloženie) pomocou voľby SET RANGES alebo obnovu predchádzajúcich hodnôt pomocou voľby REVERS RANGES. Umožňuje tiež generáciu (inicializáciu) nových hodnôt váh a biasov pomocou INITIALIZE WEIGHTS (vygeneruje nové hodnoty) alebo obnovu predchádzajúcich nastavených hodnôt.

Hodnoty váh a biasov si možno prezerať voľbou záložky ***WEIGHTS*** (obr.1.19). Na začiatku trénovania sú tu zapísané počiatočné hodnoty inicializované pomocou niektorej z inicializačných funkcií (v tomto prípade ide o prednastavenú funkciu Initlay). Počas trénovania sa tieto hodnoty menia a po ukončení trénovania sa v okne objavia nové nastavené hodnoty.



Obr.1.18 Okno záložky INITIALIZE



Obr.1.19 Okno záložky WEIGHTS

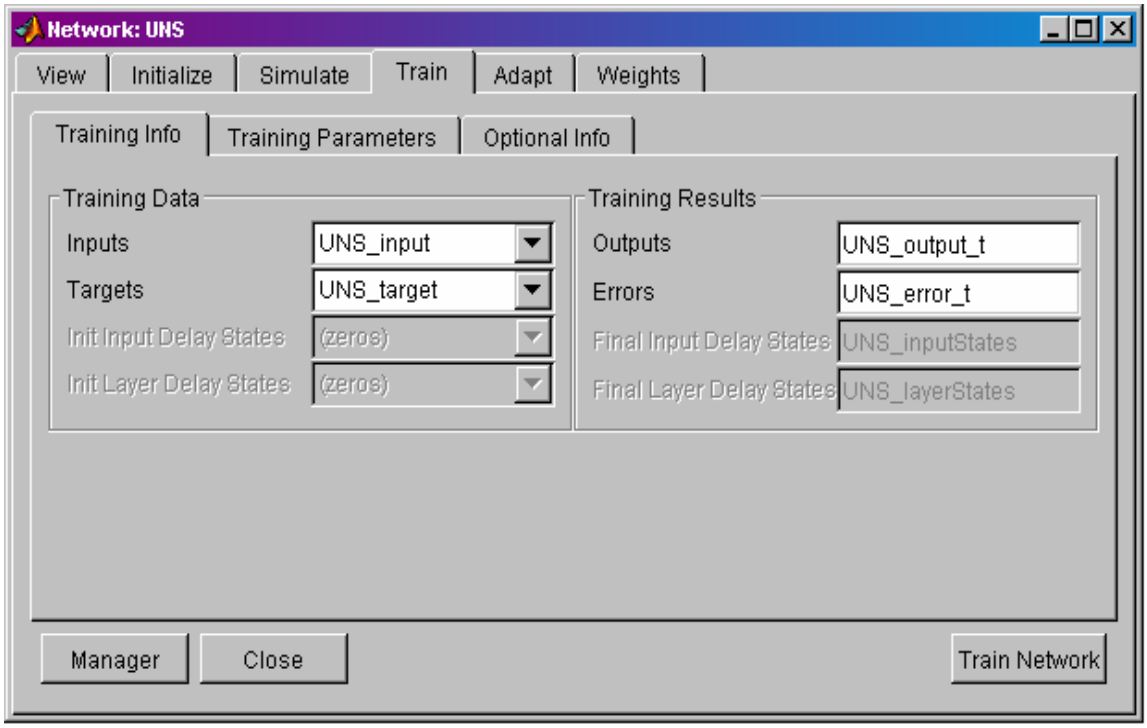
Hodnoty váh a biasov je možné meniť už spomenutým spôsobom pomocou INITIALITE WEIGHTS v záložke INITIALIZE alebo prepísaním zobrazených hodnôt v okne záložky WEIGHTS. Hodnoty sa nastavia voľbou SET WEIGHT. Voľba REVERT WEIGHT slúži na obnovu predchádzajúcich hodnôt.

Záložky *TRAIN* a *ADAPT* slúžia na voľbu formy trénovania a na zadávanie parametrov pre učenie siete. Tu je namieste vysvetlenie rozdielu medzi učením Incremental a Batch. Predpokladajme, že privádzaná trénovacia množina obsahuje N sád vstupno-výstupných dát. Pri Batch trénovaní sa obnova váh a biasov uskutočňuje až po tom, čo sa sieti predloží celá množina vstupných dát (teda všetkých N sád). To znamená, že po predložení každej sady sa vypočíta chyba odhadu (chybová funkcia), tento postup sa opakuje N krát, po predložení všetkých sád sa vypočíta konečná chybová funkcia a nastavia sa nové hodnoty váh a biasov. Pri Incremental trénovaní sa obnova váh koná po každej predloženej sade a vypočítaní chybovej funkcie, t.j. predloží sa jedna sada, vypočíta sa chybová funkcia a nastaví sa nová hodnota váh a biasov a predloží sa ďalšia sada atď.

Pri voľbe trénovania TRAIN môže byť použité len učenie Batch, pri ADAPT môžu byť použité obe formy učenia. To ktorá forma sa použije, závisí od spôsobu

zadania vstupných dát. Pri Batch sa vstupy (Inputs) zadávajú všetky v jednej matici rozmeru IxC (I = počet vstupov, C = počet sád údajov) a výstupy (Targets) ako matica rozmeru TxC (T = počet výstupov). Pri Incremental sa každá sada vstupu zadáva zvlášť ako vektor Ix1 (cell array) a výstupy ako matica Tx1. V prípade našej siete je len jeden výstup, ktorý má rozmer 1xC (riadkový vektor).

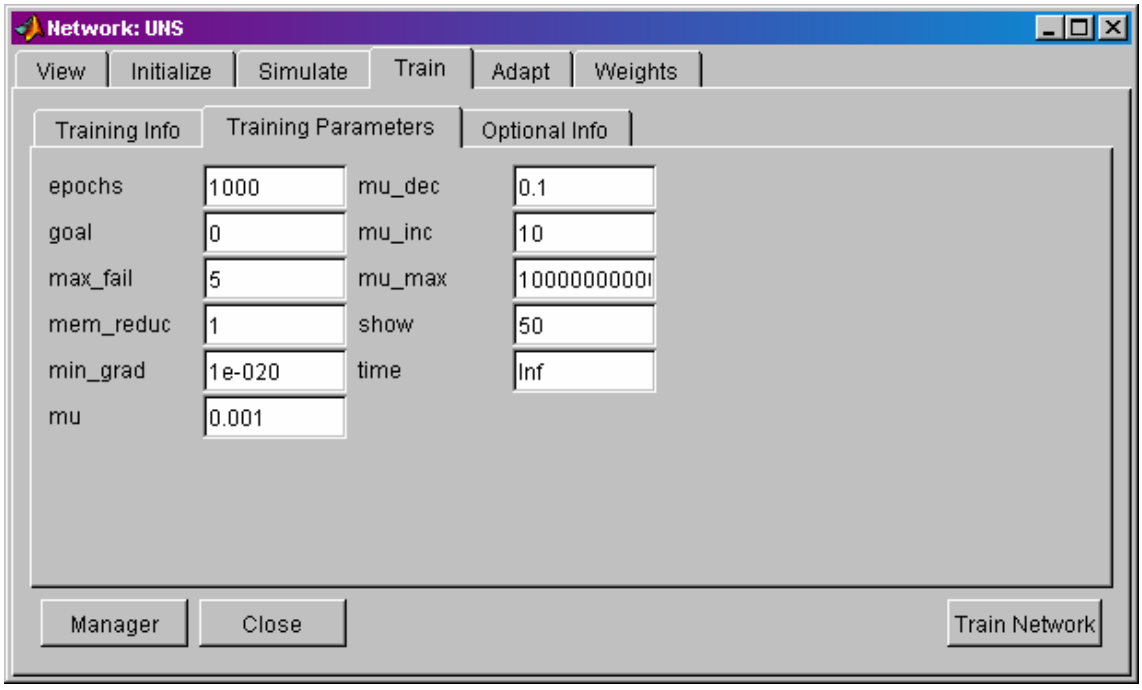
Voľbou záložky ***TRAIN*** (obr.1.20) si volíme trénovanie formou Batch učenia. Táto voľba ponúka možnosť nastavovania viacerých parametrov trénovania siete. V podzáložke ***Training Info*** sa v stĺpci *Training Data* zadávajú vstupy do siete Inputs (voľbou vstupných dát uložených ako UNS_input) a k nim prislúchajúci žiadaný výstup Target (voľbou výstupných dát uložených ako UNS_target). V stĺpci *Training Result* sa v položke Outputs zadá názov pod ktorým sa uložia generované výstupy siete ako odozva na zadané vstupy (UNS_output_t), a v položke Errors názov pod ktorým sa uložia chyby trénovania (UNS_error_t).



Obr.1.20 Okno záložky TRAIN/Training Info

V podzáložke ***Training Parameters*** (obr.1.21) sa nastavujú parametre trénovania. Trénovanie sa zastaví, ak jeden z nastavených parametrov dosiahne resp. klesne pod zadanú hodnotu. Každý z parametrov ovplyvňuje kvalitu natrénovanej siete, preto je ich voľba dôležitou súčasťou vytvárania siete. Možno tu teda meniť prednastavené hodnoty

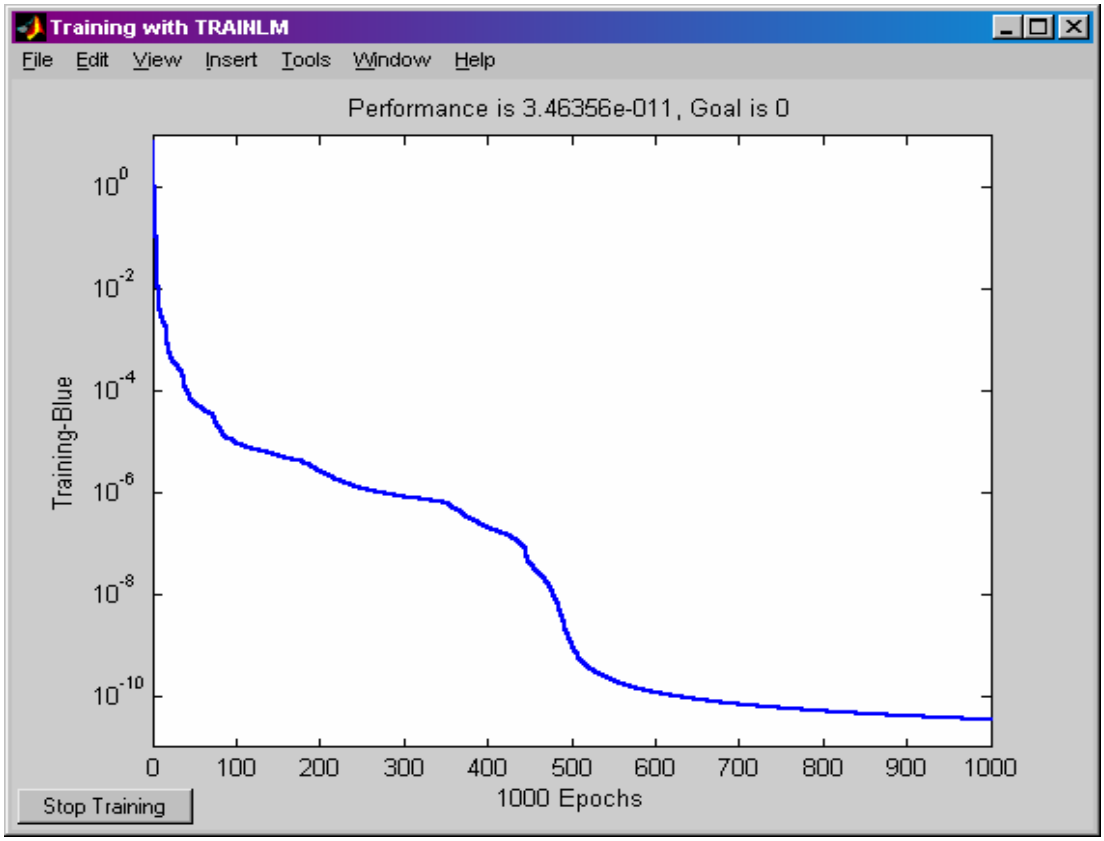
parametrov ako je počet epoch EPOCHS, hodnotu minima gradientu účelovej funkcie MIN_GRAD, hodnotu účelovej funkcie, ktorú je želané dosiahnuť GOAL. Vývoj tejto hodnoty je možné sledovať v grafe (obr.1.22), kde sa zobrazuje aj hodnota parametra GOAL (ak je nastavená iná ako 0), z grafom súvisí parameter SHOW, ktorý udáva po koľkých epochách sa bude graf obnovovať (prednastavená je hodnota 25, ak si užívateľ neželá aby sa graf objavoval hodnotu parametra treba nastaviť na Inf). Ďalším parametrom je čas trénovania TIME (zadáva sa v sekundách), prednastavená hodnota je Inf. Sieť je možné trénovať s prednastavenými hodnotami parametrov, alebo si ich užívateľ môže zmeniť v prípade, že trénovanie nie je úspešné. Nastavenie parametrov je na úsudku samotného užívateľa aplikovaním metódy pokus-omyl. Ak nepomôže zmena parametrov trénovania, je potrebné reinicializovať váhy na iné hodnoty. Parametre MU, MU_DEC, MU_INC, MU_MAX súvisia s parametrom η ktorý vystupuje v Levenberg-Marquardtovej rovnici na výpočet zmeny hodnoty váh a biasov.



Obr.1.21 Okno záložky TRAIN/Training Parameters

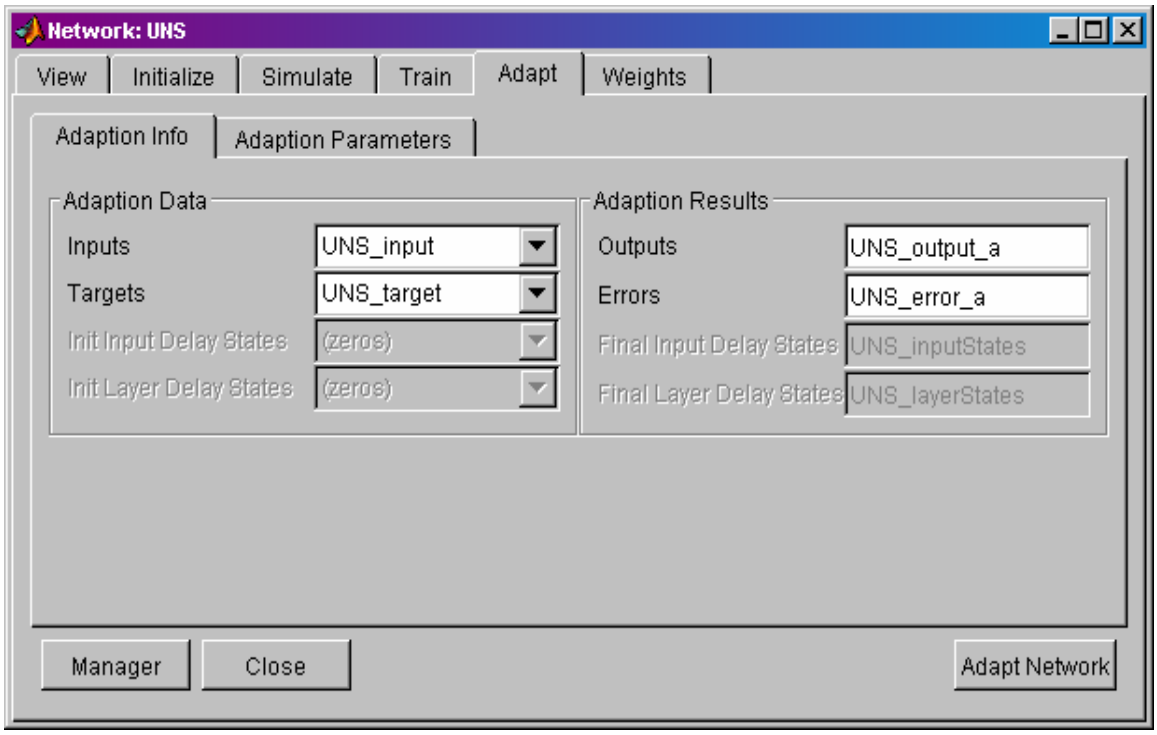
Pri trénovaní ukážkovej siete sa menil počet epoch z prednastavenej hodnoty 100 na 1000 iterácií a hodnota minima gradientu na 1e-020 (prednastavená hodnota 1e-010 bola dosiahnutá skôr než počet epoch dosiahol žiadanú hodnotu a trénovanie sa tak ukončilo skôr než bolo žiaduce).

Grafické znázornenie trénovania siete je na obr.1.22, je tu zobrazený vývoj hodnoty účelovej funkcie počas celej doby trénovania.

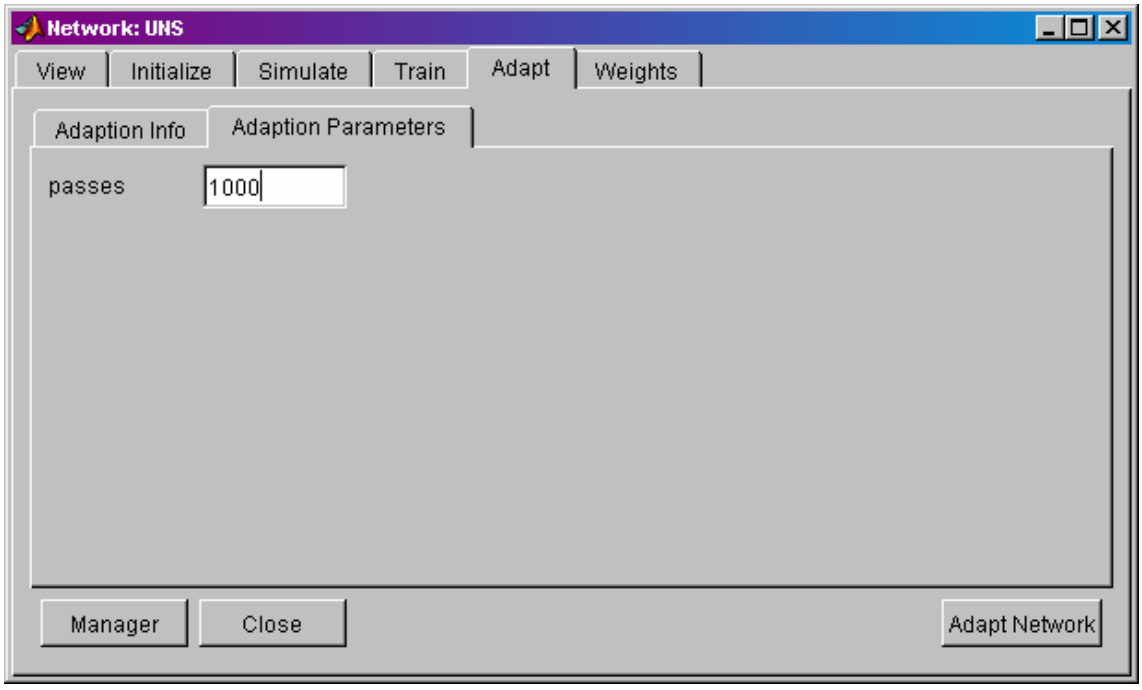


Obr.1.22 Vývoj hodnoty účelovej funkcie počas trénovania

Ďalšou záložkou je už spomínaný ***ADAPT*** (obr.1.23). Tu je možné využiť na trénovanie obe formy učenia, to ktorá sa nakoniec použije bude závisieť od formy zadania trénovacej množiny. V podzáložke ***Adaption Info*** sa podobne ako v Training Info v stĺpci *Adaption Data* zadávajú vstupy do siete Inputs (voľbou vstupných dát uložených ako UNS_input) a k nim prislúchajúci žiadaný výstup Target (voľbou výstupných dát uložených ako UNS_target). V stĺpci *Adaption Result* sa v položke Outputs zadá názov pod ktorým sa uložia generované výstupy siete ako odozva na zadané vstupy (UNS_output_a), a v položke Errors názov pod ktorým sa uložia chyby trénovania (UNS_error_a). Jediným nastavovateľným parametrom v podzáložke ***Adaption Parameters*** je tu parameter PASSES (obr.1.24), ktorého význam je ten istý ako u parametra EPOCHS pri záložke TRAIN. Ide rovnako o počet cyklov koľkokrát sa sieti predloží celá tréningová množina a zmenia sa hodnoty váh a biasov.



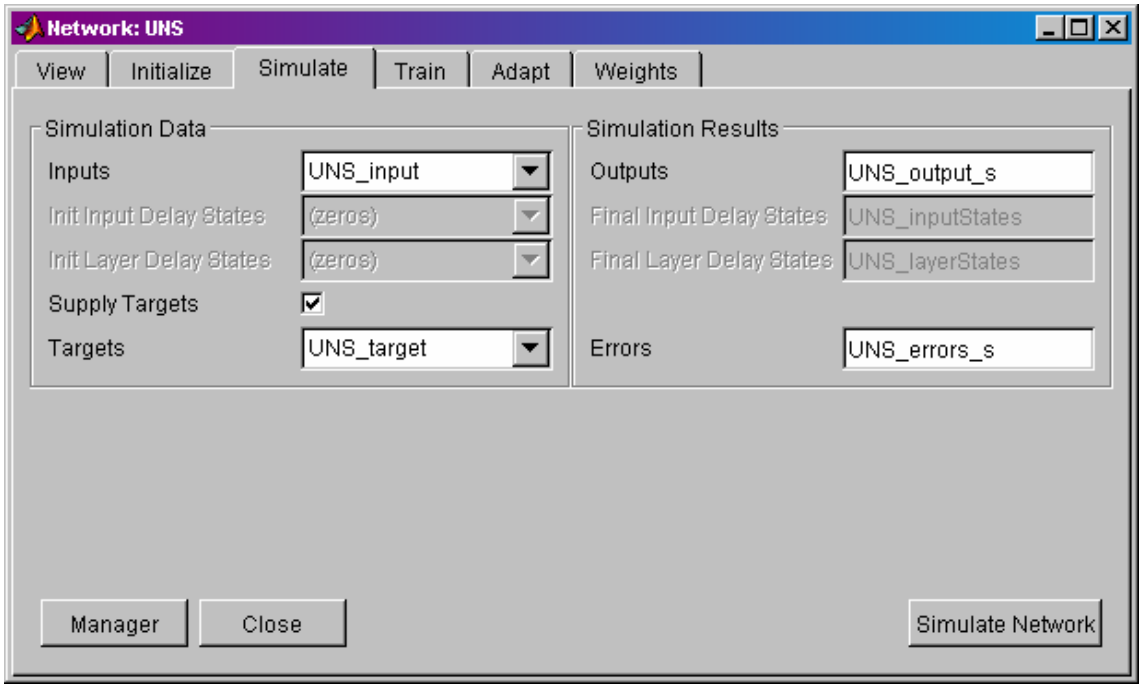
Obr.1.23 Okno záložky ADAPT/Adaption Info



Obr.1.24 Okno záložky ADAPT/Adaption Parameters

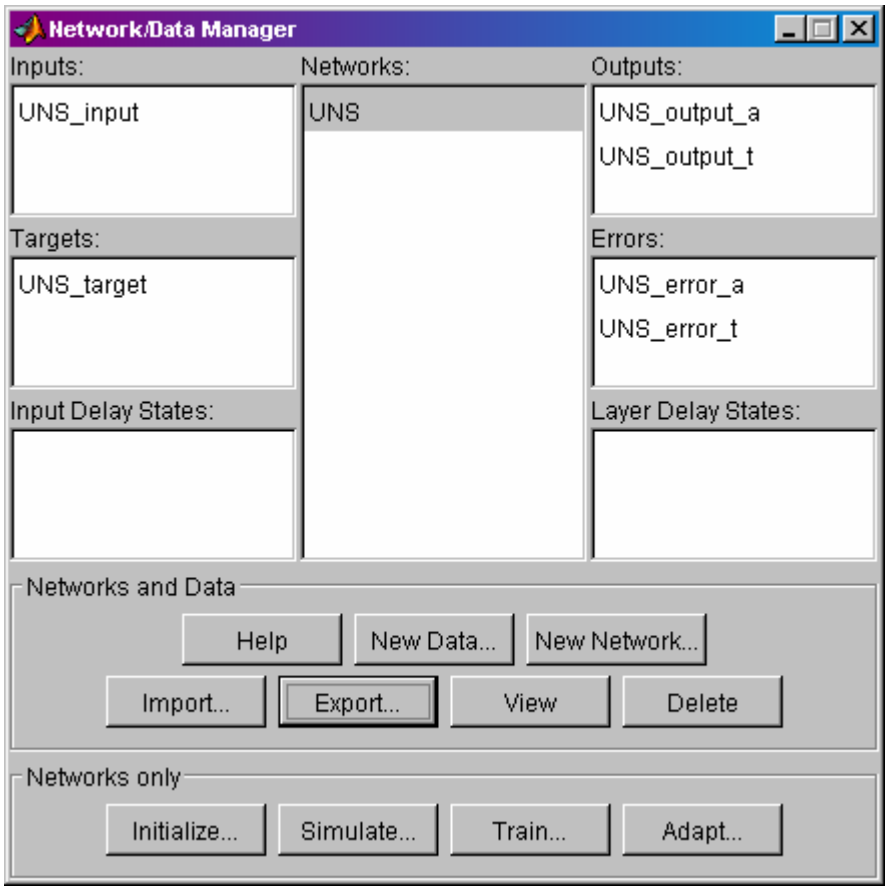
Sieť je možné aj simulovať. Túto možnosť ponúka záložka ***SIMULATE*** (obr.1.25). Ponúka možnosť simulovať výstup siete bez toho, aby sa sieť exportovala do MATLABu. Možno tu simulovať sieť bez trénovania a zistiť tak aký dobrý je prvotný

odhad váh. (sieť pracuje s váhami, ktoré sú vygenerované pri jej vytvorení, preto býva výsledok takejto simulácie neuspokojujúci). Alebo sieť najprv natrénovať a následne simulovať a získaný výstup siete porovnať so žiadaným výstupom. V *Simulation Data* sa zadávajú údaje ako sú vstupné dáta Inputs (UNS_input), pri voľbe *Supply Targets* sa zadáva aj súbor s hodnotami žiadaného výstupu (UNS_target) a tak sa počas simulácie vypočítava aj chyba medzi odhadom siete a nami žiadaným výstupom (UNS_errors_s). Výsledky simulácie (hodnoty výstupu generovaného sieťou a chyby odhadu) sa ukladajú do *Simulation Results*, kde názov položky si môže užívateľ zvoliť sám (napr. UNS_output_s).

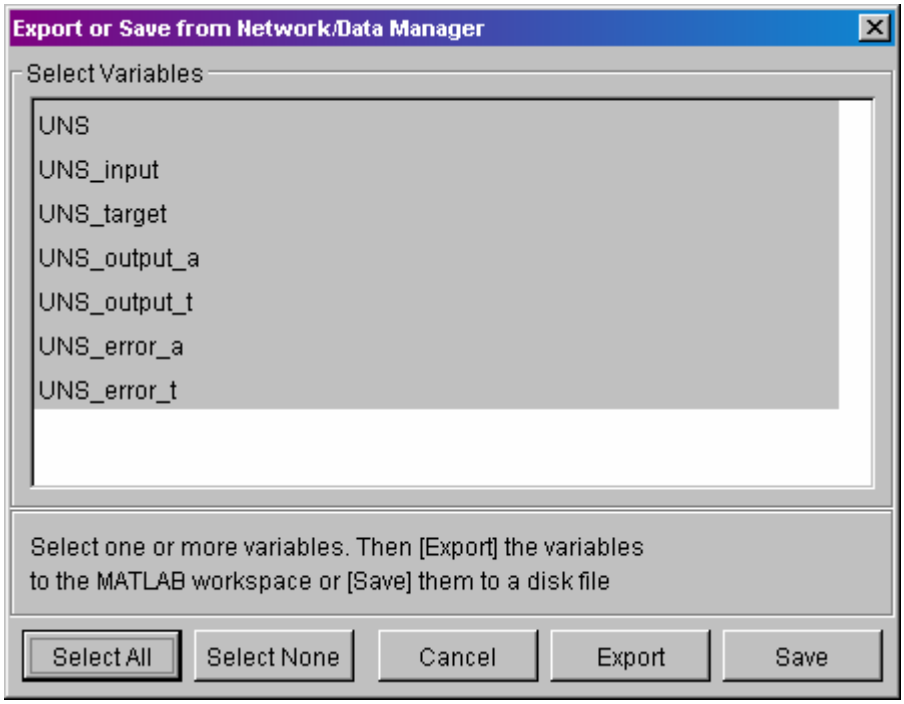


Obr.1.25 Okno záložky SIMULATE

Výsledky trénovania siete sa zobrazujú v okne Network/Data Manager (obr.1.26), voľbou možnosti EXPORT získame nové okno **Export or Save** from Network/Data Manager (obr.1.27) odkiaľ je možné označiť ľubovoľný z údajov, a následne pomocou voľby EXPORT ich exportovať do Workspace v MATLABe alebo ich uložiť na disk voľbou SAVE.

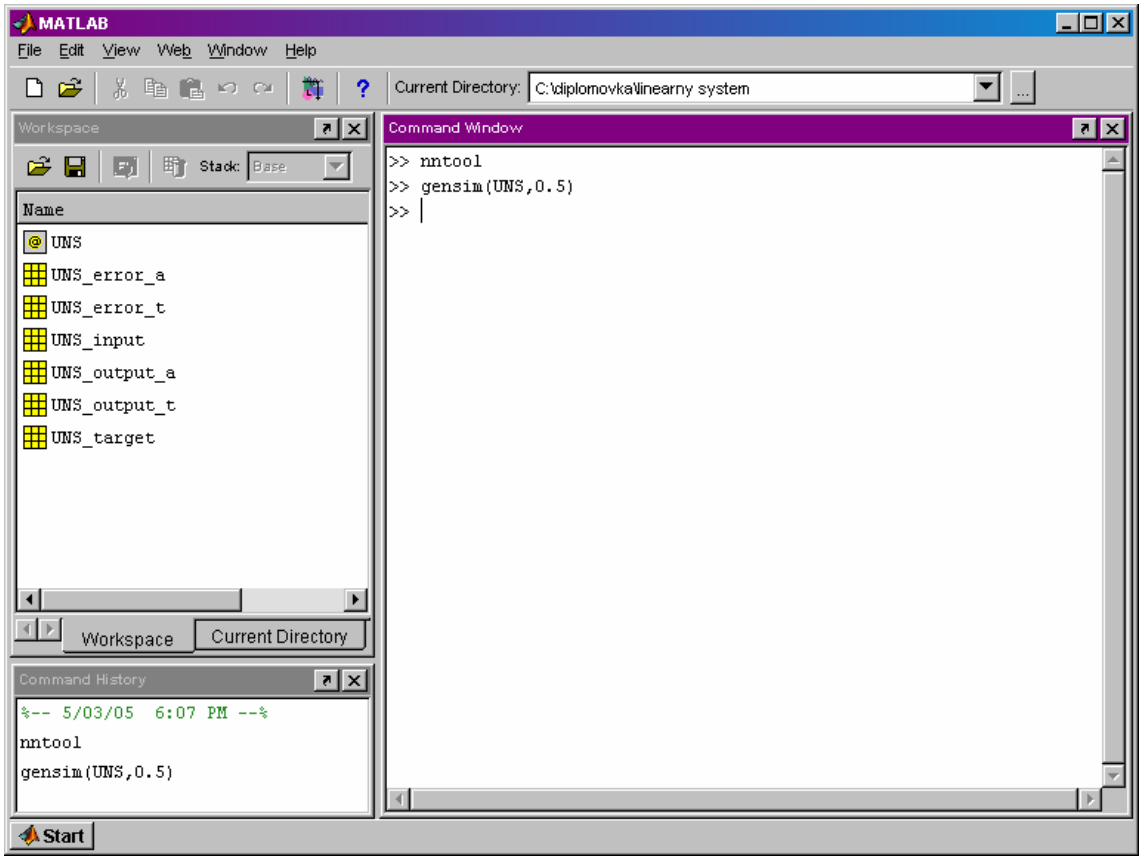


Obr.1.26 Network/Data Manager s naimportovanými vstupno-výstupnými dátami, vytvorenou sieťou a s výsledkami trénovania siete

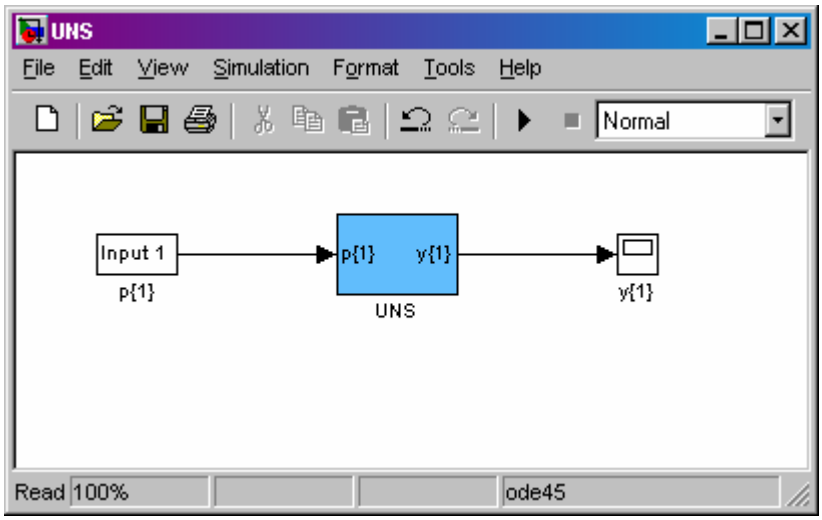


Obr.1.27 Okno pre exportovanie dát z NNT

Vyexportovanú sieť je možné v MATLABe vygenerovať ako simulinkovský blok (obr.1.28) pomocou príkazu **gensim(názov siete, perióda vzorkovania)**, štandardne je nastavená perióda vzorkovania 1. Získaný blok je na obr.1.29

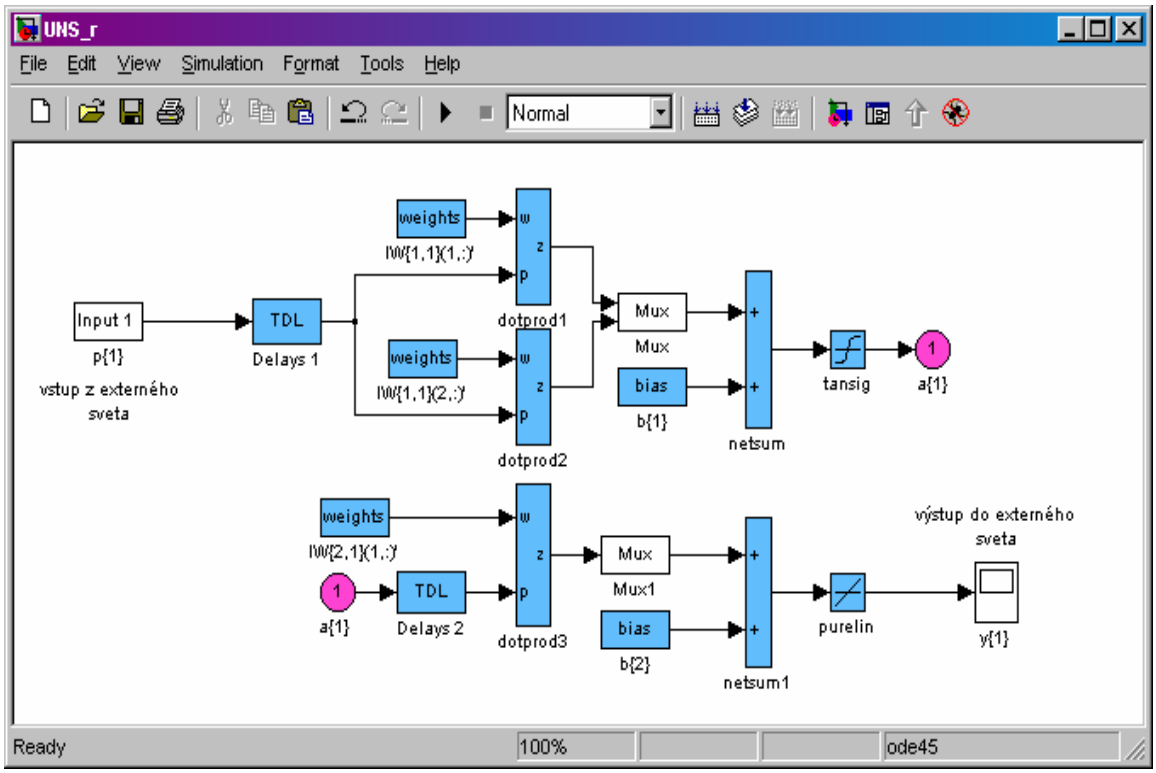


Obr.1.28 Užívateľské prostredie MATLAB po naexportovaní údajov z GUI Neural Network Toolboxu



Obr.1.29 Vygenerovaný simulinkovský blok UNS

Na obr.1.30 je zobrazená vytvorená sieť .



Obr.1.30 Zobrazenie siete

2 EXPERIMENTÁLNA ČASŤ

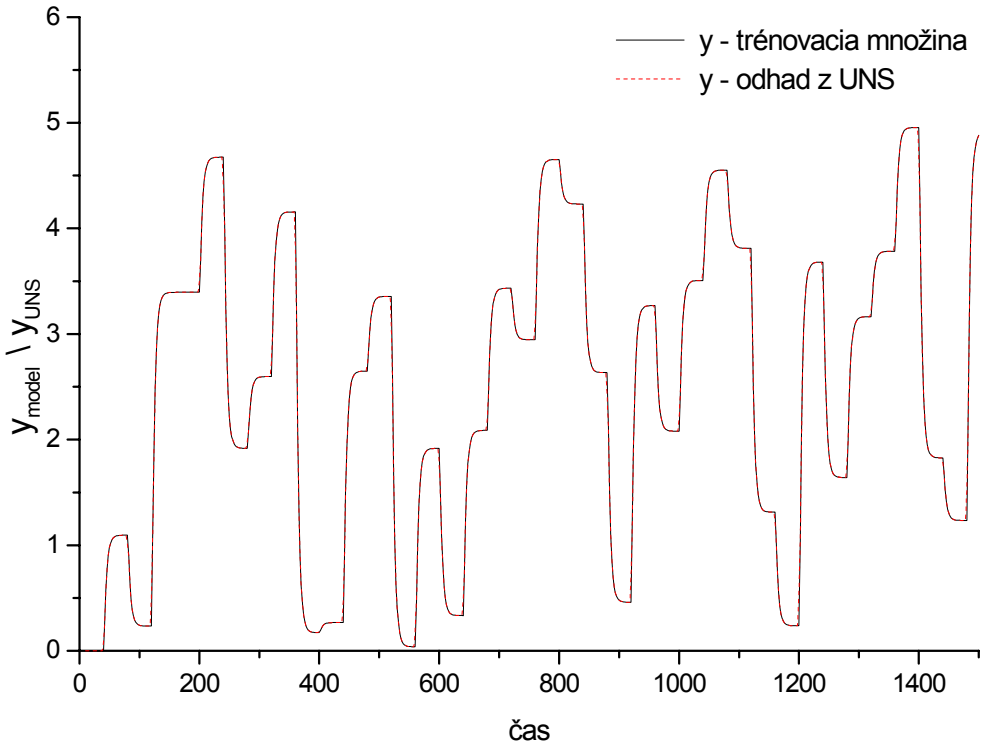
2.1 Identifikácia a riadenie lineárneho systému

2.1.1 Identifikácia lineárneho systému

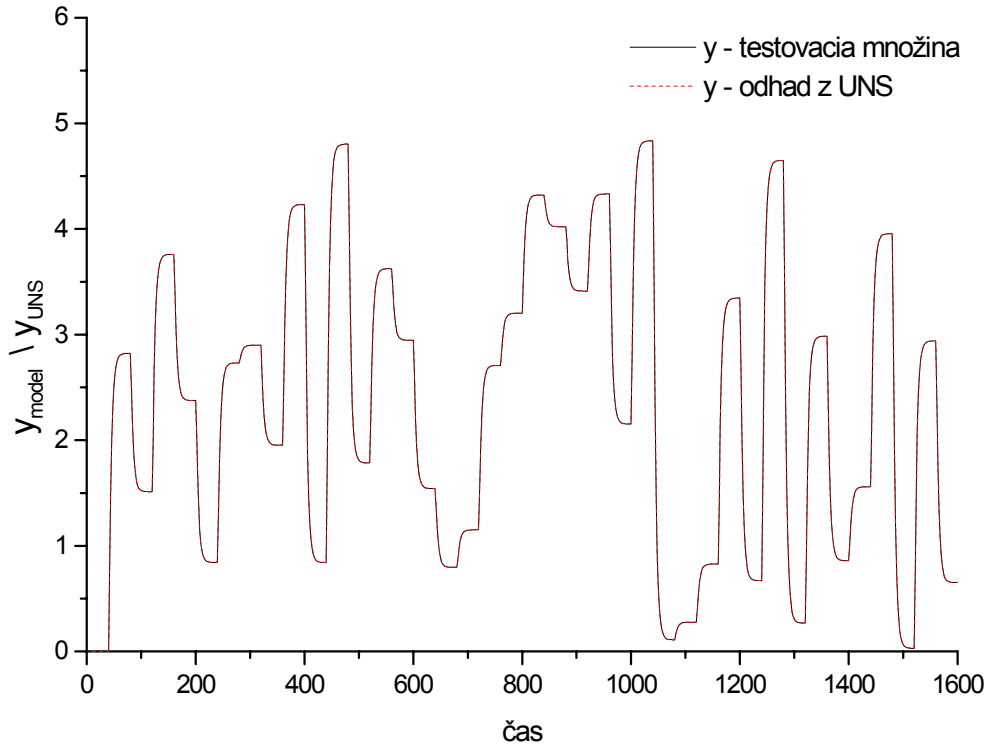
Ako príklad lineárneho systému sa vybral systém 2. rádu s prenosom $G_s = \frac{1}{4s^2 + 5s + 1}$. Identifikáciou lineárneho systému pomocou UNS sa rozumie získanie dopredného modelu systému. Štruktúra siete použitej na identifikáciu tohoto lineárneho systému bola {4,2,1}. Na vstup do siete sa privádzali aktuálna hodnota vstupu do modelu systému $u(k)$, jeho prvá história $u(k-1)$, hodnota aktuálneho výstupu $y(k)$ a jeho prvá história $y(k-1)$. Ako zodpovedajúci výstup sa zadávala hodnota $y(k+1)$ ako odozva na vstup $u(k)$. Výstup zo siete $\hat{y}(k+1)$ zodpovedal hodnote výstupu zo systému $y(k+1)$.

Identifikácia systému prebiehala off-line. trénovacia množina sa vytvorila simuláciou matematického modelu lineárneho systému. Na vstup systému sa privádzalo postupne 40 náhodných skokových zmien. Vzorkovaním s periódou vzorkovania 2 počas simulácie, sa získal súbor trénovacích dát, ktorý obsahoval 750 párov vstupno-výstupných údajov. Po 1000 epochách pri učení klesla chyba epochy na hodnotu 1.10^{-5}. Nízka chyba epochy dokazuje, že sieť s danou štruktúrou je schopná dobre popísať lineárny systém. Pre možnosť porovnania sa za rovnakých podmienok vykonali simulácie aj so sieťou štruktúry {4,3,1}, mala teda o 1 skrytý neurón viac a kde chyba epochy po skončení učenia mala hodnotu 3.10^{-5} a bola teda o niečo málo vyššia. Čo len potvrdilo výber siete s 2 skrytými neurónmi za vhodný a sieť za postačujúcu.

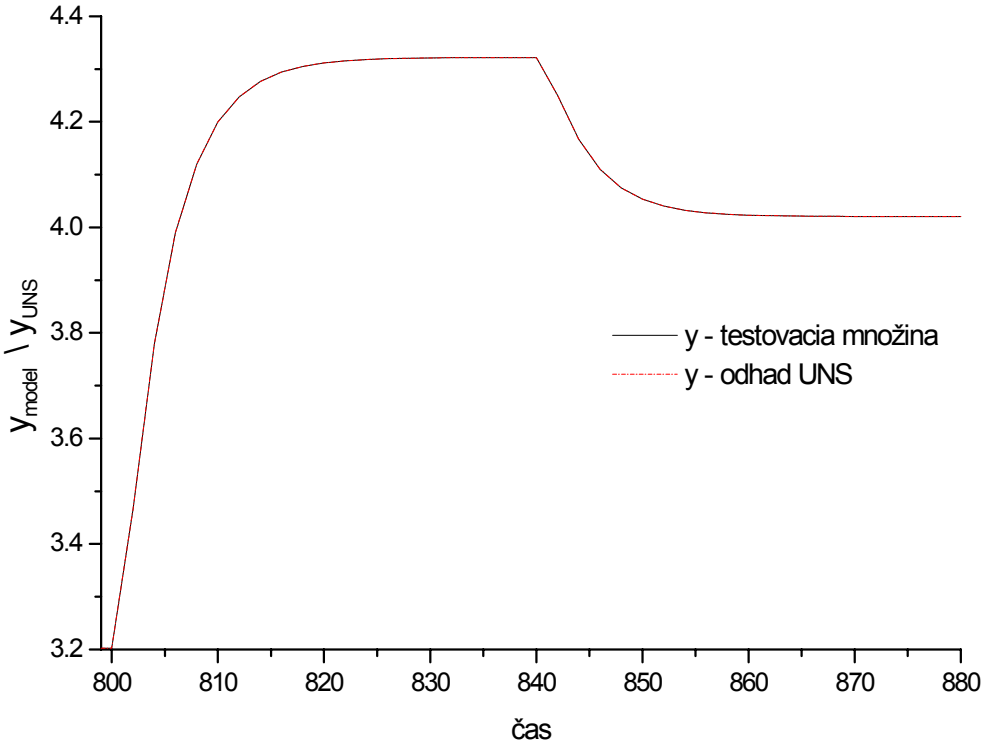
Porovnanie odhadu výstupnej veličiny z natrénovanej neurónovej siete s trénovacou množinou je na obr.2.1. Testovanie natrénovanej siete na testovacích dátach je na obr.2.2 resp. na obr.2.3, odkiaľ je zrejmé, že natrénovaná sieť výborne opisuje aj testovaciu množinu a dobre zachytáva dynamiku systému. Testovacia množina sa získala podobne ako trénovacie dáta.



Obr.2.1 Porovnanie odhadu výstupnej veličiny z UNS s trénovacími dátami



Obr.2.2 Testovanie neurónovej siete na testovacích dátach

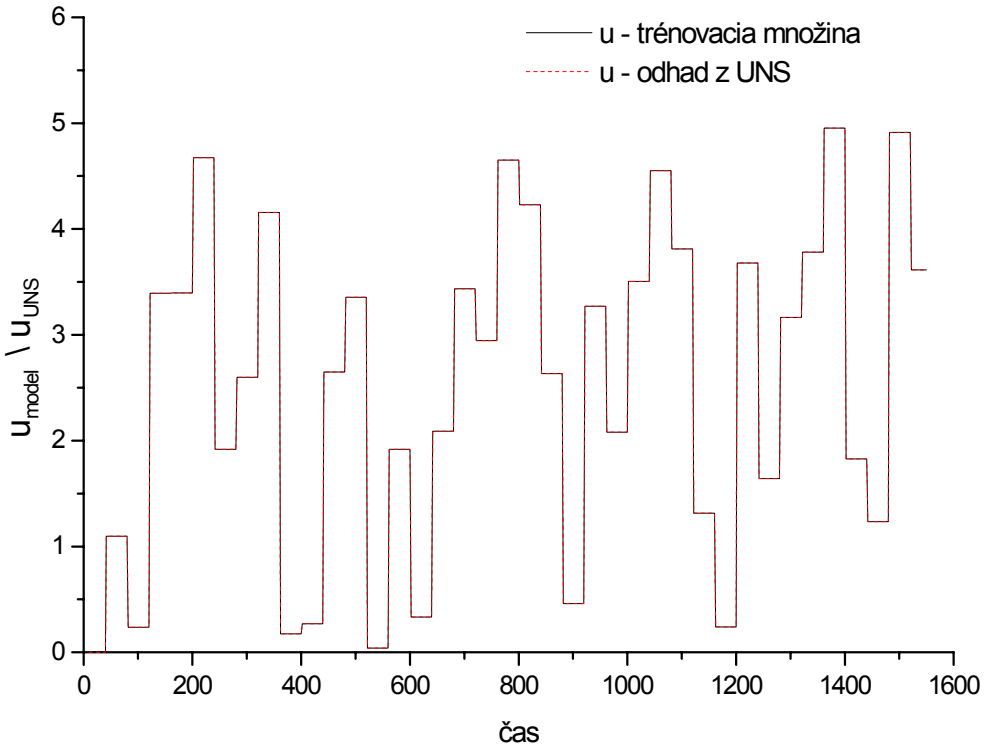


Obr.2.3 Časť porovnania testovacích dát s odhadom z neurónovej siete

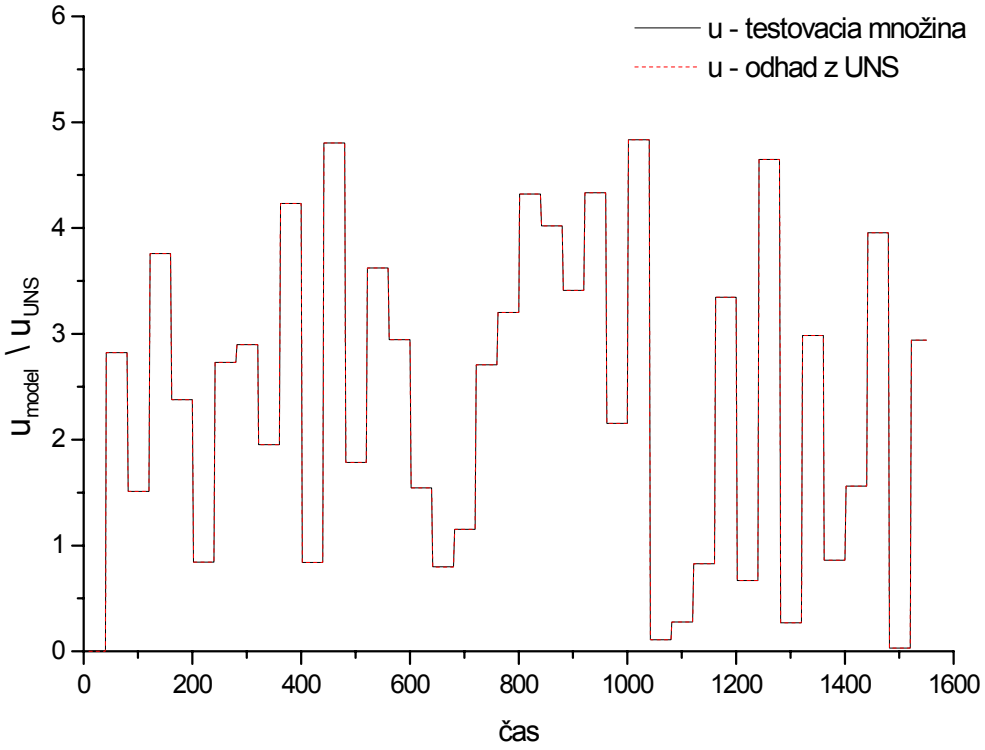
2.1.2 Riadenie lineárneho systému

Na riadenie lineárneho systému 2. rádu sa použil inverzný neurónový regulátor so štruktúrou siete {4,3,1}.

Regulátor bol trénovaný off-line. Na vstup do siete predstavujúcej inverzný model systému sa počas trénovania privádzala prvá história vstupu do systému $u(k-1)$, hodnota aktuálneho výstupu $y(k)$ a jeho prvá história $y(k-1)$ a hodnota výstupu $y(k+1)$ zodpovedajúca odozve systému na vstup $u(k)$. Odhad zo siete zodpovedal hodnote $u(k)$. Trénovací súbor obsahoval 800 vzoriek predstavujúcich vstupno-výstupné dáta odoziev systému na náhodne privedené skokové zmeny vstupnej veličiny $u(k)$. Chyba epochy po ukončení trénovania (1000 iterácií) klesla na hodnotu 1.10^{-5}. Na obr.2.4 je porovnanie odhadu výstupu z UNS s trénovacími dátami a na obr.2.5 je porovnanie pre testovacie dáta.



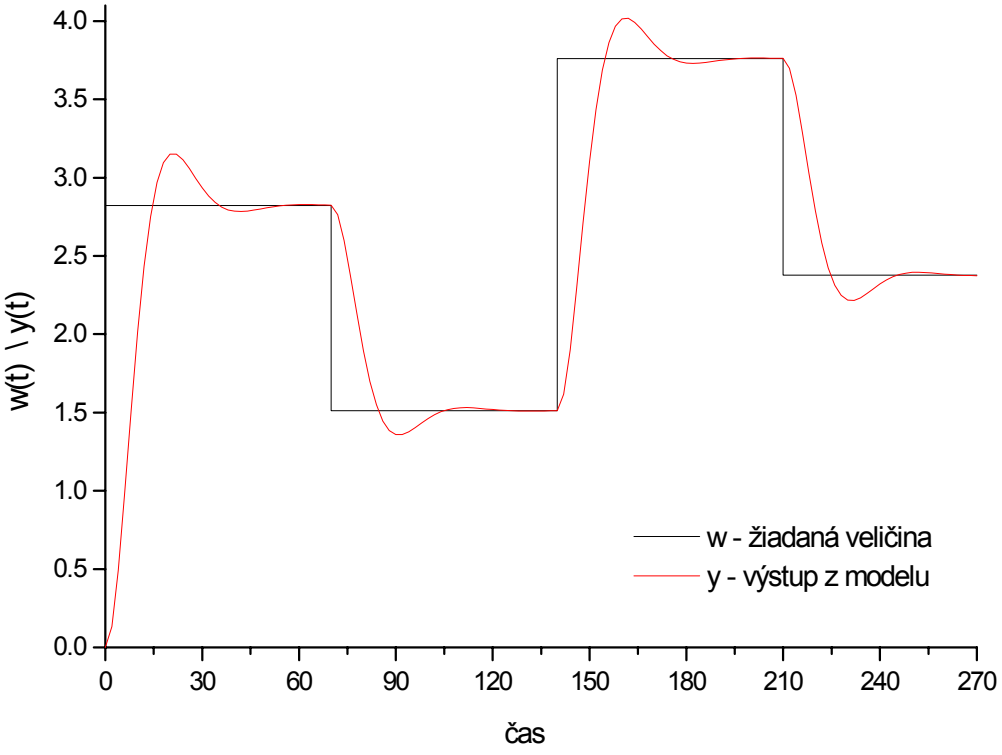
Obr. 2.4 Porovnanie odhadu riadiacej veličiny z UNS s trénovacími dátammi



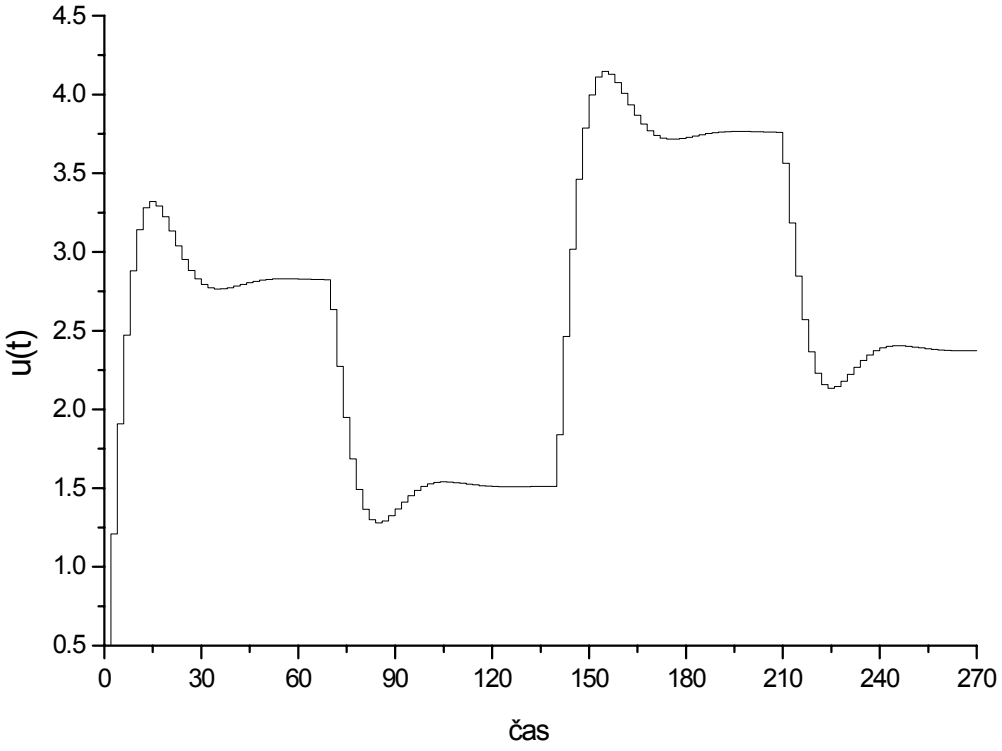
Obr. 2.5 Porovnanie odhadu riadiacej veličiny z UNS s testovacími dátami

2.1.2.1 Riadenie nominálneho systému

Natrénovaná UNS bola ďalej použitá ako inverzný neurónový regulátor na riadenie *nominálneho systému* s prenosom uvedeným v kapitole 2.1.1. Na vstup regulátora pri riadení sa miesto hodnoty $y(k+1)$ privádzala žiadaná veličina $w(k)$, prvá história výstupu regulátora $u(k-1)$, výstup zo systému $y(k)$ a jeho prvá história $y(k-1)$. Dané veličiny zabezpečovali „spätnú“ väzbu regulačného obvodu tým, že do inverzného regulátora privádzali informácie o riadenom procese. Regulátor pracoval s periódou vzorkovania 0,5. Na obr.2.6 je priebeh regulácie lineárneho nominálneho systému neurónovým regulátorom a na obr.2.7 je zodpovedajúci priebeh riadiacej veličiny regulátora.



Obr. 2.6 Riadenie nominálneho systému pomocou inverzného neurónového regulátora



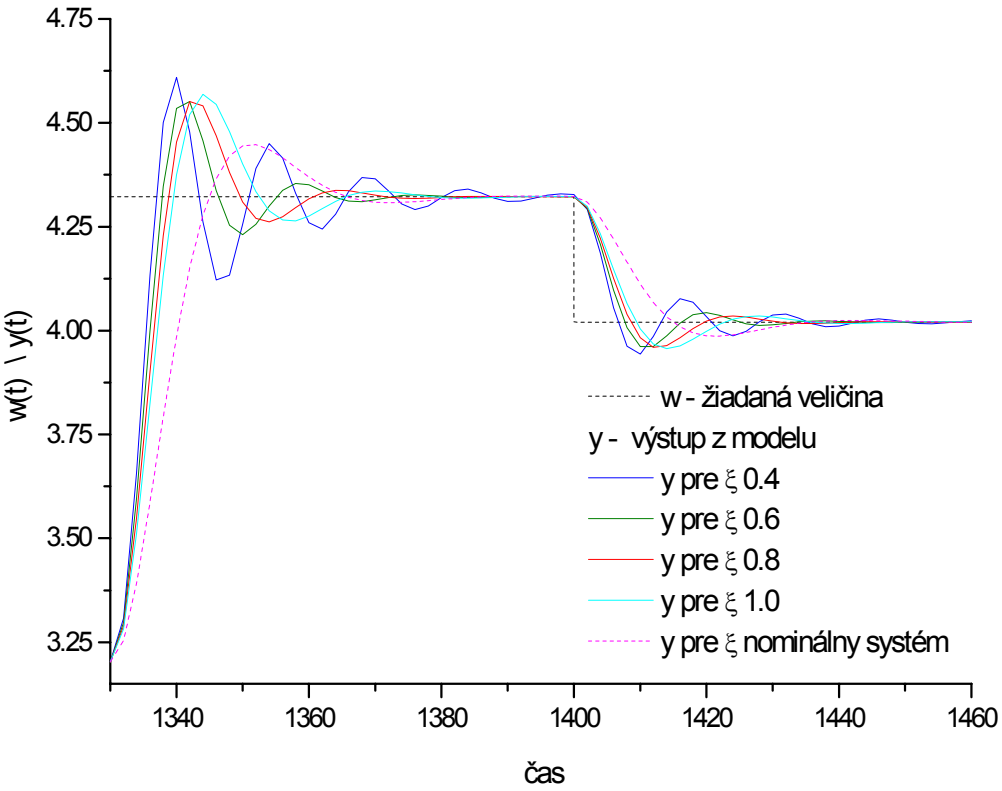
Obr. 2.7 Priebeh riadiacej veličiny inverzného neurónového regulátora

2.1.2.2 Riadenie perturbovaného systému

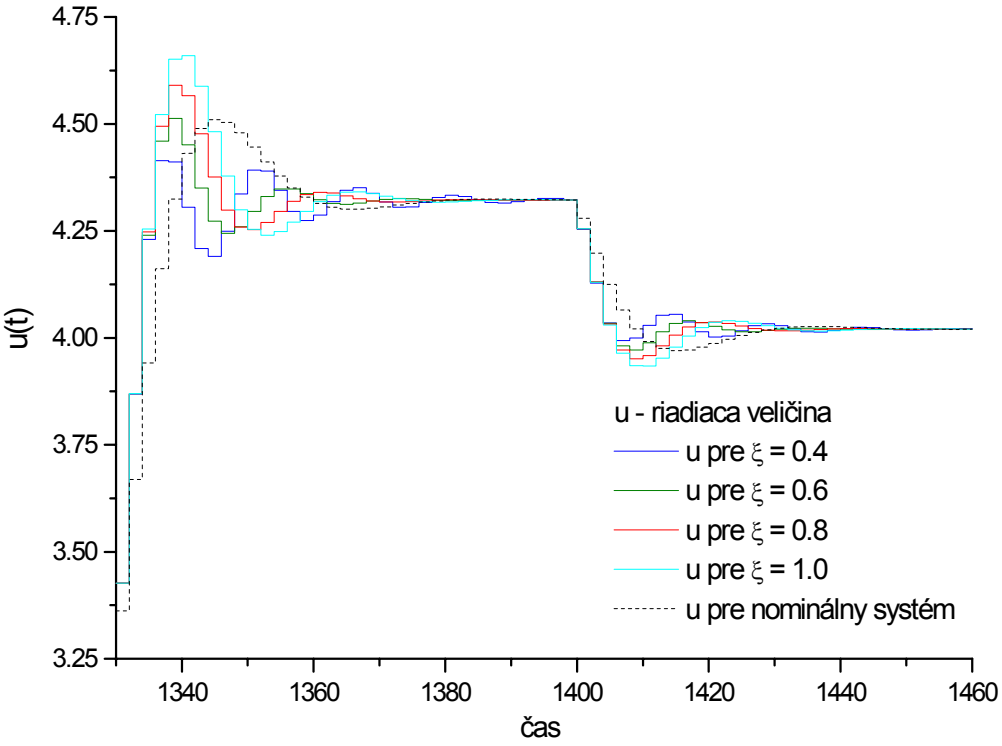
Robustnosť natrénovaného regulátora sa testovala pri riadení perturbovaného lineárneho systému 2. rádu. Regulátor podobne ako pri riadení nominálneho systému, pracoval s periódou vzorkovania 0,5. Perturboval sa koeficient tlmenia ξ a následne zosilnenie systému.

A. **Perturbácia koeficientu tlmenia lineárneho systému ξ**

Hodnota koeficientu tlmenia sa zmenšovala od nominálnej hodnoty $\xi = 1{,}25$ do oblasti periodicity systému. Regulátor uspokojivo ureguloval systém až po hodnotu koeficientu tlmenia okolo $\xi = 0{,}6$. Pri nižších hodnotách riadený systém silno kmital, čo viedlo k zvyšovaniu doby regulácie systému a nestabilite systému. Priebehy riadenia lineárneho systému s perturbovaným koeficientom tlmenia sú zobrazené na obr.2.8 a na obr.2.9 sú zobrazené priebehy riadiacej veličiny neurónového regulátora pri riadení perturbovaného systému.



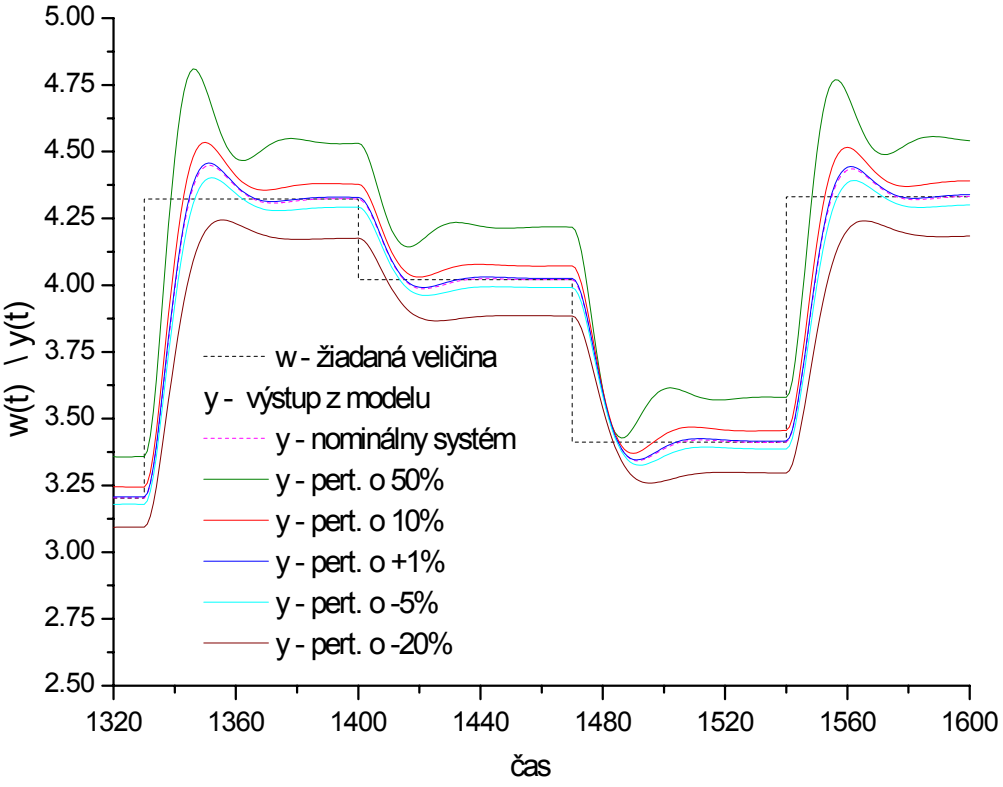
Obr. 2.8 Porovnanie priebehov riadenia systému s perturbovaným koeficientom tlmenia



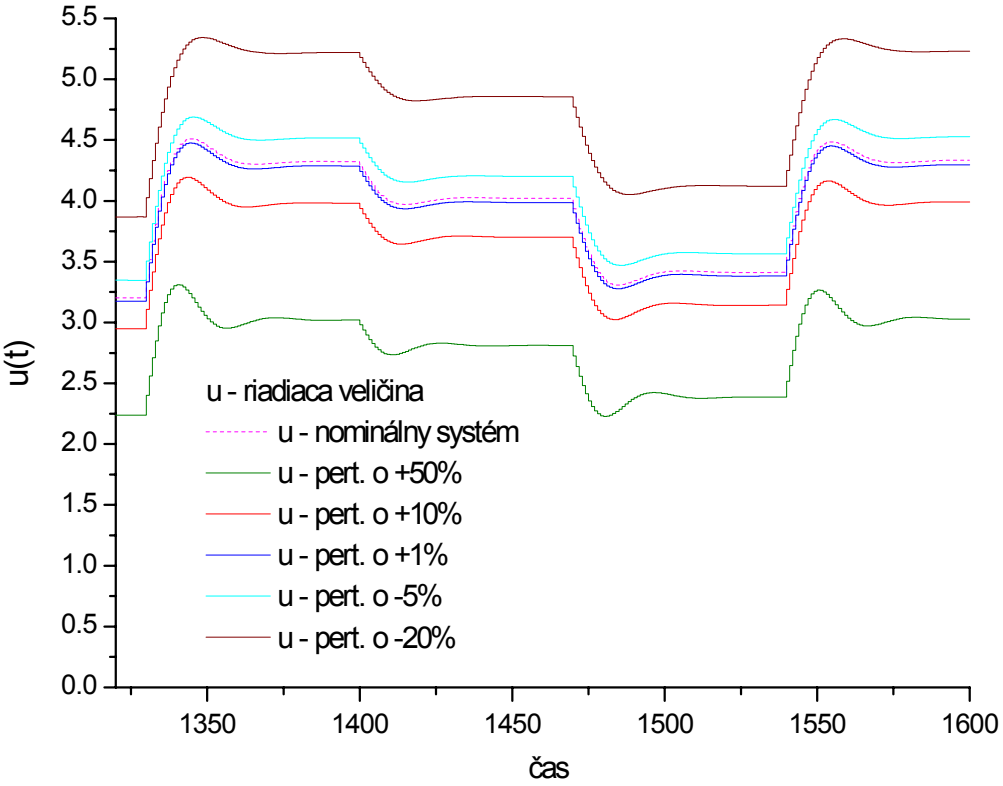
Obr. 2.9 Priebehy riadiacej veličiny neurónového regulátora pri riadení perturbovaného systému

B. Perturbácia zosilnenia systému

Hodnota zosilnenia systému sa menila v rozsahu ± 50% nominálnej hodnoty. Natrénovaný regulátor je schopný dostatočne presne uriadiť perturbovaný systém so zosilnením odlišujúcim sa od nominálnej hodnoty v rozsahu ± 1%. Ďalší nárast perturbácie zosilnenia vedie k zvyšovaniu trvalej regulačnej odchýlky, ktorú už regulátor nie je schopný odstrániť. Priebehy regulácií systému s perturbovaným zosilnením a zodpovedajúce priebehy riadiacej veličiny sú na obr.2.10 resp. obr.2.11.



Obr. 2.10 Porovnanie priebehov riadenia systému s perturbovaným zosilnením

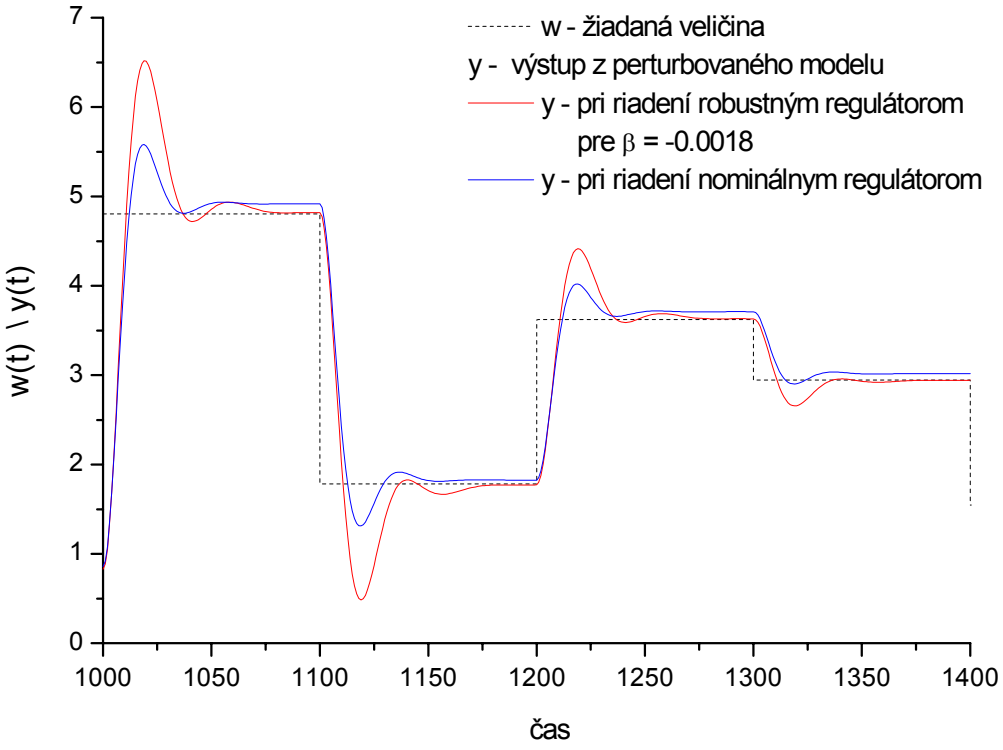


Obr. 2.11 Priebehy riadiacej veličiny regulátora

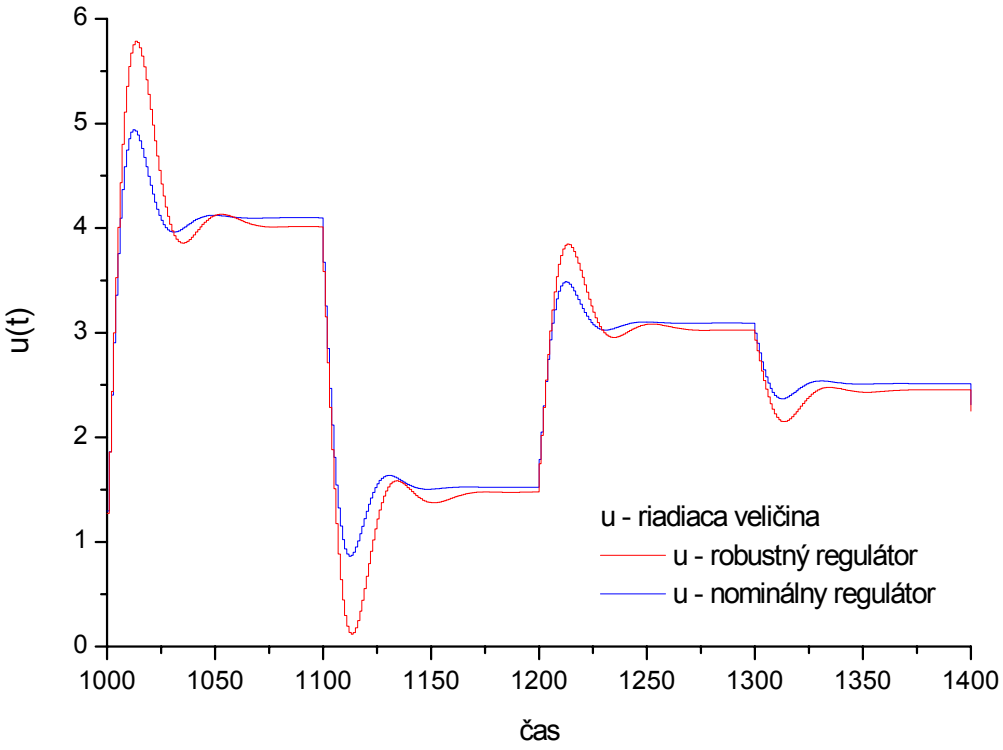
2.1.2.3 Robustné riadenie perturbovaného systému

V kapitole 2.1.2.2 bolo ukázané, že inverzný neurónový regulátor nie je schopný odstrániť trvalú regulačnú odchýlku pri riadení perturbovaných systémov, ktorých zosilnenie sa odlišuje o nominálneho zosilnenia o viac ako ±1%. Tento nedostatok regulátora je možné odstrániť pomocou adaptácie prahového koeficienta výstupného neurónu, na základe hodnoty TRO.

Získavame tak robustný inverzný neurónový regulátor, ktorý je schopný odstrániť TRO. Adaptačný koeficient $\beta = -0{,}0018$ bol navolený tak, aby sa pri riadení odstránila TRO, ale aby sa pritom systém príliš nerozkmital. Veľká absolútna hodnota adaptačného koeficientu síce urýchľuje adaptáciu, ale na úkor rozkmitania systému, malá hodnota naopak spomaluje adaptáciu. Porovnanie adaptívneho riadenia perturbovaného systému so zosilnením odlišujúcim sa od nominálneho zosilnenia o +10% a riadenia nominálnym regulátorom je na obr.2.12. Priebehy riadiacej veličiny adaptívneho a nominálneho regulátora počas riadenia perturbovaného systému sú na obr.2.13.



Obr. 2.12 Porovnanie riadenia systému s perturbovaným zosilnením pomocou nominálneho a robustného regulátora



Obr. 2.13 Porovnanie priebehov riadiacej veličiny oboch typov regulátora pri riadení perturbovaného systému

Z priebehu riadenia vidieť, že pri riadení nominálneho systému navrhnutým inverzným neurónovým regulátorom, ten vykazoval dobré regulačné vlastnosti (obr.2.6) a v regulačnom obvode nezanechával TRO. Pri riadení perturbovaného systému, kedy sa najprv perturboval koeficient tlmenia do oblasti periodicity (obr.2.8), sa postupným znižovaním koeficientu tlmenia zvyšovali oscilácie systému. Regulátor vykazoval dobré regulačné vlastnosti po hodnotu koeficientu tlmenia ξ=0.6, pri nižších hodnotách sa systém stával nestabilným. Pri perturbácii zosilnenia systému (obr.2.10) v rozsahu ±50% nominálnej hodnoty už neadaptovaný regulátor nedokázal uriadiť perturbovaný systém na žiadanú hodnotu a zanechával v regulačnom obvode výraznú trvalú regulačnú odchýlku. Príčinou je značný rozdiel medzi inverzným modelom perturbovaného systému a inverzným modelom nominálneho systému predstavujúceho inverzný neurónový regulátor. Tento nedostatok sa odstraňuje adaptáciou výstupu neurónovej siete cez prahový neurón. Takto získaný regulátor po skončení adaptácie potom predstavuje presnú inverziu riadeného systému a je schopný uriadiť aj perturbovaný systém bez zanechania TRO v regulačnom pochode (obr.2.12).

2.2 Identifikácia a riadenie nelineárneho systému

Pri identifikácii a riadení nelineárneho procesu sa ako príklad nelineárneho systému použil model bioprocesu popisujúci rast kultúry Saccharomyces cerevisae na glukóze v biochemickom prietokovom reaktore.

2.2.1 Matematický model bioprocesu

Uvedený matematický model bol prebraný z práce [11].

Použité symboly :

c_i	$mol.l^{-1}$	koncentrácia
D	h^{-1}	zrieďovacia rýchlosť
f_{ic}		faktor indukcie alebo potlačenia
k_i	$mol.l^{-1}$	saturačná konštanta
k_I	$mol.l^{-1}$	inhibičná konštanta
k_m	$mol.l^{-1}$	saturačná konštanta substrátu
k_n	$mol.l^{-1}$	saturačná konštanta glukózy
$k_L a$	h^{-1}	objemový koeficient prenosu na objem kvapaliny
m	$mol.mol^{-1}$	distribučný koeficient plyn – kvapalina
Q_i	$mol.C.mol.DW^{-1}h^{-1}$	špecifická rýchlosť spotreby alebo produkcie
q	$l.h^{-1}$	prietoková rýchlosť
Y_{ij}	$mol.mol^{-1}$	koeficient výťažku
V	l	objem
μ	h^{-1}	špecifická rýchlosť rastu
τ_i	h	časová konštanta

zoznam indexov :

c	oxid uhličitý CO_2	max	maximum
e	etanol	o	kyslík
g	plynná fáza	ox	oxidácia
I	inhibícia	p	parameter symetrie rastu
i	komponent i	pr	produkcia
in	vstup	red	redukcia
l	kvapalina	s	substrát
lim	limitná kapacita	x	biomasa

Kinetický model procesu

Mechanizmus	**Rovnica**
Spotreba glukózy	$Q_s = Q_{s,\max}\frac{c_s}{k_s + c_s}$
Oxidačná kapacita	$Q_{o,\lim} = Q_{o,\max}\frac{c_o}{k_o + c_o}$
Oxidačný metabolizmus glukózy	$Q_{s,ox} = \min\begin{cases} Q_s \\ Y_{os}.Q_{o,\lim} \end{cases}$
Redukčný metabolizmus glukózy	$Q_{s,red} = Q_s - Q_{s,ox}$
Spotreba etanolu	$Q_e = Q_{e,\max}\frac{c_e}{k_e + c_e}\frac{k_I}{k_I + c_s}$
Oxidačný metabolizmus etanolu	$Q_{e,ox} = \min\begin{cases} Q_e \\ Y_{oe}.\left(Q_{o,\lim} - Q_{s,ox}.Y_{so}\right) \end{cases}$
Produkcia etanolu	$Q_{s,pr} = Y_{se}.Q_{s,red}$
Rast	$\mu = Y_{sx}^{ox}.Q_{s,ox} + Y_{sx}^{red}.Q_{s,red} + Y_{ex}.Q_e$
Produkcia CO_2	$Q_c = Y_{sc}^{ox}.Q_{s,ox} + Y_{sc}^{red}.Q_{s,red} + Y_{ec}.Q_e$
Spotreba O_2	$Q_o = Y_{so}.Q_{s,ox} + Y_{eo}.Q_{s,ox}$
Prenos O_2	$Na = k_L a\left(\frac{c_g}{m} - c_o\right)$

Maximálna rýchlosť spotreby $$\frac{dQ_{i,\max}}{dt} = \frac{1}{\tau_i}\left(Q^p_{i,\max} f_{ic} - Q_{i,\max}\right)$$

indukčné alebo represívne faktory sú definované nasledovne :

$$f_{oc} = \frac{c_o}{k_o + c_o}\frac{2.c_s + c_e}{k_m + 2.c_s + c_e}$$

$$f_{sc} = \frac{c_s}{k_n + c_s}$$

$$f_{ec} = \frac{c_e}{k_e + c_e}\frac{k_I}{k_I + c_s}\frac{c_o}{k_o + c_o}$$

Dynamický model procesu

Koncentrácia biomasy

$$\frac{dc_x}{dt} = D_l\left(c_{x,in} - c_x\right) + \mu \cdot c_x \tag{2.1}$$

Koncentrácia glukózy (substrátu)

$$\frac{dc_s}{dt} = D_l\left(c_{s,in} - c_s\right) - Q_s c_x \tag{2.2}$$

Koncentrácia etanolu (produktu)

$$\frac{dc_e}{dt} = D_l\left(c_{e,in} - c_e\right) + \left(Q_{e,pr} - Q_e\right)c_x \tag{2.3}$$

Koncentrácia oxidu uhličitého

$$\frac{dc_c}{dt} = D_g\left(c_{c,in} - c_c\right) - Q_c c_x \tag{2.4}$$

Koncentrácia rozpusteného kyslíka

$$\frac{dc_o}{dt} = D_g\left(c_{o,in} - c_o\right) + Na - Q_o c_x \tag{2.5}$$

Koncentrácia plynnej fázy kyslíka

$$\frac{dc_g}{dt} = D_g\left(c_{g,in} - c_s\right) - Na\frac{V_l}{V_g} \tag{2.6}$$

kde $D_l = \frac{q_l}{V_l}$ a $D_g = \frac{q_g}{V_g}$

počiatočné koncentrácie :

$c_x(0) = 3.174722.10^{-3} \quad Cmol.l^{-1}$ $c_c(0) = 3.793648.10^{-4} \quad mol.l^{-1}$

$c_s(0) = 1.212480.10^{-4} \quad mol.l^{-1}$ $c_o(0) = 5.111012.10^{-6} \quad mol.l^{-1}$

$c_e(0) = 1.863964.10^{-5} \quad mol.l^{-1}$ $c_g(0) = 1.879319.10^{-4} \quad mol.l^{-1}$

2.2.2 Identifikácia nelineárneho systému

Pri identifikácii nelineárneho systému bioprocesu sa použila sieť so štruktúrou {4,3,1}, teda s počtom neurónov v skrytej vrstve o jeden väčším ako u lineárneho systému. Väčší počet neurónov v skrytej vrstve mal urýchliť trénovanie siete a umožniť tak dosiahnutie menšej chyby trénovania pri menšom počte epoch.

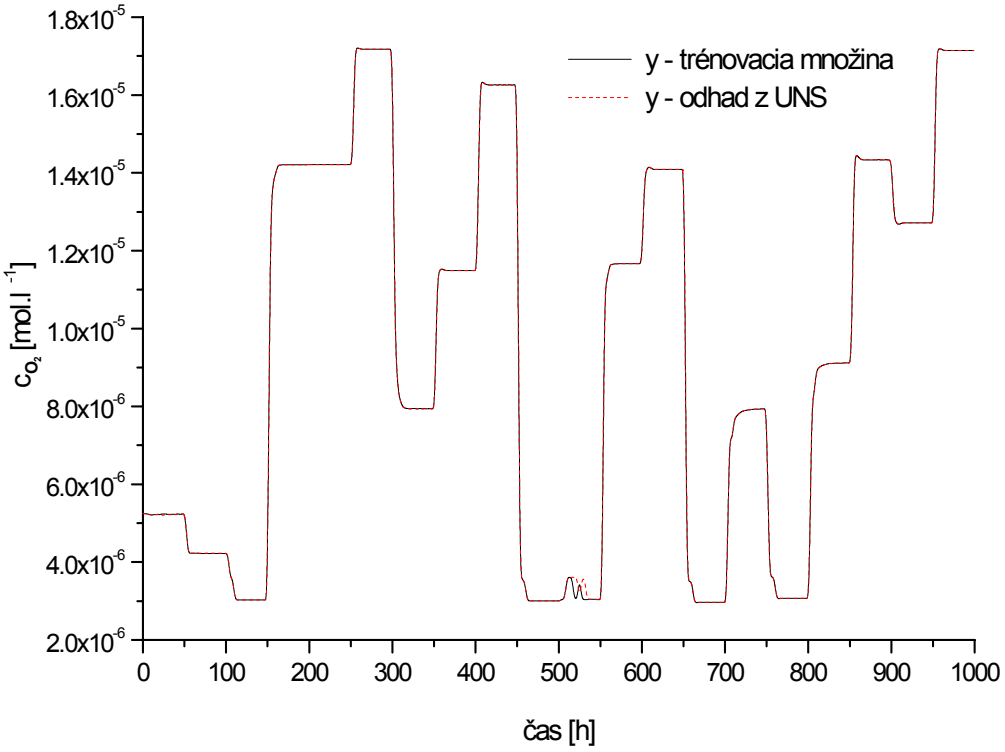
Trénovacia množina dát sa získala simuláciou nelineárneho systému, počas ktorej sa na vstup do systému priviedlo viacero náhodných skokových zmien. Trénovacia množina obsahovala 1500 vzoriek údajov predstavujúcich vstupno-výstupný opis systému. Vstupom do systému počas získavania informácií o systéme bola hodnota zrieďovacej rýchlosti plynnej fázy $D_g(t)$. Zodpovedajúcim výstupom systému bola hodnota koncentrácie rozpusteného kyslíka $DO(t+1)$.

Trénovanie siete prebiehalo off-line. Pri samotnom trénovaní sa na vstup do siete privádzala aktuálna hodnota výstupnej veličiny z modelu bioprocesu $DO(t)$, jej prvá a druhá história $DO(t-1)$ *resp.* $DO(t-2)$ a aktuálna hodnota vstupu do systému $D_g(t)$. Ako zodpovedajúci výstup sa predkladala hodnota $DO(t+1)$ ako odozva systému na vstup $D_g(t)$.

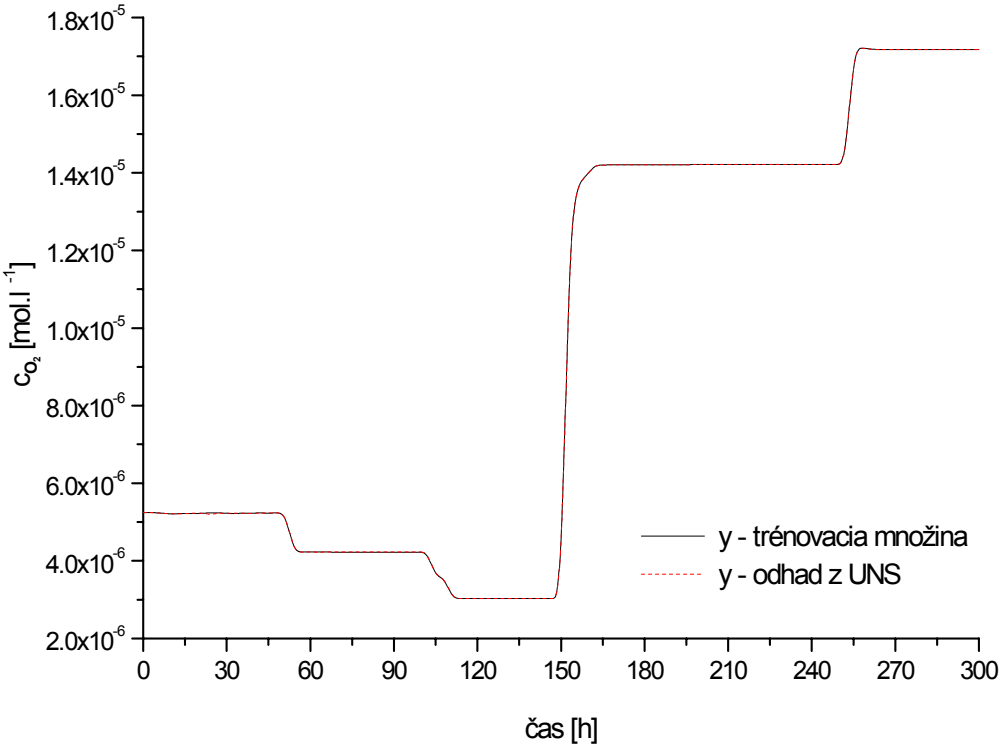
Hodnoty výstupnej veličiny, koncentrácie rozpusteného kyslíka, z modelu bioprocesu sa pri získavaní trénovacej množiny normovali do okolia 1, pretože sa pohybovali rádovo okolo hodnoty 10^{-5} $mol.l^{-1}$. Výsledkom normovania bolo spresnenie numerických výpočtov derivácií vykonávaných počas trénovania siete.

Po ukončení trénovania sieť predstavovala dopredný model systému. Chyba epochy klesla po 1000 iteráciách na hodnotu blízku 1.10^{-3}. Na obr.2.14 resp. obr.2.15 je

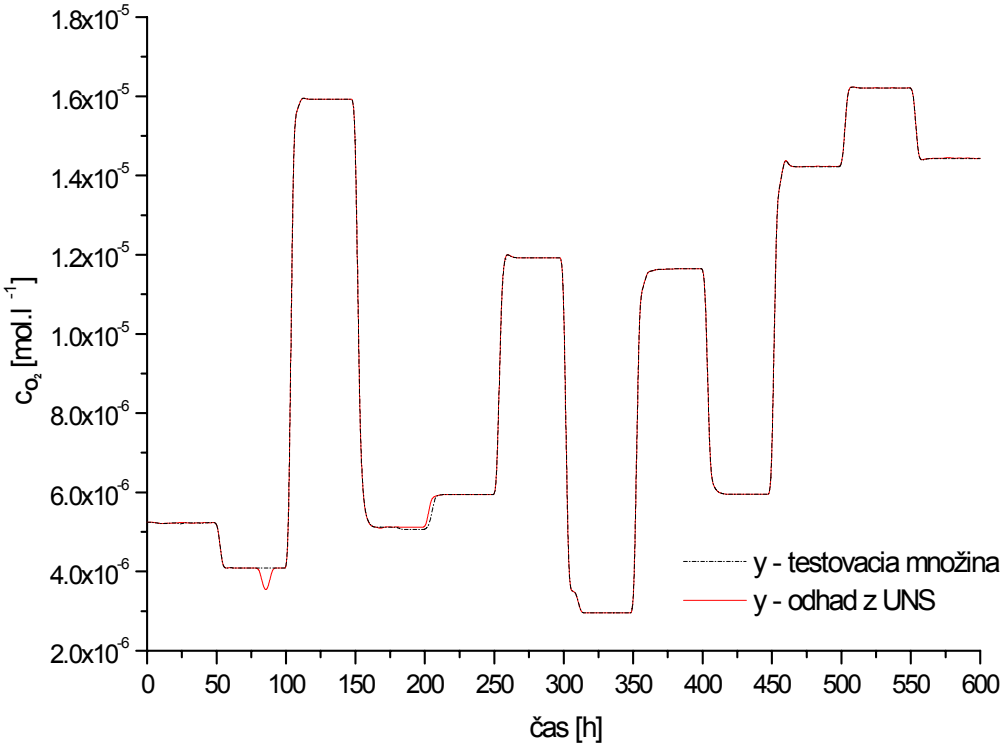
porovnanie odhadu výstupu z natrénovanej siete s trénovacími dátami, na obr.2.16 je porovnanie výstupu z natrénovanej siete s testovacími dátami.



Obr. 2.14 Porovnanie odhadu výstupu z natrénovanej siete s trénovacími dátami



Obr. 2.15 Časť porovnania odhadu výstupu z natrénovanej siete s trénovacími dátami

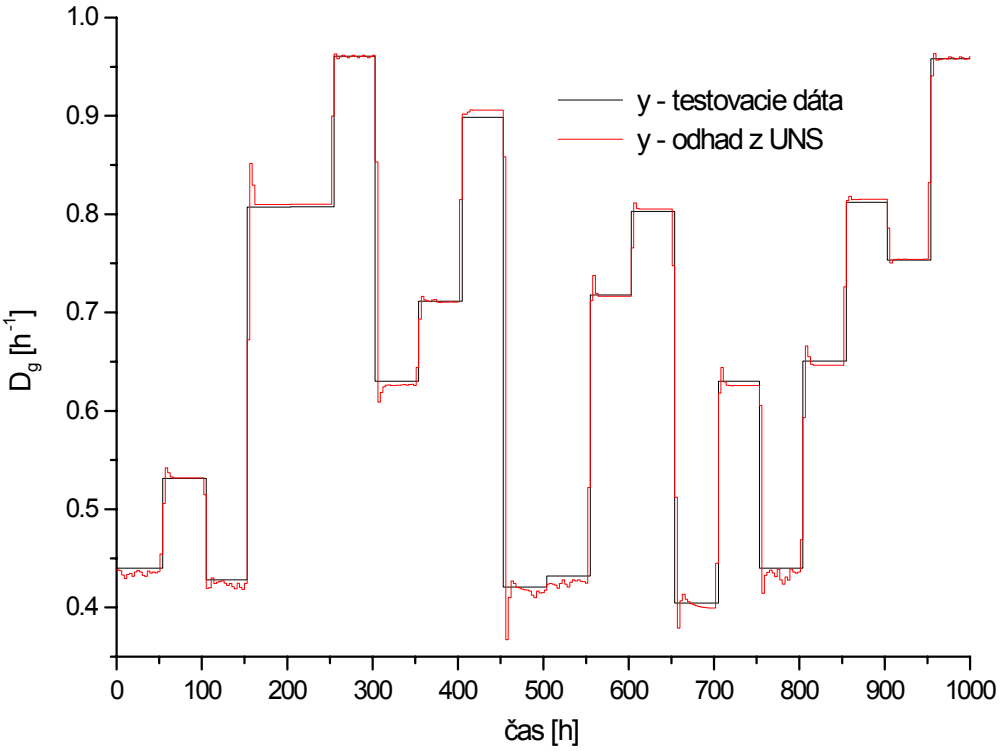


Obr. 2.16 Porovnania odhadu výstupu z natrénovanej siete s testovacími dátami

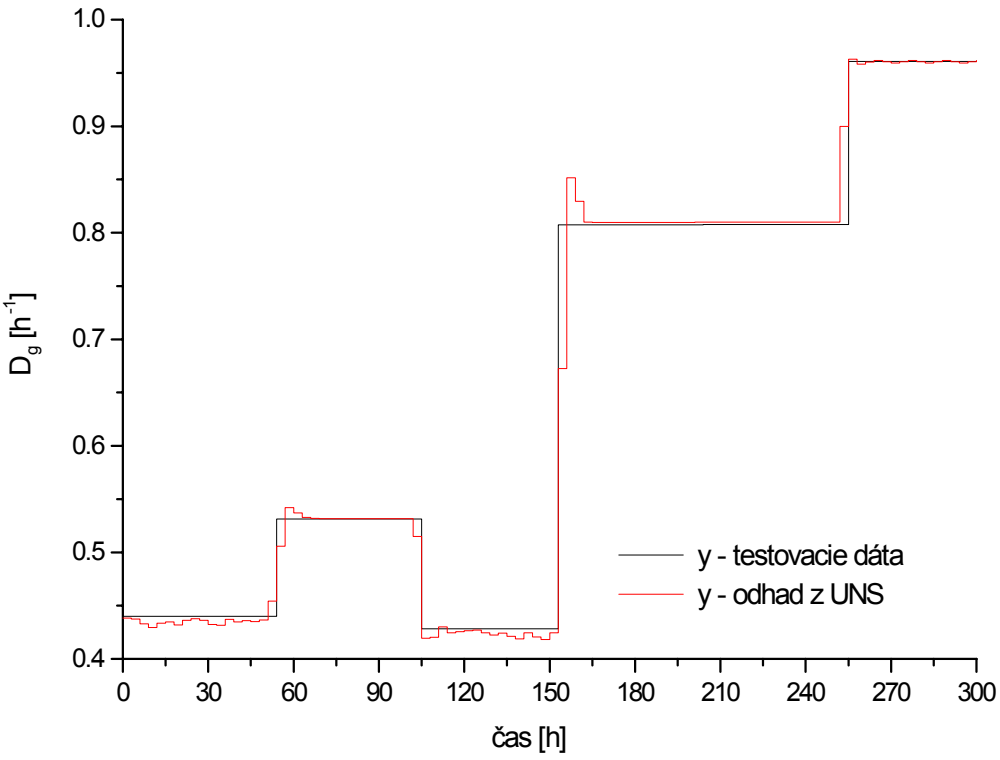
2.2.3 Riadenie nelineárneho systému neurónovým regulátorom

Na riadenie nelineárneho systému sa použil inverzný neurónový regulátor so štruktúrou siete {4,4,1}. Sieť bola trénovaná off-line. Trénovacia množina dát sa získala simuláciou nelineárneho systému, pričom sa na jeho vstup počas simulácie privádzali náhodné skokové zmeny parametra $D_g(t)$. Trénovacia množina obsahovala 1500 vzoriek údajov.

Na vstup do trénovanej siete sa privádzal výstup z modelu *DO(t),* jeho prvá a druhá história *DO(t-1)* resp. *DO(t-2)*, a hodnota výstupu *DO(t+1)* ako odozva systému na vstup $D_g(t)$. Ako zodpovedajúci výstup zo siete sa predkladala aktuálna hodnota vstupu do systému $D_g(t)$. Podobne ako pri identifikácii bioprocesu aj pri získavaní inverzného modelu bolo treba výstup z modelu, koncentráciu rozpusteného kyslíka, normovať do okolia 1.Po ukončení trénovania sieť predstavovala inverzný model systému. Chyba epochy klesla po 1000 iteráciách na hodnotu blízku 6.10^{-3}. Na obr.2.17 a obr.2.18 je porovnanie resp. časť porovnania odhadu výstupu z natrénovanej siete s testovacími dátami.

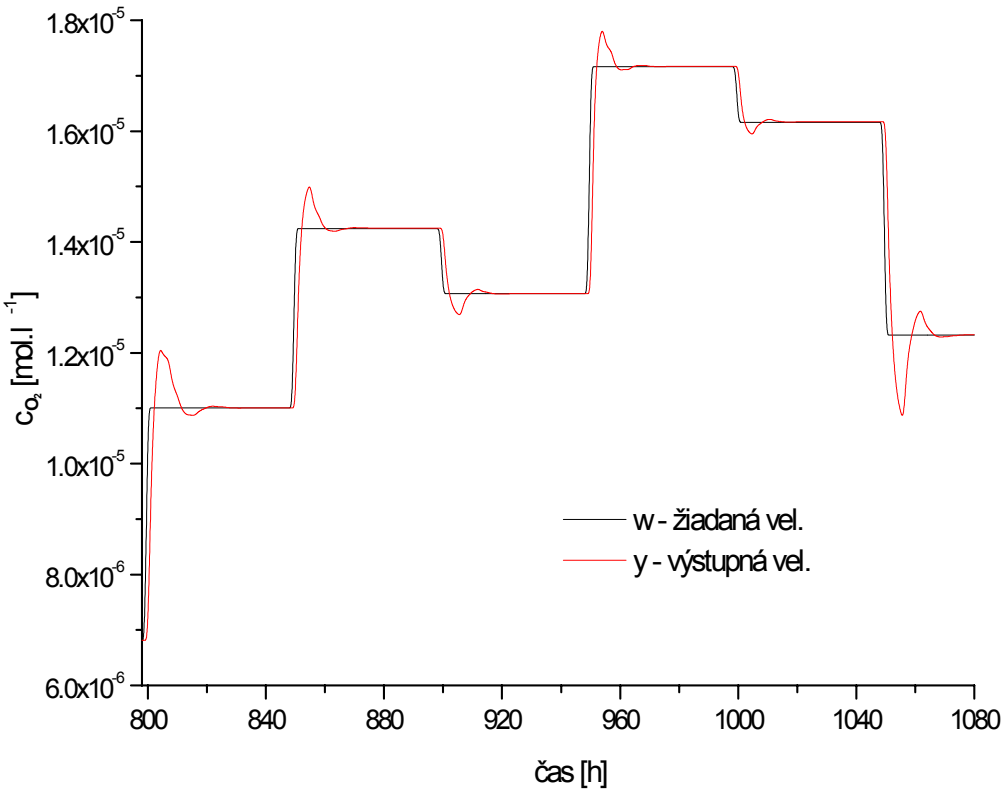


Obr. 2.17 Porovnanie odhadu výstupu z natrénovanej siete s testovacími dátami

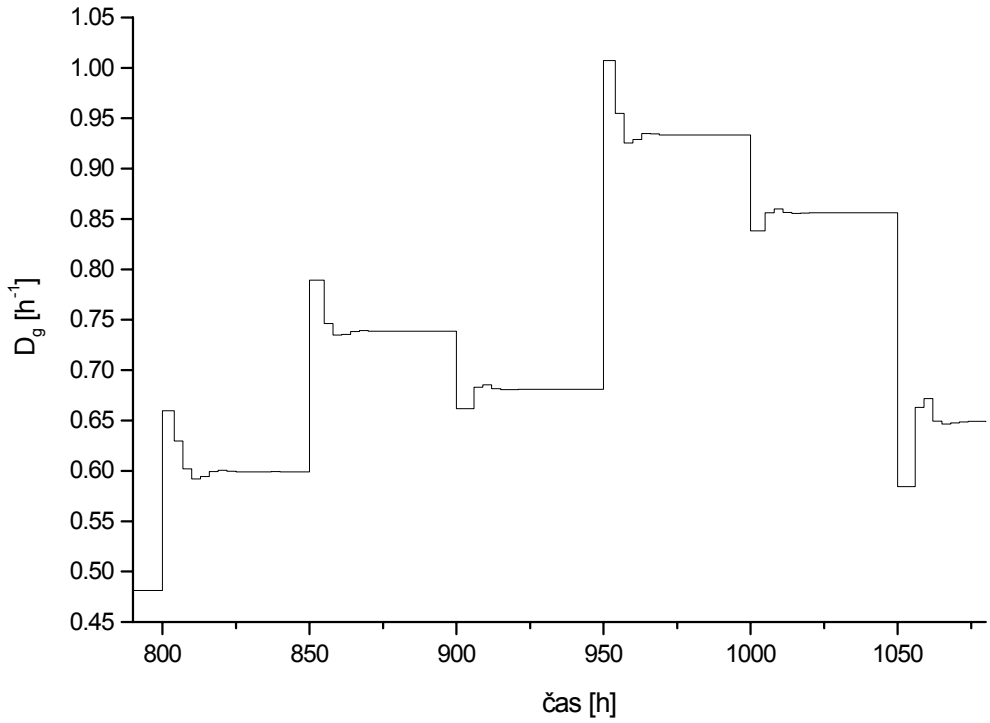


Obr. 2.18 Časť porovnania odhadu výstupu z natrénovanej siete s testovacími dátami

Natrénovaný inverzný model sa následne použil ako inverzný neurónový regulátor na riadenie bioprocesu. Regulátor pracoval s periódou vzorkovania 0,5 h. V prípade riadenia nominálneho systému takto natrénovaný inverzný regulátor postačoval. Na obr.2.19 je časť priebehu riadenia nominálneho systému pomocou inverzného neurónového regulátora a na obr.2.20 je priebeh riadiacej veličiny regulátora.



Obr. 2.19 Priebeh riadenia nelineárneho systému pomocou neurónového regulátora predstavujúceho inverzný model systému



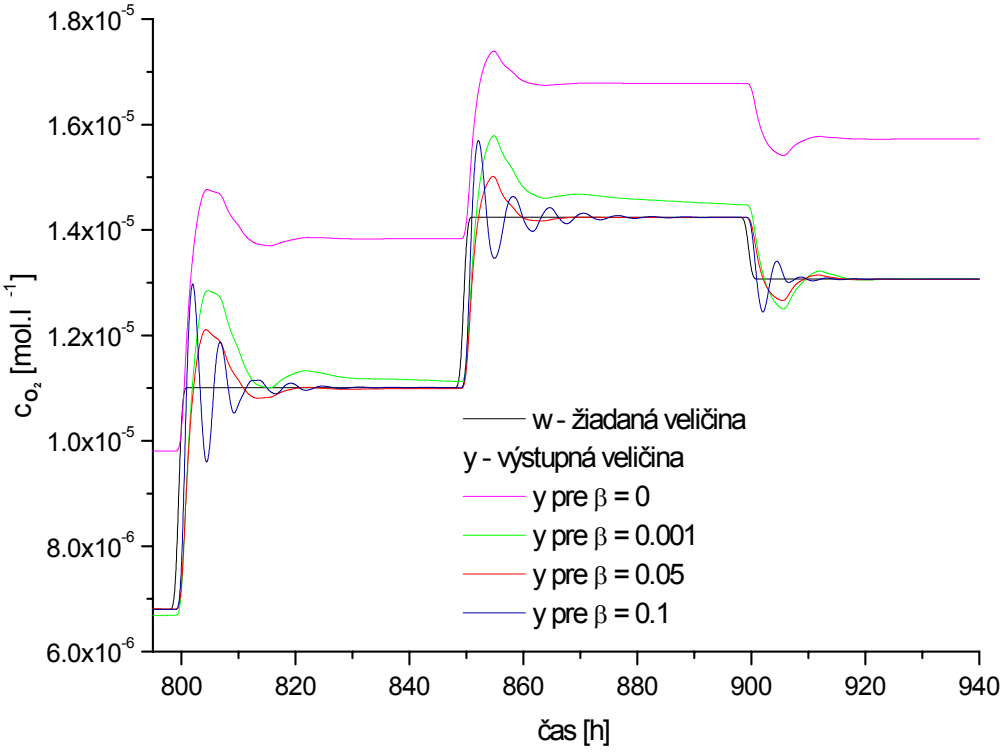
Obr. 2.20 Priebeh riadiacej veličiny inverzného neurónového regulátora

V prípade riadenia perturbovaného systému bioprocesu natrénovaný inverzný neurónový regulátor nepostačoval (obr.2.21 pre $\beta = 0$) a zanechával v regulačnom pochode výraznú trvalú regulačnú odchýlku. Preto sa kombináciou inverzného regulátora z kapitoly 2.2.3 a adaptačného zákona opísaného rovnicou (1.24) vytvoril robustný inverzný neurónový regulátor.

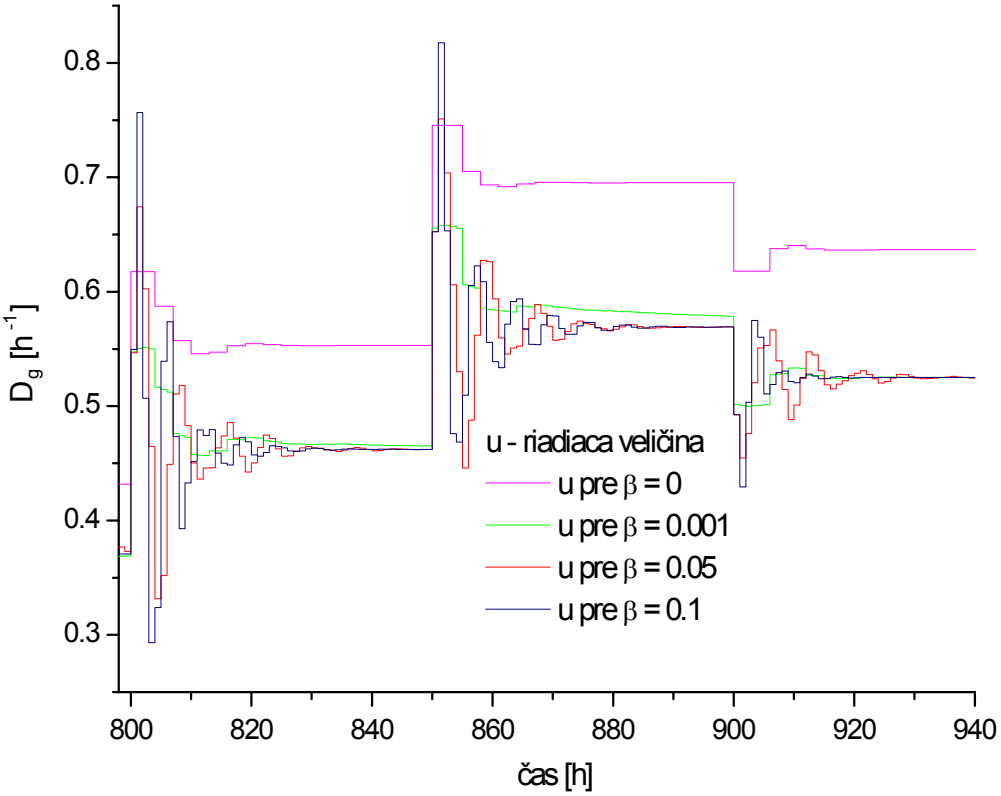
2.2.3.1 Robustný neurónový regulátor s jednoduchým integračným členom

Ako prídavný adaptér k inverznému neurónovému regulátoru sa použil *jednoduchý integračný člen* a regulátor sa adaptoval cez bias výstupného neurónu siete. Ako porucha sa privádzala skoková zmena vstupnej hodnoty koncentrácie substrátu $c_{s,in}$, ktorej hodnoty sa pohybovali v rozsahu $\pm$ 20% pôvodnej hodnoty.

Na obr.2.22 sú priebehy adaptívneho riadenia perturbovaného systému neurónovým regulátorom pre rôzne hodnoty adaptačného koeficientu β. Na obr.2.23 sú zodpovedajúce priebehy riadiacej veličiny regulátora.



Obr. 2.21 Priebehy adaptívneho riadenia neurónovým regulátorom pre rôzne hodnoty adaptačného koeficientu β



Obr. 2.22 Priebehy riadiacej veličiny regulátora pre rôzne hodnoty β

2.2.3.2 Fuzzy-neurónový regulátor

Predchádzajúce simulačné experimenty ukázali, že inverzný neurónový regulátor nedokáže odstrániť trvalú regulačnú odchýlku. Ladenie výstupu neurónového regulátora cez bias výstupného neurónu ukázalo, že takýmto spôsobom je možné odstrániť regulačnú odchýlku avšak za cenu zhoršenia kvality regulácie. Malé hodnoty adaptačného zosilnenia značne predlžujú dobu regulácie, veľké hodnoty zasa spôsobujú kmitavosť regulačného pochodu a značné preregulovanie. Toto je možné odôvodniť nasledovnou úvahou:

Predpokladajme systém so zosilnením blízkym nominálnemu zosilneniu. Po skokovej zmene žiadanej veličiny inverzný regulátor vygeneruje akčný zásah tak, aby sa výstup systému dostal do blízkosti žiadanej veličiny. Môžeme teda povedať, že generuje „takmer správny“ akčný zásah. Ak je však zapojená adaptácia cez bias výstupného neurónu, adaptér – integrátor práve v čase po skokovej zmene žiadanej veličiny najviac prispieva k zmene výstupu regulátora, pretože regulačná odchýlka dosahuje najväčšie hodnoty. Toto má však za následok, že „takmer správny“ akčný zásah je korigovaný o veľkú hodnotu a pochopiteľne spôsobí veľké preregulovanie.

Integrátor teda spočiatku naintegruje veľkú hodnotu, ktorá sa neskôr už len pomaly mení v dôsledku poklesu hodnoty regulačnej odchýlky. Z uvedenej úvahy teda plynie skutočnosť, že nie je vhodné, aby inverzný regulátor a adaptér naraz menili akčnú veličinu, najmä v čase po skokovej zmene.

Z úvahy teda vyplýva, že adaptér by nemal adaptovať, keď dochádza k prudkým zmenám regulačnej odchýlky. A zasa naopak, ak sa zmena regulačnej odchýlky bude blížiť k nule, adaptér by mal začať pracovať. Adaptér s požadovanými vlastnosťami je možné efektívne navrhnúť ako fuzzy systém, pretože ak predchádzajúce dve vety predstavujú zákon riadenia pre fuzzy regulátor, tak ich následná implementácia do fuzzy logiky je pomerne jednoduchá.

Pri tvorbe pravidiel a funkcií príslušnosti fuzzy regulátora sa teda prihliada na základné dva princípy a to:

1. adaptácia má byť minimálna po skokovej zmene hodnoty žiadanej veličiny (parameter ***I*** má byť minimálny)
2. adaptácia má byť maximálna pri vzniku (pretrvávaní) trvalej regulačnej odchýlky (parameter ***I*** má byť maximálny)

Fuzzy regulátor bol vytvorený pomocou grafického užívateľského prostredia Fuzzy Toolboxu v prostredí MATLAB. Vstupnými premennými fuzzy systému vyhovujúcemu týmto požiadavkám sú premenná ***e*** ako odchýlka medzi žiadanou a výstupnou veličinou a ***de*** ako jej derivácia. Výstupnou premennou je integračný koeficient ***I***.

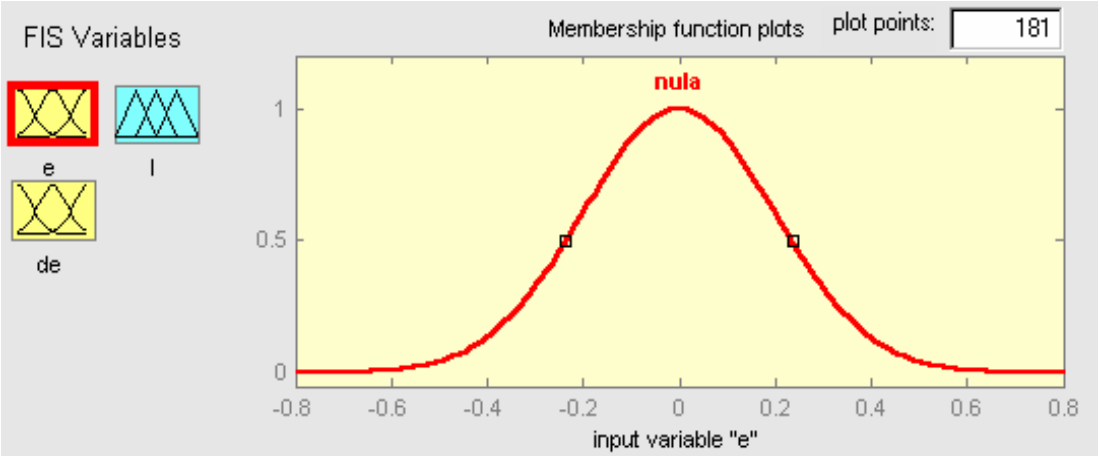
Pre tieto jazykové premenné sa definovali fuzzy množiny a to po dve pre každú zo vstupných premenných ***e*** a ***de*** a tri pre premennú ***I***. Pre vstupné premenné sa definovala Gaussova funkcia príslušnosti, pre výstupnú premennú sa zvolili triangulárna a trapezoidálna funkcia (obe funkcie sú po častiach lineárne).

Báza pravidiel pre daný fuzzy systém bola nasledovná :

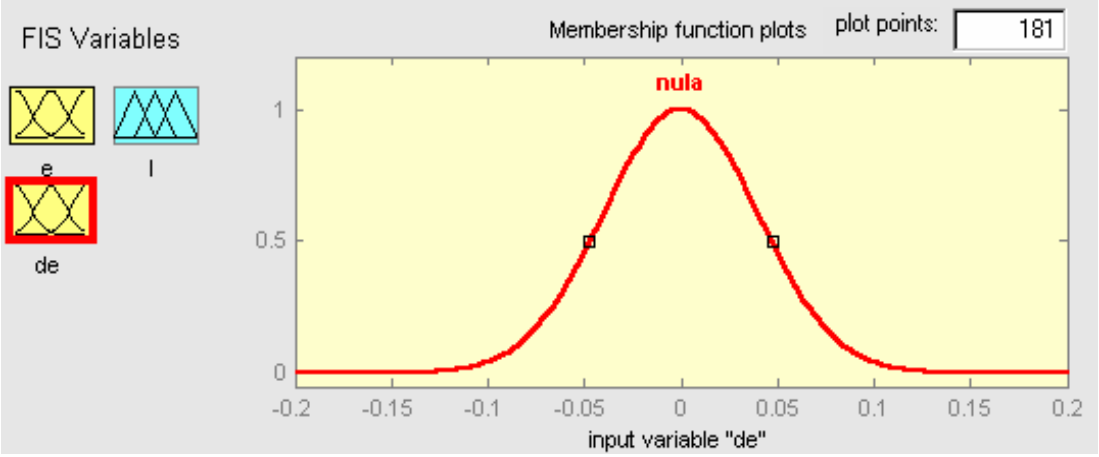
1. Ak ***e*** je nula a ***de*** je nula, potom ***I*** je max (maximálne)
2. Ak ***e*** je nula a ***de*** je „nie je nula“, potom ***I*** je stred (stredné)
3. Ak ***e*** „nie je nula“ a ***de*** je nula, potom ***I*** je stred (stredné)
4. Ak ***e*** „nie je nula“ a ***de*** „nie je nula“, potom ***I*** je min (minimálne)

pričom fuzzy množina „nie je nula“ je doplnok k fuzzy množine nula.

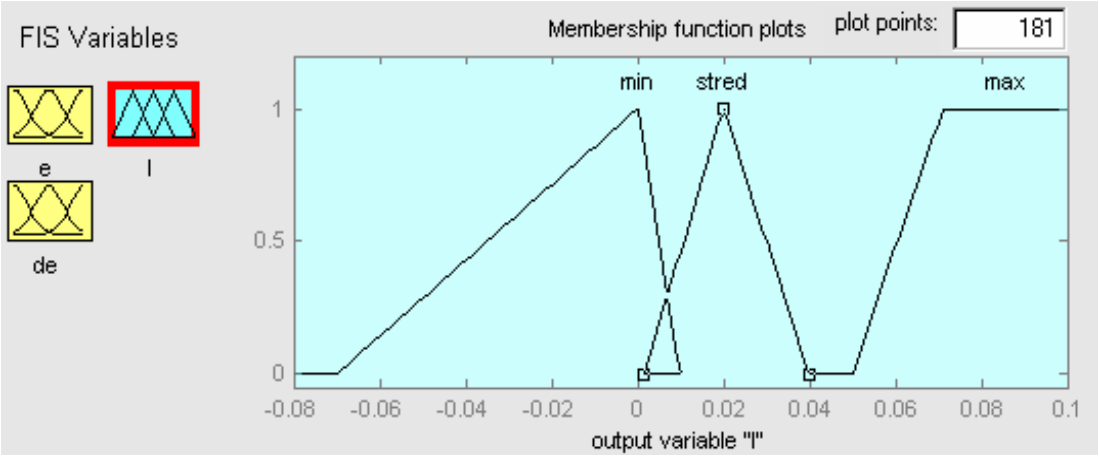
Na obr.2.24 až 2.26 sú zobrazené funkcie príslušností (FP) vstupných veličín fuzzy regulátora ***e***, ***de*** a výstupnej veličiny ***I***. Na obr.2.27 je grafické zobrazenie výstupu navrhnutého fuzzy regulátora.



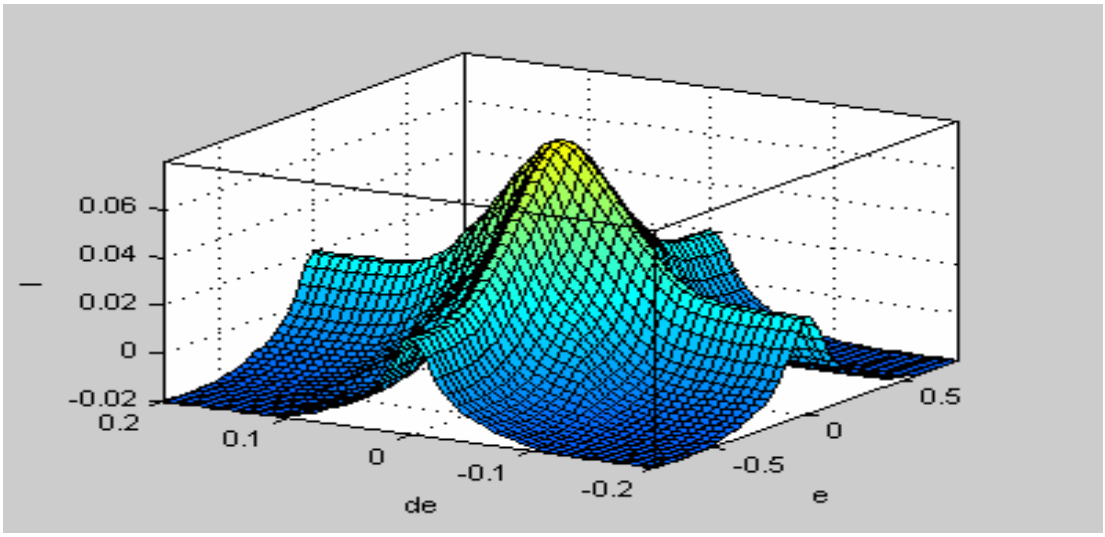
Obr. 2.23 FP fuzzy množiny nula pre vstupnú premennú ***e*** (*gaussmf*[0.2 0])



Obr. 2.24 FP fuzzy množiny nula pre vstupnú premennú ***de*** (*gaussmf*[0.04 0])

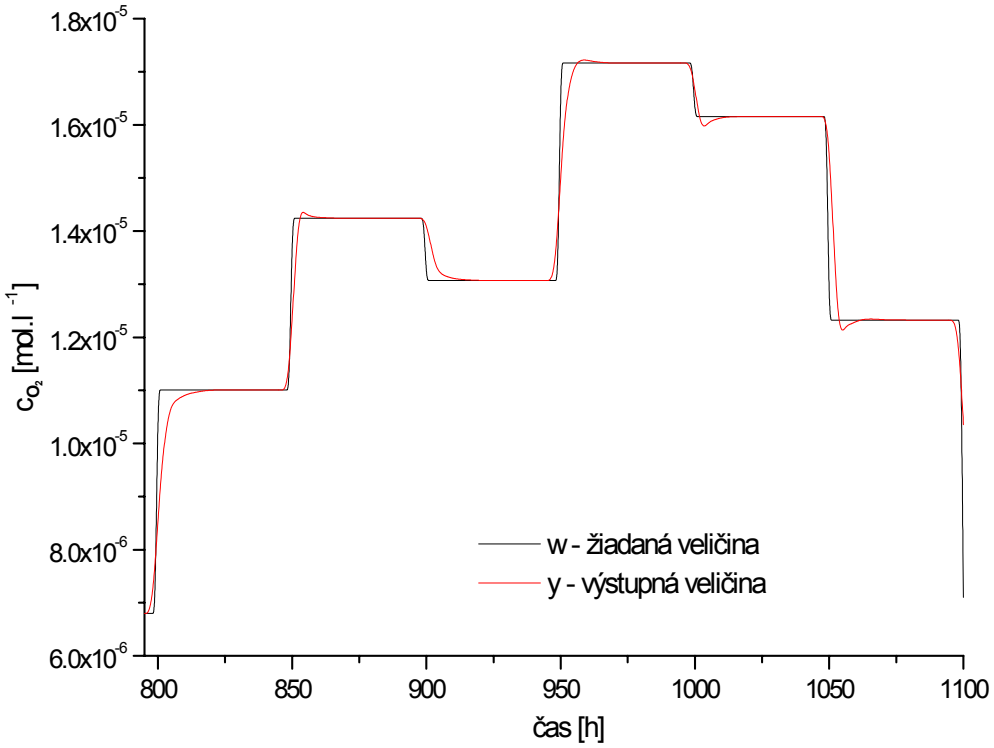


Obr. 2.25 FP fuzzy množiny min (*trimf*[-0.07 0 0.01]), stred (*trimf*[[0.00145 0.02 0.04]) a max (*trapmf*[[0.05 0.071 0.0843 Inf]) pre výstupnú veličinu ***I***

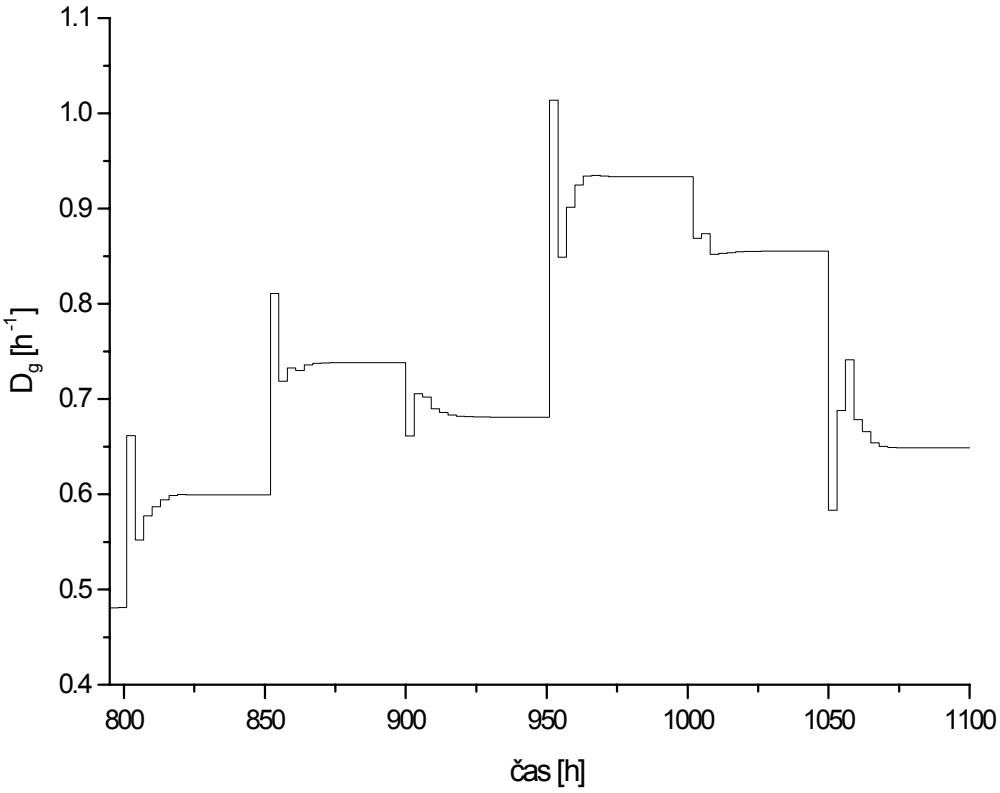


Obr. 2.26 Grafické zobrazenie výstupu fuzzy regulátora

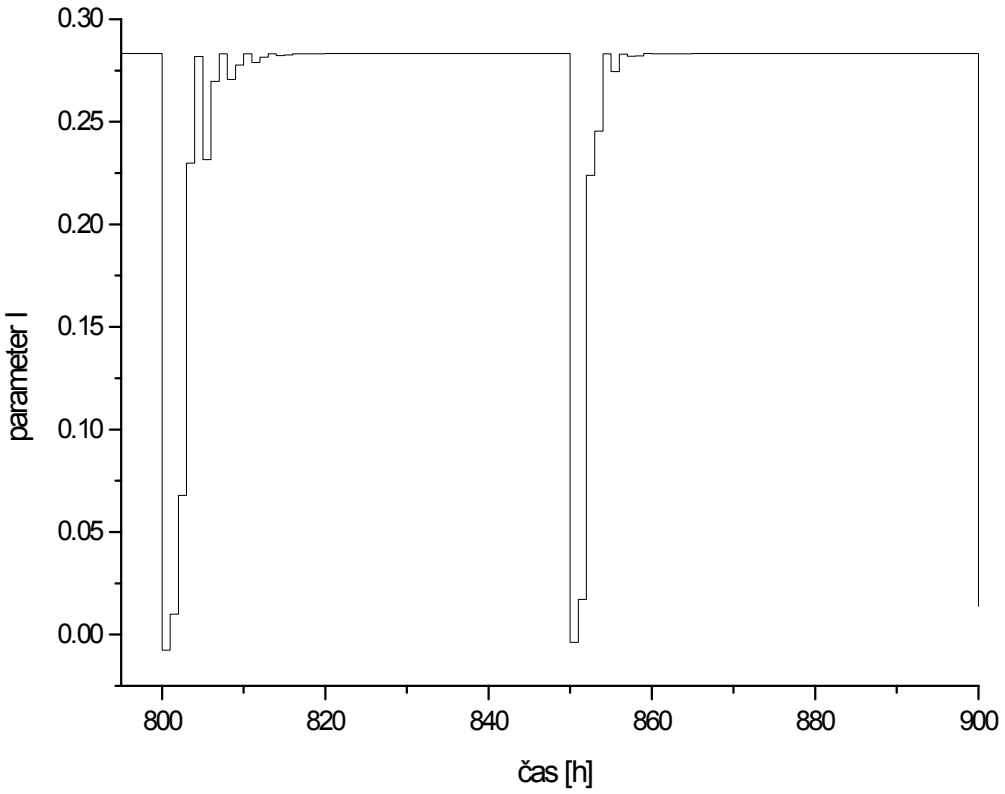
Fuzzy-neurónový regulátor sa najskôr použil na riadenie nominálneho systému bioprocesu, na obr.2.28 je priebeh riadenia nominálneho systému bioprocesu fuzzy-neurónovým regulátorom, na obr.2.29 je priebeh riadiacej veličiny fuzzy-neurónového regulátora pri tomto riadení a na obr.2.30 je vývoj parametra ***I*** pri tomto riadení.



Obr. 2.27 Priebeh riadenia nominálneho systému bioprocesu fuzzy-neurónovým regulátorom



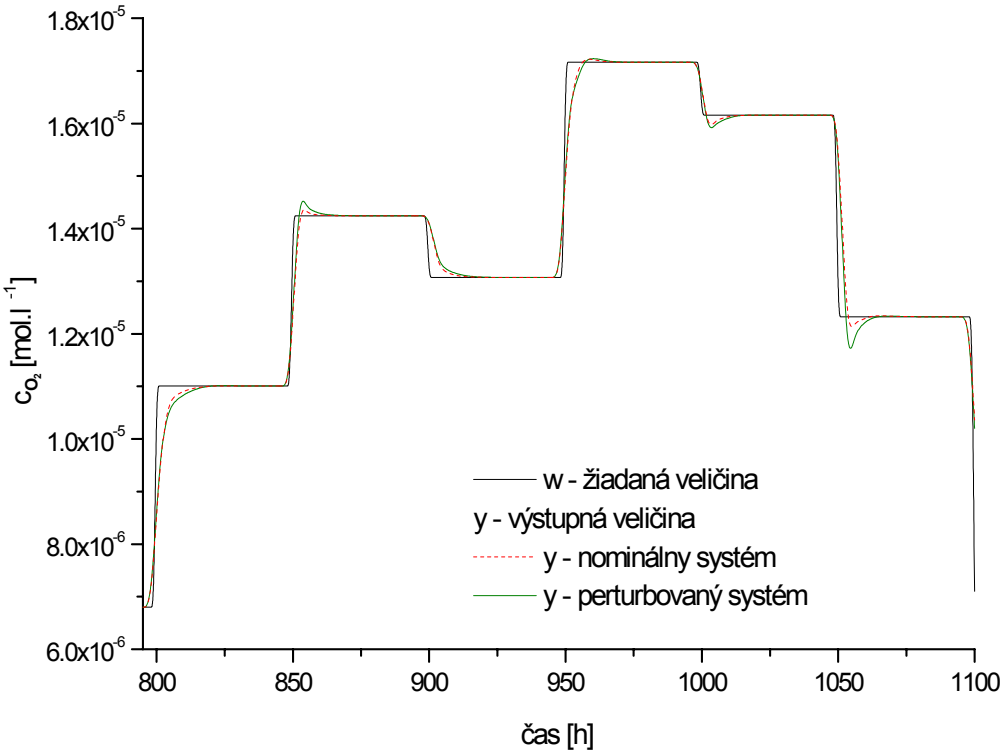
Obr. 2.28 Priebeh riadiacej veličiny fuzzy-neurónového regulátora



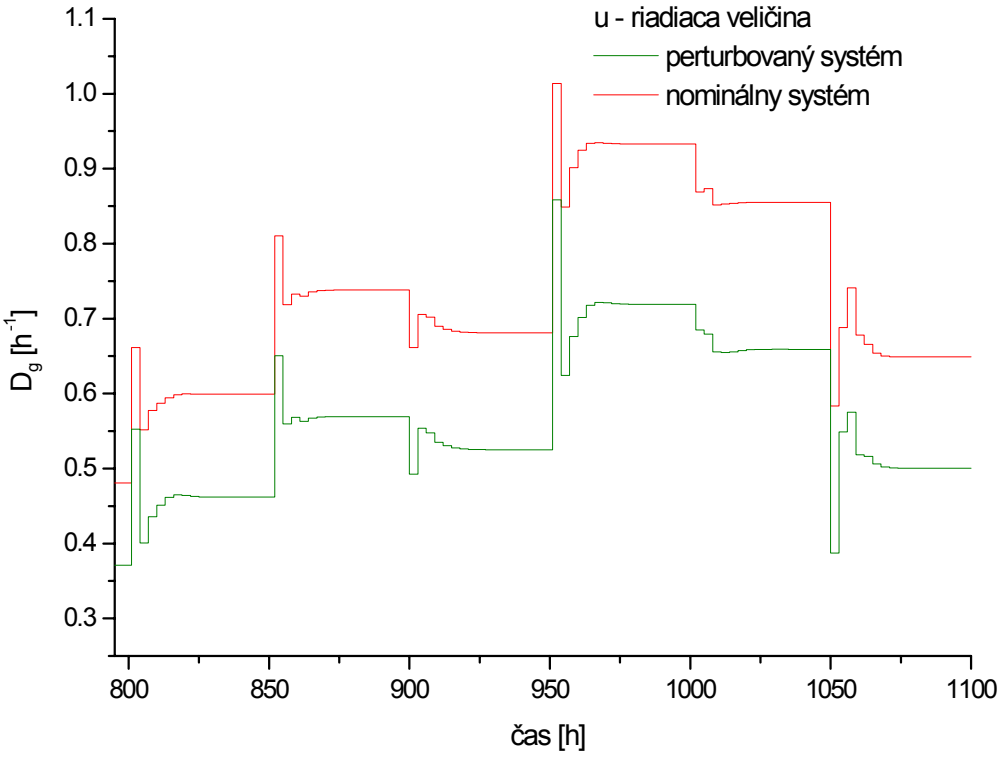
Obr. 2.29 Vývoj parametra ***I*** počas riadenia fuzzy-neurónovým regulátorom

Následne sa pomocou fuzzy-neurónového regulátora riadil perturbovaný systém bioprocesu. Ako porucha sa privádzala skoková zmena vstupnej hodnoty koncentrácie substrátu $c_{s,in}$, ktorej hodnoty sa pohybovali v rozsahu $\pm$ 20% pôvodnej hodnoty.

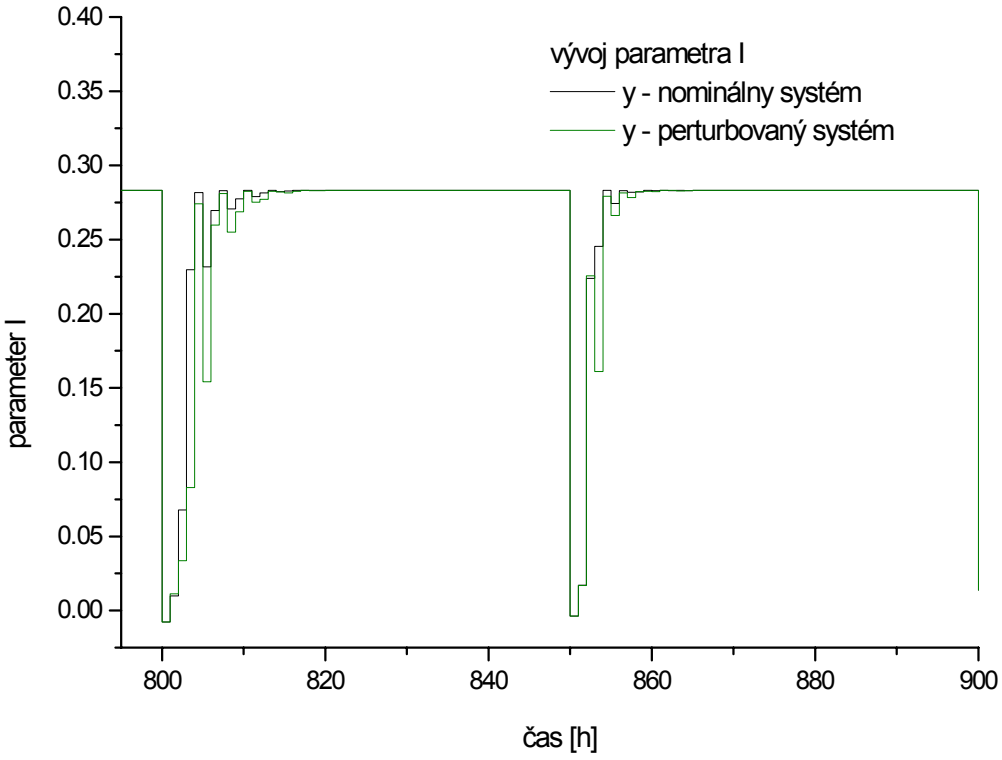
Na obr.2.31 je porovnanie priebehov riadenia nominálneho systému a perturbovaného systému bioprocesu fuzzy-neurónovým regulátorom, na obr.2.32 sú porovnané priebehy riadiacej veličiny regulátora pri riadení oboch systémov a na obr.2.33 je porovnanie vývoja parametra ***I*** počas tohoto riadenia.



Obr. 2.30 Porovnanie riadenia perturbovaného a nominálneho systému bioprocesu fuzzy-neurónovým regulátorom



Obr. 2.31 Porovnanie priebehov riadiacej veličiny fuzzy-neurónového regulátora

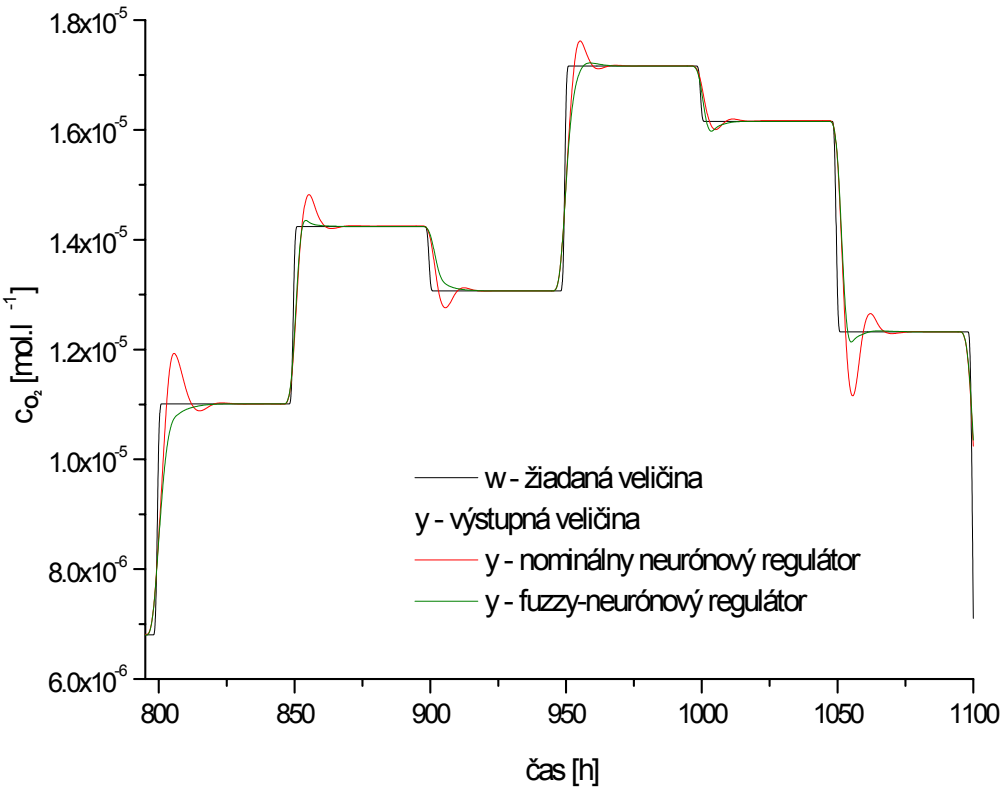


Obr. 2.32 Porovnanie vývoja parametra I počas riadenia fuzzy-neurónovým regulátorom

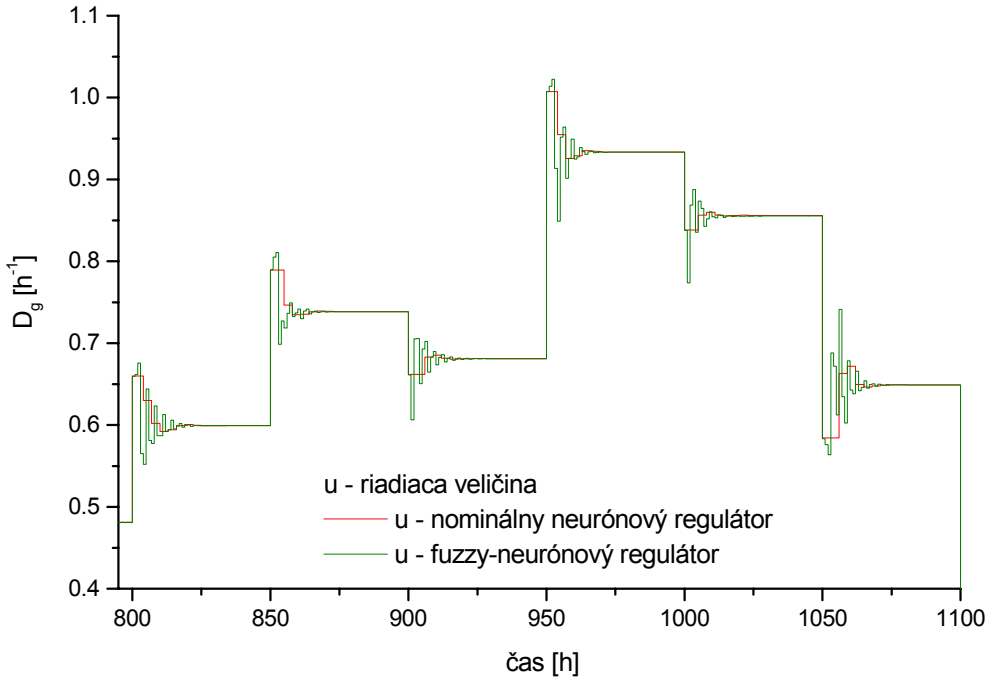
2.2.4 Porovnanie vlastností jednotlivých regulátorov pri riadení nelineárneho modelu bioprocesu

Na obr.2.34 je porovnanie priebehu riadenia nominálneho systému inverzným neurónovým regulátorom a fuzzy-neurónovým regulátorom, na obr.2.35 sú zodpovedajúce priebehy riadiacej veličiny oboch regulátorov.

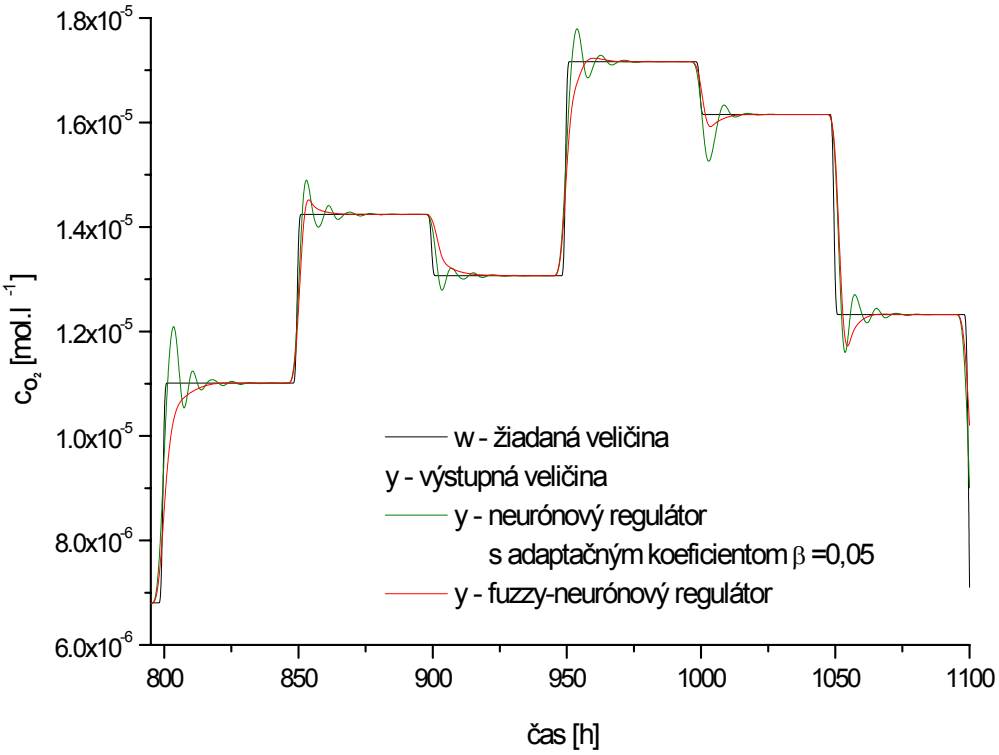
Na obr.2.36 je porovnanie priebehu riadenia perturbovaného systému neurónovým regulátorom s hodnotou adaptačného koeficientu $\beta = 0{,}05$ a fuzzy-neurónovým regulátorom, na obr.2.36 sú zodpovedajúce priebehy riadiacej veličiny oboch regulátorov.



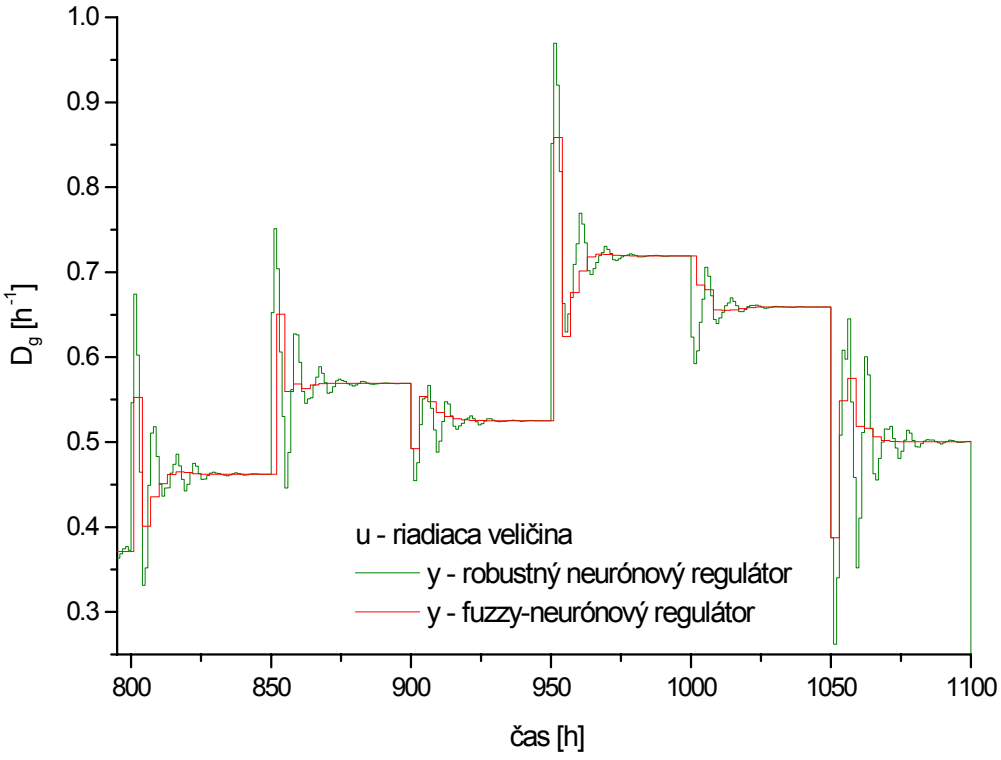
Obr. 2.33 Porovnanie riadenia nominálneho systému pomocou inverzného neurónového regulátora a fuzzy-neurónového regulátora



Obr. 2.34 Porovnanie priebehov riadiacej veličiny nominálneho a fuzzy-neurónového regulátora pri riadení nominálneho systému



Obr. 2.35 Porovnanie riadenia perturbovaného systému pomocou robustného neurónového regulátora pre $\beta = 0{,}05$ a fuzzy-neurónového regulátora



Obr. 2.36 Porovnanie priebehov riadiacej veličiny robustného neurónového regulátora a fuzzy-neurónového regulátora pri riadení perturbovaného systému

Pri nelineárnom modeli bioprocesu bol inverzný model systému natrénovaný s väčšou chybou ako u lineárneho systému. Chyba identifikácie by sa dala znížiť zvýšením počtu neurónov v skrytej vrstve alebo dlhším trénovaním siete, snahou však bolo ukázať robustnosť získaného neurónového regulátora, preto sa daný natrénovaný inverzný model už ďalej nezlepšoval. Takto natrénovaný inverzný model bol použitý ako regulátor, ktorý uspokojivo uriadil nominálny systém (obr.2.19), pri riadení perturbovaného systému už zanechával výraznú trvalú regulačnú odchýlku (obr.2.21 pre $\beta = 0$). Neurónový regulátor sa preto adaptoval a získaný robustný regulátor sa následne testoval na riadení perturbovaného systému.

Jedným spôsobom adaptácie neurónového regulátora bolo pridanie jednoduchého integračného člena, kde sa následne skúmal vplyv adaptačného koeficientu na priebeh regulovanej veličiny (obr.2.21). Výber adaptačného koeficientu sa uskutočňuje simulačne metódou „pokus-omyl“. Vhodné sa ukázali hodnoty β v rozsahu 0,01-0,05. Pri vyšších hodnotách už systém výrazne osciloval a zvyšoval sa čas regulácie, pri nižších sa v regulačnom pochode objavovala TRO. Nevýhodou takto vytvoreného robustného inverzného regulátora je jeho tendencia riadiť systém s preregulovaním aj v prípade, že riadi nominálny systém. Tento nedostatok spôsobuje adaptér s integračnými vlastnosťami, ktorý adaptuje neurónový regulátor, pokiaľ existuje nenulová regulačná odchýlka.

Následne sa skúmala možnosť využitia fuzzy-neurónového regulátora, kde sa nominálny inverzný regulátor, ktorý formálne vystupuje ako PD neurónový regulátor, rozšíri o fuzzy regulátor predstavujúci fuzzy I zložku. Fuzzy-neurónový regulátor sa najprv použil pri riadení nominálneho systému (obr.2.27). Následne sa jeho robustnosť skúšala pri riadení perturbovaného systému (obr.2.30) Výhodou takto zostaveného regulátora je možnosť potlačiť vplyv integračnej zložky pri skokovej zmene a zvýšenia jeho vplyvu pri vzniku a pretrvávaní regulačnej odchýlky.

ZÁVER

Predmetom tejto práce bolo oboznámiť sa s problematikou umelých neurónových sietí a ich možným využitím v identifikácii a riadení systémov.

UNS použité pri identifikácii a riadení systémov boli vytvorené pomocou Neural Network Toolboxu v prostredí MATLAB. Postup vytvorenia a trénovanie siete v grafickom užívateľskom prostredí (GUI) toolboxu je popísaný v kapitole 1.3 Teoretickej časti práce. Toto GUI je jednoducho ovládateľné, prehladné a poskytuje prostredie pre realizáciu a simuláciu rôznych typov sietí. Hlavnou výhodou je skutočnosť, že sa od užívateľa nevyžaduje znalosť jednotlivých príkazov, s ktorými toolbox pracuje. Výsledky získané v GUI je možné jednoducho exportovať do pracovného priestoru MATLABu, rovnako ako je možné importovať potrebné údaje z neho do GUI.

Natrénovaná UNS predstavujúca dopredný model systému sa použila na identifikáciu dvoch typov systému a to najprv lineárneho systému 2. rádu a následne nelineárneho systému biprocesu. Z priebehov porovnania trénovacích resp. testovacích dát s odhadom zo siete pre lineárny systém (obr.2.2 -2.3) a nelineárny systém bioprocesu (obr.2.14 –2.16) je zrejmé, že dobre natrénované UNS dokážu dobre aproximovať ľubovolnú spojitú funkciu.

UNS použitá pri riadení jednotlivých systémov predstavovala inverzný model systému, ktorý sa použil ako spätnoväzbový nominálny neurónový regulátor.

Pri lineárnom systéme sa nominálny neurónový regulátor použil pri riadení nominálneho aj perturbovaného systému. Pri riadení nominálneho systému vykazoval nominálny regulátor dobré regulačné vlastnosti, v regulačnom obvode nezanechával TRO. Pri riadení perturbovaného systému sa najprv perturboval koeficient tlmenia do oblasti periodicity, pričom znižovaním koeficientu tlmenia sa postupne zvyšovali oscilácie systému. Nominálny regulátor vykazoval dobré regulačné vlastnosti po hodnotu koeficientu tlmenia ξ=0.6, pri nižších hodnotách sa systém stával nestabilným. Následne sa perturbovalo zosilnenie systému v rozsahu ±50% nominálnej hodnoty. Nominálny regulátor už nedokázal uriadiť perturbovaný systém na žiadanú hodnotu a zanechával v regulačnom obvode výraznú TRO. Tento nedostatok sa odstránil adaptáciou výstupu neurónovej siete cez prahový neurón. Takto získaný robustný regulátor po skončení adaptácie potom predstavuje presnú inverziu riadeného systému

a je schopný uriadiť aj perturbovaný systém bez zanechania TRO v regulačnom pochode.

Pri nelineárnom modeli bioprocesu bol inverzný model systému natrénovaný s väčšou chybou ako u lineárneho systému. Snahou bolo ukázať robustnosť získaného neurónového regulátora, preto sa daný natrénovaný inverzný model už ďalej nezlepšoval. Takto natrénovaný inverzný model bol použitý ako nominálny regulátor, ktorý uspokojivo uriadil nominálny systém, pri riadení perturbovaného systému už zanechával výraznú TRO. Nominálny neurónový regulátor sa preto adaptoval a získaný robustný regulátor sa následne testoval na riadení perturbovaného systému. Jedným zo spôsobov adaptácie neurónového regulátora je pridanie jednoduchého integračného člena. Nevýhodou takto vytvoreného robustného inverzného regulátora je jeho tendencia riadiť systém s preregulovaním aj v prípade, že riadi nominálny systém.

Ako vhodnejší typ regulátora na báze UNS sa ukázalo vytvorenie fuzzy-neurónového regulátora, kde sa inverzný neurónový regulátor rozšíril o fuzzy systém predstavujúci fuzzy I zložku. Fuzzy-neurónový regulátor si sám riadi vplyv integračného člena, čím sa zníži hodnota preregulovania ako u nominálneho tak aj u perturbovaného systému v porovnaní s robustným regulátorom.

LITERATÚRA

[1] KVASNIČKA, V.; BEŇUŠKOVÁ, Ľ.; POSPÍCHAL, J.; FARKAŠ, I.; TIŇO, P.; KRÁĽ, A.: Úvod do teórie neurónových sietí. Bratislava : Iris, 1997.

[2] SINČÁK, P.; ANDREJKOVÁ, G.: Neurónové siete – Inžiniersky prístup (1.diel). Košice : Elfa.s.r.o, 1996.

[3] MÉSZÁROS, A.; ANDRÁŠIK, A.: Robust and adaptive control in terms of neural networks. In *Proc. 13th International Conference on Process Controll*. Štrbské Pleso, 2001.

[4] MÉSZÁROS, A.; ANDRÁŠIK, A.; ŠPERKA, Ľ.: Influence of the adaptation gain on robust neural control. In *Proc. 5th International Scientific Technical Conference on Process control*. Kouty nad Desnou, 2002.

[5] HORNIK, K.; STINCHCOMBE, M.; WHITE, H.: Multilayer feedforward networks are universal approximators. *Neural Networks* ***2***, 1989. 259-366.

[6] HUNT, K. J.: Neural networks for control systems – A survey. *Automatica* ***28***, 1992. 1083-1111.

[7] BAKOŠOVÁ, M.; MÉSZÁROS, A.; ŠPERKA, Ľ.: Control of bioprocess using fuzzy-neural controller. In *Proc. of the 31th International Conference SSCHE.* Tatranské Matliare. 2004

[8] TSAPTSINOS, D.: Back-propagation and its variations. *Neural Networks for Chemical Engineers*. Elsevier Science B.V., 1995. 34-75.

[9] KOLESÁROVÁ, A.; KOVÁČOVÁ, M.: Fuzzy množiny a ich aplikácie. Bratislava: Vydavateľstvo STU, 2004.

[10] JURA, P: Základy fuzzy logiky pro řízení a modelování. Brno: VUTIUM, 2003.

[11] ŠPERKA, Ľ.: Identifikácia a riadenie systémov pomocou umelých neurónových sietí. *Diplomová práca*. FCHPT STU, Bratislava, 2001.

PRÍLOHY

Obsah priloženého CD

Súčasťou tejto diplomovej práce je CD, ktoré obsahuje napísanú diplomovú prácu vo formátoch DOC a PDF.