Univerzita Pardubice
Fakulta chemicko-technologická

Umělé neuronové sítě v modelování a řízení kontinuálního bioreaktoru

Petr Doležel

Diplomová práce
2008

Univerzita Pardubice
Fakulta chemicko-technologická
Katedra řízení procesů a výpočetní techniky
Akademický rok: 2007/2008

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Petr DOLEŽEL**

Studijní program: **M2802 Chemie a technická chemie**

Studijní obor: **Řízení technologických procesů**

Název tématu: **Umělé neuronové sítě v modelování a řízení bioreaktoru**

Zásady pro vypracování:

1. Teoretický úvod, literaturní rešerše
2. Statistický model bioreaktoru
3. Dynamický model bioreaktoru
4. Možnosti řízení bioreaktoru
5. Uživatelská příručka pro tvorbu UNS v MATLAB/SIMULINK

Rozsah grafických prací:
Rozsah pracovní zprávy:
Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

Dokumentace k bioreaktoru
Dokumentace k MATLAB/SIMULINK

Vedoucí diplomové práce: **prof. Ing. Ivan Taufer, DrSc.**
Ústav elektrotechniky a informatiky

Datum zadání diplomové práce: **26. února 2008**
Termín odevzdání diplomové práce: **9. května 2008**

prof. Ing. Petr Lošťák, DrSc.
děkan

L.S.

doc. Ing. František Dušek,
vedoucí katedry

V Pardubicích dne 28. února 2008

Děkuji prof. Ing. Ivanu Tauferovi, DrSc. za odborné vedení v průběhu tvorby diplomové práce. Rovněž patří velký dík mé rodině za nepřetržitou podporu po celou dobu studia.

Abstrakt

Problematika umělých neuronových sítí je relativně nová vědní disciplína, která se v posledních letech uplatňuje v širokém spektru oborů. Tato práce zkoumá možnosti jejího uplatnění v modelování a řízení technologických procesů a své závěry demonstruje na konkrétním příkladu identifikace a regulace kontinuálního bioreaktoru.

V úvodu práce jsou shrnuty teoretické základy zkoumaného vědního oboru a jeho historie, jsou zde popsány principy, které budou využívány v dalších částech. V následujícím oddíle je popsán matematicko-fyzikální model bioreaktoru sloužící jednak pro simulaci sběru dat a také pro pozdější porovnání s modely neuronovými. Dále jsou již aplikovány poznatky o neuronových sítích při tvorbě neuronových statických a dynamických modelů bioreaktoru, přičemž jsou současně zhodnoceny použité algoritmy učení neuronových sítí. V poslední části práce jsou navrženy a zhodnoceny dvě možnosti využití neuronových sítí při řízení bioreaktoru.

Klíčová slova: Umělé neuronové sítě, bioreaktor, modelování, řízení

Abstract

Artificial neural networks are included in relatively new branch of science, which has been applied to many different places, recently. This Thesis examines possibilities of their use in process modelling and control. The conclusions are demonstrated on concrete example of continual bioreactor identification and control.

The Thesis is opened by recapitulation of neural networks principles and their history. There are described the basics here which are used in other parts of the Thesis. In the second section, there is defined analytical model of continual bioreactor. It serves for data acquisition as well as for further compare with neural models. Further, neural networks basics are applied for creations of static and dynamic neural models. In parallel, there are rated different learn algorithms. In the last section, there are proposed and rated two possibilities of bioreactor control with neural controllers.

Keywords: Artificial neural networks, bioreactor, modelling, control

Obsah

Seznam obrázků

Seznam tabulek

Seznam symbolů

Skalární hodnoty jsou značeny kurzívou, matice a vektory navíc tučně.
Pokud symbol vyjadřuje veličinu a ta není bezrozměrná, v popisku je uvedena její jednotka.

a	… fyziologický koeficient
b	… posunutí funkce
e	… chyba na výstupu z neuronu
e_χ	… rozdíl skutečného a modelovaného výstupu bioreaktoru $\chi - \chi_M$
e_σ	… rozdíl skutečného a modelovaného výstupu bioreaktoru $\sigma - \sigma_M$
e_ω	… rozdíl skutečného a modelovaného výstupu bioreaktoru $\omega - \omega_M$
f	… obecné označení funkce
$\boldsymbol{g}$	… vektor gradientů změny vah spojení
t	… časová proměnná
$\boldsymbol{u}_R$	… obecný vektor vstupů do soustavy
u	… vstupní potenciál neuronu
v	… obecné označení poruchy
w_S	… žádaná hodnota
$\boldsymbol{w}$	… matice vah spojení
$\boldsymbol{x}$	… vektor vstupů do neuronu, $\boldsymbol{x} = [x_0, x_1, x_2, \cdots]^{\mathrm{T}}$
y_j	… výstup z neuronu j
$\boldsymbol{y}_S$	… obecný vektor výstupů ze soustavy
$\boldsymbol{y}_M$	… obecný vektor výstupů z neuronové sítě (neuronového modelu)
z^{-i}	… operátor posunutí o i period intervalu vzorkování
C	… koncentrace kyslíku v objemu kultivační tekutiny, $g \cdot l^{-1}$
C^R	… rovnovážná koncentrace kyslíku v objemu kultivační tekutiny, $g \cdot l^{-1}$
D	… rychlost zředění (reciproká hodnota k době přítomnosti kultivační tekutiny v reaktoru), $l \cdot h^{-1}$
E	… celková chybová energie pro všechny neurony ve výstupní vrstvě (kriteriální funkce)
$E(N)$	… hodnota kriteriální funkce po ukončení trénování
E_{AV}	… průměrná chybová energie za jednu epochu trénování
F_{IM}	… označení přenosu inverzního modelu soustavy
F_R	… označení přenosu regulátoru
F_{RI}	… označení přenosu regulátoru s vnitřním modelem
F_S	… označení přenosu soustavy
$\boldsymbol{H}$	… Hessova matice druhých derivací
$\boldsymbol{I}$	… Jednotková matice
$\boldsymbol{J}$	… Jacobiho matice
K	… objemový koeficient
K_C	… konstanta nasycení (konstanta Monoda) kyslíkem, $g \cdot l^{-1}$
$K_L a$	… objemový koeficient přestupu hmoty pro kyslík, $l \cdot h^{-1}$
K_S	… konstanta nasycení (konstanta Monoda) substrátem, $g \cdot l^{-1}$

Kr … kriterium kvality regulace

S … koncentrace substrátu v kultivační tekutině, $g \cdot l^{-1}$

S_0 … koncentrace substrátu v toku živné látky, $g \cdot l^{-1}$

T … znaménko transpozice

T_S … interval vzorkování

X … koncentrace biomasy v objemu kultivační tekutiny, $g \cdot l^{-1}$

Y_C … ekonomický koeficient udávající hmotnost biomasy na výstupu v přepočtu na jednotku hmoty spotřebovaného kyslíku

Y_S … ekonomický koeficient udávající hmotnost biomasy na výstupu v přepočtu na jednotku hmoty spotřebovaného substrátu

α … koeficient rychlosti učení

δ … průtok suroviny bioreaktorem

δ_V … průtok suroviny bioreaktorem odpovídající rychlosti vymývání

δ_j … lokální gradient neuronu j

ε_j … chybová energie neuronu j

φ … obecné označení funkce

φ^{-1} … obecné označení inverzní funkce

μ_m … maximální rychlost růstu mikroorganismů, $l \cdot h^{-1}$

χ … koncentrace biomasy na výstupu z bioreaktoru

χ_M … koncentrace biomasy na výstupu z neuronového modelu bioreaktoru

σ … koncentrace substrátu na výstupu z bioreaktoru

σ^0 … počáteční koncentrace substrátu (na vstupu do bioreaktoru)

σ_M … koncentrace substrátu na výstupu z neuronového modelu bioreaktoru

ω … koncentrace kyslíku na výstupu z bioreaktoru

ω^0 … rovnovážná koncentrace kyslíku

ω_M … koncentrace kyslíku na výstupu z neuronového modelu bioreaktoru

Θ … strmost funkce

1 Úvod

Je známo, že biologické systémy mohou řešit složité problémy různého druhu pomocí mnoha funkcí svých nervových soustav, přičemž mezi jednu z nejpozoruhodnějších a zároveň jednu z nejoceňovanějších se řadí schopnost učit se. Organismy se učí pomocí soustavy biologických neuronů spojených mezi sebou dendrity a axony. Umělá neuronová síť je jakýsi zjednodušený matematický model biologické neuronové sítě. Podobně jako nervový systém, je schopna i umělá neuronová síť zpracovat široké množství vstupních dat za dodržení požadovaných výstupů, ovšem tuto schopnost se nejprve musí naučit (natrénovat). Správně natrénované umělé neuronové sítě jsou pak užitečné a nezřídka se používají v mnoha oborech lidského bádání.

Neuronové sítě se v historii svého vývoje vyskytly na obou pólech zájmu veřejnosti. Vlny jejich rozvoje i útlumu závisely především na technologických možnostech a rozvoji výpočetní techniky. Využití umělých neuronových sítí je široké, je však nutno k nim přistupovat realisticky a vědět, že řadu problémů je vhodnější řešit jinými metodami.

2 Cíl práce

Na konkrétním technologickém procesu – provozu kontinuálního bioreaktoru – je v této práci snahou ukázat různé možnosti aplikace umělých neuronových sítí.

Po shrnutí a teoretických základech problematiky umělých neuronových sítí uvedených úvodní části je třeba nejprve sestavit matematicko-fyzikální statický i dynamický model kontinuálního bioreaktoru, který bude sloužit k simulaci reálného zařízení. Na tomto modelu pak budou aplikovány veškeré experimenty týkající se neuronových sítí. Hlavními cíly v této oblasti je sestavit statický i dynamický neuronový model bioreaktoru a porovnat s matematicko-fyzikálním modelem a následně využít možnosti neuronových sítí při řízení bioreaktoru.

Ačkoliv je problematika neuronových sítí poměrně nová oblast vědeckého zaměření, jejíž rozvoj mohl nastat až souběžně s rozvojem výkonné výpočetní techniky, dodnes bylo navrženo mnoho různých typů architektur umělých neuronových sítí a k nim příslušných algoritmů učení. V této práci se budou používat výhradně vícevrstvé dopředné neuronové sítě. K dopředným sítím však byla navržena řada algoritmů učení a bylo tedy třeba aspoň nejznámější z nich určitým způsobem zhodnotit, o což se tato práce také pokusila.

Věřím, že po přečtení vyplynou jasné výhody umělých neuronových sítí v oblasti modelování a že výsledky řízení nelineární soustavy pomocí různých typů zapojení s neuronovými regulátory aspoň přinutí zamyslet se nad tímto alternativním typem regulace.

3 Teoretický popis umělých neuronových sítí

3.1 Historický vývoj

Počátek moderní éry neuronových sítí je datován do roku 1943, kdy psychiatr a neuroanatom Warren S. McCulloch společně s matematikem Waltrem Pittsem publikovali dnes již klasický článek, ve kterém ukázali, že správně navržená neuronová síť teoreticky může vypočíst jakoukoliv spočitatelnou funkci. Tímto článkem vznikla nová vědní disciplína zabývající se umělou inteligencí a umělými neuronovými sítěmi.

Velký krok ve vývoji umělých neuronových sítí byl učiněn v roce 1949 po vydání knihy Donalda Hebba *The Organization of Behavior*, ve které autor popisuje pravidla fyziologického učení. Kniha byla zdrojem inspirace pro výpočetní modely učení a adaptivních systémů. Na základě těchto informací odvodil v roce 1958 Frank Rosenblatt algoritmus učení vrstevnaté neuronové sítě s dopředným šířením signálu. Tuto síť nazval PERCEPTRON.

V roce 1959 pánové Bernard Widrow a M. E. Hoff odvodili a popsali neuronovou síť tvořenou jedním neuronem s jedním výstupem, několika vstupy a doplňkovým signálem, kterou nazvali ADALINE, posléze MADALINE pro více neuronů v síti.

V sedmdesátých letech 20. století však nastal útlum ve vývoji umělých neuronových sítí, který byl způsoben několika příčinami:

- V sedmdesátých letech nebyla dostatečně výkonná výpočetní technika na to, aby umožnila dostatek experimentů s umělými neuronovými sítěmi.
- V roce 1969 vydali matematikové Minski a Papert vědeckou práci, ve které popsali meze tehdy známých architektur umělých neuronových sítí. Dospěli k závěru, že umělé neuronové sítě nemohou nahradit klasické metody například kvůli tomu, že s jejich pomocí nelze simulovat všechny druhy základních logických funkcí. (Jednovrstvým perceptronem nelze nahradit XOR funkci).
  Tato kritika částečně způsobila útlum přítoku financí na vývoj umělých neuronových sítí.

Ovšem v roce 1983 došlo ve Spojených státech amerických k obrovským investicím do vývoje umělých neuronových sítí a po třech letech výzkumu byl vědci Rumelhartem a LeCunem z organizace DARPA odvozen nový algoritmus učení vrstevnatých neuronových sítí – algoritmus zpětného šíření chyby. Aplikace tohoto algoritmu zcela vyvrátila kritiku Minskeho a Paperta a způsobila novou vlnu zájmu o umělé neuronové sítě.

Po tomto impulsu se objevilo velké množství vědeckých prací, které posunuly vývoj kupředu.

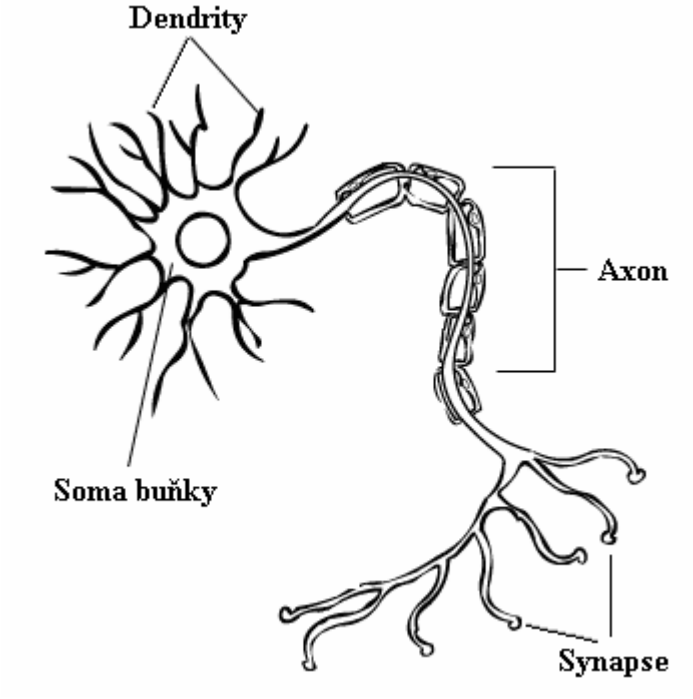
3.2 Analogie biologických a umělých neuronových sítí

Umělé neuronové sítě si ve své podstatě kladou za vzor neuronové sítě biologické. Podstatou návrhu umělé neuronové sítě je tedy model struktury a činnosti biologické neuronové sítě. Činnost nervové soustavy živočichů je velmi komplikovaná a vykonává obrovské množství spletitých úkolů. Základní stavební jednotkou nervové soustavy je však neuron (obr. 3.1), jehož činnost jako jednotlivce vzhledem k celku je až překvapivě jednoduchá.

Neuron, jako většina buněk v lidském těle, je složen z obvyklých částí (jádro, cytoplazma, plazmatická membrána, ...), ovšem z těla neuronu navíc vystupují výběžky dvojího druhu – dendrity a axony. Dendritů bývá větší počet a vedou informaci do neuronu, zatímco axon

bývá pro každý neuron pouze jediný a vede informaci ven z neuronu. Konec axonu se větví a každá větev je opatřena synapsí. Synapse je složitý útvar, který zajišťuje přenos informací do okolních neuronů a podílí se na procesu učení.

Spojením neuronů v organismech vzniká neuronová síť, často velmi složitý útvar čítající miliardy neuronů.



Obr. 3.1 – Biologický neuron

Činnost neuronu lze popsat následujícím způsobem. K aktivaci neuronu dochází v okamžiku, kdy souhrn vstupů do neuronu (podnětů) překročí určitou prahovou hodnotu. V tom případě neuron souhrn vstupů ve svém těle zpracuje a do svého axonu vyšle výstupní signál jako reakci na vstupy, čímž se informace nese sítí dále. V opačném případě (prahová hodnota nebyla vstupy dosažena) je na výstupu z neuronu signál odpovídající pasivnímu stavu. Je potřeba zdůraznit, že biologická neuronová síť se neustále učí (probíhá proces adaptace), tedy reakce neuronu na stejné vstupy může být v rozdílných časových okamžicích jiná.

Proces učení biologické neuronové sítě lze velmi zjednodušeně popsat takto: vjem, který je třeba si zapamatovat, je receptorem transformován na elektrický signál, který se stává nositelem informace o sledovaném vjemu. Signál je veden po drahách, které propojují jednotlivé neurony v sítí a každý průchod signálu po dráze zanechá paměťovou stopu v délce trvání několika vteřin. Při opakovaném průchodu stejného signálu danou drahou dochází k modifikaci synapsí tak, že se dráha stává pro signál propustnější a trvání paměťové stopy delší. Naopak nepoužívání dráhy pro daný signál vede k oslabení propustnosti daného signálu.

Topologie umělé neuronové sítě má obdobnou strukturu, pro srovnání jsou dále uvedeny základní části biologického a umělého neuronu.

Složení biologického neuronu:

- Tělo neuronu (soma) – eukariotická buňka, obsahuje buněčné jádro
- Dendrit – nervový výběžek, který vede vzruch směrem k buňce, do každého neuronu vstupuje několik dendritů
- Axon – nervový výběžek, který vede vzruch směrem z buňky, z každého neuronu vystupuje pouze jeden axon
- Synapse – axony jsou s okolím spojeny přes synapse, které tvoří jakési komunikační rozhraní vyznačující se velkou plasticitou (často používané synapse se rozšiřují, nepoužívané synapse zanikají)
  – proměnná průchodnost synapsí souvisí s principem učení

Analogicky popsané složení umělého neuronu:

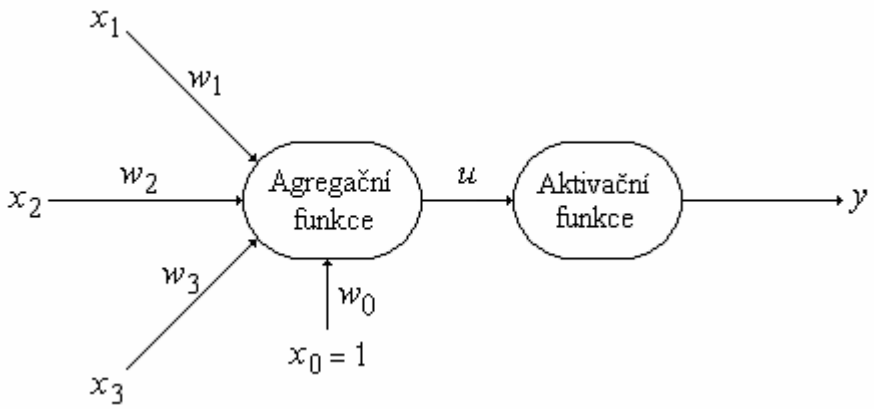
- Tělo neuronu – matematický procesor
- Vstup – n-rozměrný vektor hodnot
- Výstup – skalární hodnota, pouze jedna
- Vektor vah spojení – vyjadřuje uložení zkušeností do neuronu, neuron je pomocí tohoto vektoru schopen adaptovat se na nově získané zkušenosti během učení

Další text se bude týkat již pouze umělých neuronů, proto výrazem *neuron* bude dále označen *umělý neuron*.

3.3 Popis neuronu

3.3.1 Formální neuron

V umělém neuronu dochází oproti originálu k náhradě biologických funkcí funkcemi matematickými. Je možno se setkat s velkým počtem různých modelů, nejčastěji je však používán tzv. *formální neuron*, nazývaný někdy též podle svých autorů *McCullochův-Pittsův model neuronu*. Jeho schéma je uvedeno na obrázku 3.2.



Obr. 3.2 – McCullochův-Pittsův model neuronu

Jak je patrno z obrázku 3.2, hodnoty vstupů x_1, x_2, ... x_n do neuronu jsou transformovány pomocí agregační funkce na jedinou skalární hodnotu vstupního potenciálu *u* a tato hodnota je pak pomocí aktivační funkce převedena na konečnou hodnotu výstupu *y* z neuronu.

Prahem neuronu je nazýván první vstup do neuronu neboli součin $w_0 \cdot x_0$. Je nutno poznamenat, že neuronová síť je obecně dynamický systém, proto může být stav neuronu v čase *k* rozdílný od stavu neuronu v čase *k*+1 a podobně.

3.3.2 Agregační funkce neuronu

Agregační funkce určuje, jakým způsobem jsou vstupní parametry kombinovány uvnitř neuronu.

Nejčastěji používané agregační funkce jsou následující:

- Lineární bazické funkce
  Mezi lineární bazické funkce patří agregační funkce popsané vztahem (3-1), resp. (3-2):

$$u = \sum_i w_i x_i \tag{3-1}$$

$$u = \prod_i w_i x_i \tag{3-2}$$

- Radiální bazické funkce
  Tato skupina agregačních funkcí je zastoupena vztahem (3-3):

$$u = \sqrt{\sum_i (x_i - w_i)^2} \tag{3-3}$$

3.3.3 Aktivační funkce neuronu

Aktivační funkce určují vztah mezi hodnotou agregační funkce a výstupem neuronu. Obecně lze tento vztah vyjádřit rovnicí (3-4):

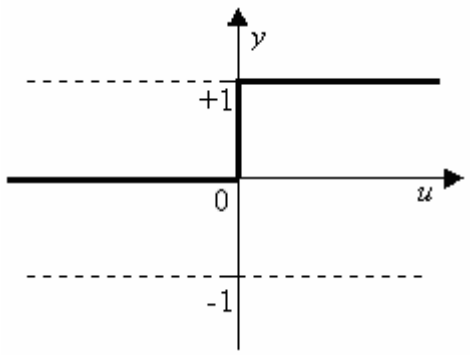
$$y = f(u) \tag{3-4}$$

Nejčastěji používané aktivační funkce jsou uvedeny v následujícím výčtu.

- Skoková aktivační funkce

$$f(u) = \begin{cases} 1 & \text{pro } u \geq 0 \\ 0 & \text{pro } u < 0 \end{cases} \tag{3-5}$$

  Průběh je znázorněn na obrázku 3.3.

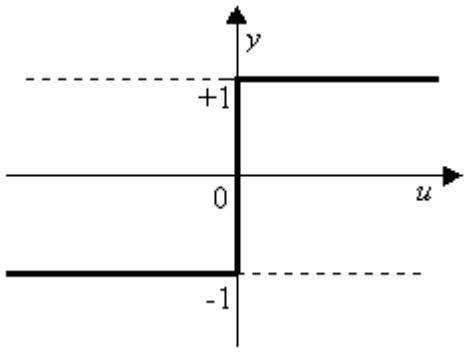


Obr. 3.3 – Skoková aktivační funkce

- Symetrická skoková aktivační funkce

$$f(u) = \begin{cases} 1 & \text{pro } u \geq 0 \\ -1 & \text{pro } u < 0 \end{cases} \tag{3-6}$$

Průběh je znázorněn na obrázku 3.4.

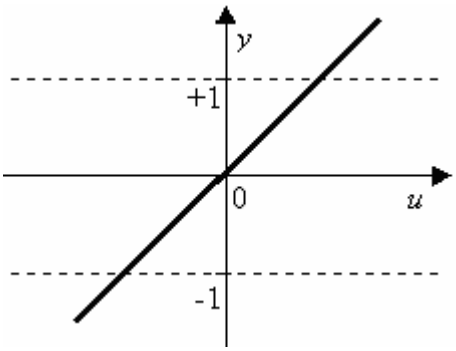


Obr. 3.4 - Symetrická skoková aktivační funkce

- Lineární aktivační funkce

$$f(u) = \Theta \cdot u + b \tag{3-7}$$

Průběh pro $b = 0$ je znázorněn na obrázku 3.5.

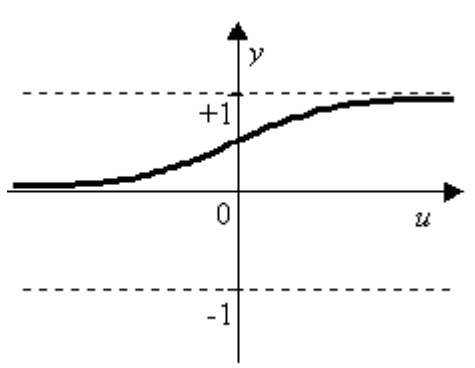


Obr. 3.5 - Lineární aktivační funkce

- Sigmoidální aktivační funkce

$$f(u) = \frac{1}{1 + e^{\Theta \cdot u}} \tag{3-8}$$

Průběh je znázorněn na obrázku 3.6.

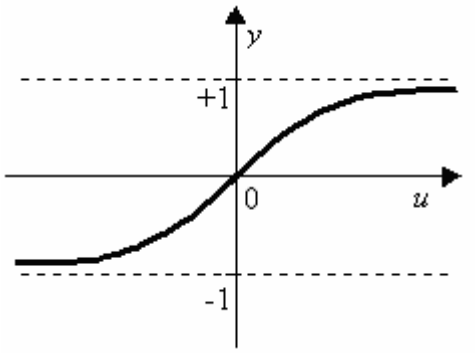


Obr. 3.6 - Sigmoidální aktivační funkce

- Hyperbolicko-tangenciální aktivační funkce

$$f(u) = \tanh\left(\Theta \cdot u\right) \tag{3-9}$$

Průběh je znázorněn na obrázku 3.7:



Obr. 3.7 – Hyperbolicko-tangenciální aktivační funkce

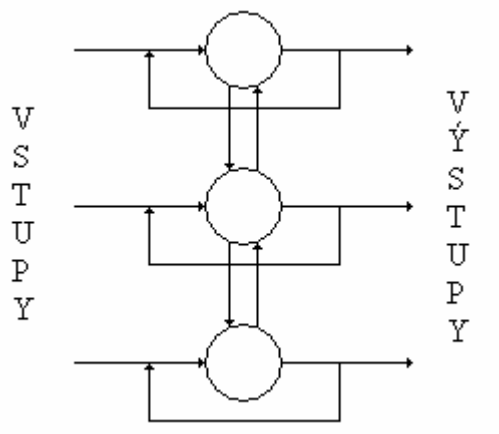
Další typy aktivačních funkcí je možno nalézt v literatuře.

3.4 Architektury neuronových sítí

Architekturou neuronové sítě se rozumí uspořádání neuronů a jejich vzájemné propojení.

Z hlediska počtu vrstev se architektura neuronových sítí dělí na:

- Jednovrstvé sítě:
  Jednovrstvé sítě neobsahují skryté vrstvy.
  Příkladem může být Hopfieldova síť znázorněná na obrázku 3.8.

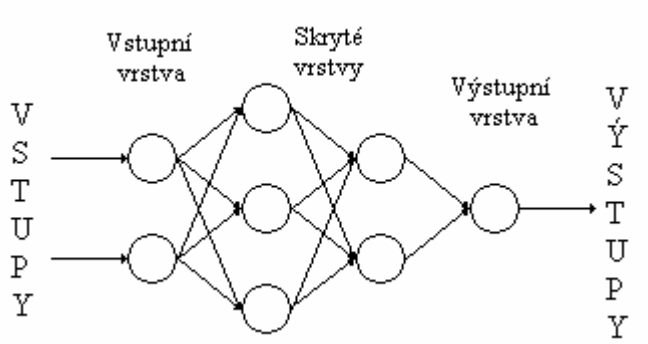


Obr. 3.8 – Hopfieldova síť

- Vícevrstvé (vrstevnaté) sítě:
  Každá vrstevnatá síť je složena z tří typů vrstev:

  - Jedna vstupní vrstva
  - Obecně více skrytých vrstev
  - Jedna výstupní vrstva

  Příklad vzhledu vrstevnaté neuronové sítě je na obrázku 3.9.



Obr. 3.9 – Vrstevnatá neuronová síť

Zápis 2 – 3 – 2 – 1 pak znamená, že neuronová síť má dva vstupní neurony, tři neurony v první skryté vrstvě, dva neurony v druhé skryté vrstvě a jeden výstupní neuron.

Z hlediska průchodu informací neuronovou sítí se rozlišují dva základní typy:

- Dopředné sítě:
  Tok informací prochází přímo od vstupu k výstupu sítě. Tato architektura byla použita u neuronové sítě na obrázku 3.9.

- Rekurentní sítě:
  Kromě informací v přímém směru je díky zpětným vazbám v rámci architektury sítě umožněn i tok informací ve zpětné vazbě z jakéhokoliv výstupního bodu zpět na vstup sítě či určité vrstvy. Příkladem může být síť na obrázku 3.8.

Z hlediska hodnot parametrů v neuronových sítích se rozlišují následující dva typy:

- Statické sítě:
  Parametry sítě jsou nastaveny v procesu trénování, dále při využívání neuronové sítě k řešení daného problému jsou tyto parametry konstantní.

- Dynamické sítě:
  Parametry sítě jsou přednastaveny procesem trénování a dále jsou určitými metodami adaptovány přímo při jejím využívání k řešení daného problému.

3.5 Principy učení umělých neuronových sítí

3.5.1 Úvod

Pojmy jako učení a paměť jsou spjaty s možností vývoje nervových buněk a s jejich propojením. Tato vlastnost se nazývá plasticita synapsí a je způsobena eliminací nebo posilováním synapsí. Stejný princip je používán i při učení (trénování) umělých neuronových sítí.

Proces učení je možno definovat jako modifikaci vah spojení a prahů neuronů příslušné neuronové sítě tak, aby odchylka mezi skutečným a požadovaným výstupem z neuronové sítě byla minimální. Tréninkovou množinou dat se rozumí takový výběr dat ze základního prostoru vstupů do neuronové sítě, který tento základní prostor dobře reprezentuje. Tréninková množina dat se pak použije k učícímu procesu neuronové sítě.

Principy učení neuronových sítí je opět možno rozdělit podle celé řady hledisek, v této části bude zmíněno pouze rozdělení podle toho, jestli jsou známy požadované výsledky výstupu či nikoliv.

- Učení s učitelem:
  Při tomto typu učení jsou známy požadované výsledky odpovídající tréninkové množině dat. Tyto výsledky se během učení porovnávají s výstupem ze sítě. Síť během učení nastavuje své parametry (prahy a váhy spojení) tak, aby se výstupy blížily požadovaným hodnotám.

- Učení bez učitele (samoorganizace):
  V tomto případě není množina vzorů (uspořádané dvojice [vstup; požadovaný výstup]) k dispozici. Využívá se schopnosti neuronových sítí rozeznat ve vstupech blízké vlastnosti a tak třídit vstupní data do shluků.

V další části bude nastíněn průřez vývojem učících algoritmů po algoritmus zpětného šíření chyby a jeho modifikace.

3.5.2 Hebbův zákon učení

Na tento zákon je třeba se dívat jako na základ všech současných algoritmů učení. Donald Hebb byl neuropsycholog, který své závěry činil pro biologickou neuronovou síť, ovšem stejně tak se tyto závěry dají použít pro umělé neuronové sítě. Jeho postuláty byly parafrázovány do dvou hlavních pravidel:

- Pokud jsou dva spolu propojené neurony aktivovány současně, vazba mezi nimi se posílí.
- Pokud jsou dva spolu propojené neurony aktivovány asynchronně (v rozdílných časových okamžicích), vazba mezi nimi je oslabena či zaniká.

Každý neuron má pouze dva stavy, aktivní (1) a neaktivní (0). Hebbův zákon učení lze zapsat vztahem (3-10).

$$w_{ji}(k+1) = w_{ji}(k) + \alpha \cdot y_i(k) \cdot x_j(k) \tag{3-10}$$

Tedy váha spojení neuronů i a j w_{ji} v čase $k + 1$ je rovna hodnotě w_{ji} v čase k zvýšené o součin presynaptického stavu neuronu j x_j a postsynaptického stavu neuronu i y_i násobený konstantou rychlosti učení α.

3.5.3 Chybová učení

3.5.3.1 Úvod

Chybová učení jsou učení s učitelem, během kterých se nastavují hodnoty vah spojení úměrně mezi požadovanými a vypočtenými hodnotami. Obecně pro chybová učení platí dvojice vztahů (3-11) a (3-12).

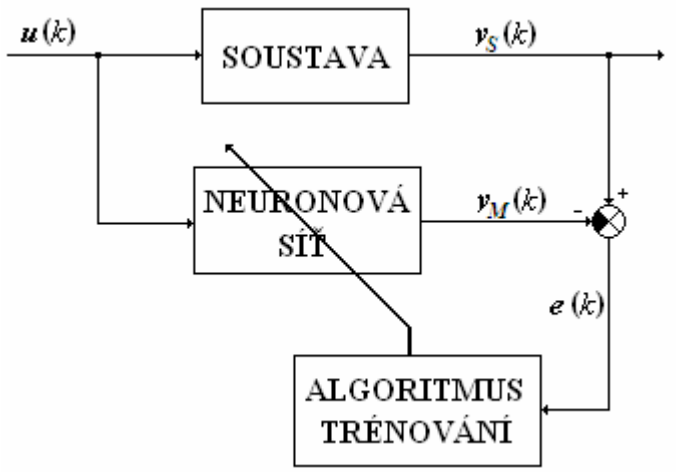
$$w_{ji}(k+1) = w_{ji}(k) + \Delta w_{ji}(k) \tag{3-11}$$

$$\Delta w_{ji}(k) = \alpha \cdot x_i(k).\left[y_{Sj}(k) - y_{Mj}(k) \right] \tag{3-12}$$

Tedy oprava váhy spojení mezi neuronem i a neuronem j je rovna součinu hodnoty signálu z neuronu i do neuronu j x_i a rozdílu požadované hodnoty výstupního signálu z neuronu j y_{Sj} a skutečného výstupu z neuronu j y_{Mj} násobenému ještě koeficientem rychlosti učení α.

V takovéto podobě ovšem nelze chybové učení aplikovat na vrstevnaté sítě, neboť v těchto případech není možno přímo měřit chyby na výstupu z neuronů ve skrytých vrstvách. Byly však vytvořeny modifikace, které budou zmíněny v následujících odstavcích.

Obecně lze učení neuronové sítě znázornit blokově podle obrázku 3.10.



Obr. 3.10 – Blokové schéma učení (trénování) neuronové sítě

3.5.3.2 LMS algoritmus (delta pravidlo)

LMS algoritmus (*Least Mean Square Algorithm* – algoritmus střední kvadratické chyby) slouží k učení sítí bez skrytých vrstev, přičemž neurony těchto sítí mají agregační funkci definovanou vztahem (3-1) a aktivační funkce je lineární (jak píše Haykin, [1999] či Tučková, [2005], naproti tomu Nguyen, [2003] připouští obecnou diferencovatelnou aktivační funkci, v dalším textu se však bude uvažovat lineární aktivační funkce tvaru (3-13)).

$$y = u \tag{3-13}$$

Snahou je pro každý neuron minimalizovat účelovou funkci definovanou vztahem (3-14).

$$E(k) = \frac{1}{2} e^2(k) \tag{3-14}$$

kde: $e(k) = y_S(k) - y_M(k)$ ... chyba na výstupu v okamžiku k

$$y(k) = \boldsymbol{x}(k)^{\mathrm{T}} \cdot \boldsymbol{w}(k) \tag{3-15}$$

Tedy účelová funkce je dána druhou mocninou chyby na výstupu z neuronu, přičemž skutečná hodnota výstupu z neuronu se díky platnosti vztahu (3-13) vypočte jednoduše jako skalární součin vektoru vstupů do neuronu $\boldsymbol{x}(k)$ a vektoru vah $\boldsymbol{w}(k)$.

Nezávislý parametr, pomocí kterého lze hodnotu účelové funkce $E(k)$ optimalizovat, je v tomto případě vektor vah spojení $\boldsymbol{w}(k)$. Pro nalezení extrému funkce (3-14) je tedy třeba ji derivovat podle $\boldsymbol{w}$.

$$\left.\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = \left.\frac{\partial \frac{1}{2} e^2(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = e(k) \cdot \left.\frac{\partial e(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} \tag{3-16}$$

A s ohledem na (3-15) je možno psát vztah (3-17).

$$\left.\frac{\partial e(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = \left.\frac{\partial\left[y_M(k) - \boldsymbol{x}(k)^{\mathsf{T}} \cdot \boldsymbol{w}\right]}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = -\boldsymbol{x}(k)^{\mathsf{T}} \tag{3-17}$$

Kombinací vztahů (3-16) a (3-17) je pak možno obdržet vztah (3-18).

$$\left.\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = -\boldsymbol{x}(k)^{\mathsf{T}} \cdot e(k) = \boldsymbol{g}(k)^{\mathsf{T}} \tag{3-18}$$

Vztah (3-18) se v praxi používá jako odhad vektoru gradientu závislosti $\boldsymbol{w} = \boldsymbol{w}(k)$, jak ukazuje vztah (3-19).

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) - \alpha \cdot \boldsymbol{g}(k) = \boldsymbol{w}(k) + \Delta \boldsymbol{w}(k) \tag{3-19}$$

Oprava vektoru vah v čase $k + 1$ oproti vektoru vah v čase k je tedy dána odhadem vektoru gradientu $\boldsymbol{g}(k)$ definovaným vztahem (3-18) násobeným zápornou hodnotou koeficientu rychlosti učení α.

Celkový postup *LMS algoritmu* lze shrnout do následujících bodů

1) Volba vhodné tréninkové množiny
2) Inicializace parametrů sítě
3) Volba koeficientu rychlosti učení
4) Vlastní proces učení podle vztahu (3-19)
5) Test podmínek pro ukončení učení – např. změna vah v poslední iteraci je nižší než zvolená hodnota…
6) Diskuse průběhu účelové funkce v závislosti na počtu iterací, v případě nepříznivého závěru návrat k bodu 2)

3.5.3.3 BPG algoritmus

BPG algoritmus (*Backpropagation Gradient Descent Algorithm* – algoritmus se zpětným šířením chyby) byl vyvinut pro neuronové sítě obsahující aspoň jednu skrytou vrstvu a řeší problém nemožnosti přímého měření chyby na výstupu neuronu nacházejícího se ve skryté vrstvě. Je to vlastně zobecněný *LMS algoritmus*.

V principu se *BPG algoritmus* aplikuje takovým způsobem, že na vstup neuronové sítě se přivede matice vstupních parametrů, po průchodu sítí je výstup porovnán s požadovanou hodnotou, je spočítána hodnota chybové funkce, která se zpětně přepočítá do předchozích vrstev a váhy spojení jsou opraveny. Provádí se další a další iterace a hledá se minimum chyby mezi skutečnou a požadovanou hodnotou.

Nevýhodou této metody je velká citlivost na správnost tréninkových dat a na inicializaci parametrů sítě.

Genezi k výpočtovým vztahům algoritmu zpětného šíření chyby lze popsat například následujícím způsobem.

Pro chybu na výstupu z neuronu *j*, který je ve výstupní vrstvě, platí v iteraci *k* vztah (3-20).

$$e_j(k) = y_{S_j}(k) - y_j(k) \tag{3-20}$$

Nechť chybová energie neuronu *j* v iteraci *k* je definována vztahem (3-21).

$$\varepsilon_j(k) = \frac{1}{2} e_j^{\,2}(k) \tag{3-21}$$

Celková chybová energie je pak obdržena sumací chybových energií všech neuronů ve výstupní vrstvě – vztah (3-22).

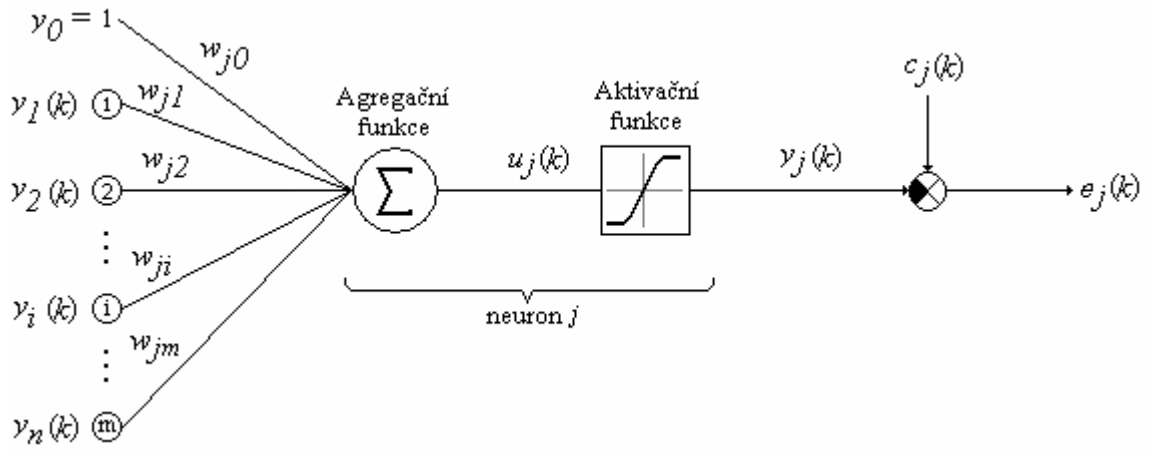
$$E(k) = \frac{1}{2} \sum_j e_j^{\,2}(k) \tag{3-22}$$

Pokud se uvažuje *N* uspořádaných dvojic [vstup, výstup] v tréninkové množině dat, pak průměrná chybová energie za jednu epochu trénování bude definována vztahem (3-23).

$$E_{AV} = \frac{1}{N} \sum_{k=1}^{N} E(k) \tag{3-23}$$

Průměrná energie je zřejmě funkcí všech parametrů sítě (vah spojení a prahů), bude tedy považována za účelovou funkci hledání optimálního nastavení všech parametrů.

Libovolný neuron *j* ve výstupní vrstvě je znázorněn na obrázku 3.11.



Obr. 3.11 – Neuron výstupní vrstvy

Z obrázku 3.11 je zřejmé, že platí vztahy (3-24) a (3-25).

$$u_j(k) = \sum_{i=0}^{m} w_{ji}(k) \cdot y_i(k) \tag{3-24}$$

$$y_j(k) = f\left[u_j(k)\right] \tag{3-25}$$

Podobně jako *LMS algoritmus*, *BPG algoritmus* využívá korekci Δw_{ji} váhy spojení w_{ji}, která je úměrná parciální derivaci $\frac{\partial E(w_{ji})}{\partial w_{ji}}$. Tuto složenou derivaci lze rozepsat jako součin dílčích derivací – vztah (3-26).

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial u_j} \cdot \frac{\partial u_j}{\partial w_{ji}} \tag{3-26}$$

Výraz $\frac{\partial E(w_{ji})}{\partial w_{ji}}$ představuje jakýsi faktor citlivosti, který určuje, jakým směrem se budou ubírat změny váhy spojení w_{ji}.

Nyní je třeba vyjádřit jednotlivé členy na pravé straně rovnice (3-26). Derivací obou stran rovnice (3-22) podle e_j se získá vztah (3-27).

$$\left.\frac{\partial E}{\partial e_j}\right|_{e_j = e_j(k)} = e_j(k) \tag{3-27}$$

Derivací obou stran rovnice (3-20) podle y_j se získá vztah (3-28).

$$\left.\frac{\partial e_j}{\partial y_j}\right|_{y_j = y_j(k)} = -1 \tag{3-28}$$

Dále derivací obou stran rovnice (3-25) podle u_j se získá vztah (3-29).

$$\left.\frac{\partial y_j}{\partial u_j}\right|_{u_j = u_j(k)} = f'\left[u_j(k)\right] \tag{3-29}$$

Přičemž derivační znaménko na pravé straně rovnice signalizuje derivaci podle argumentu.

Nakonec derivací obou stran rovnice (3-24) podle w_{ji} se získá vztah (3-30).

$$\left.\frac{\partial u_j}{\partial w_{ji}}\right|_{w_{ji} = w_{ji}(k)} = y_i(k) \tag{3-30}$$

Dosazením vztahů (3-27) až (3-30) do rovnice (3-26) se obdrží rovnice (3-31).

$$\left.\frac{\partial E}{\partial w_{ji}}\right|_{w_{ji} = w_{ji}(k)} = -e_j(k) \cdot f'\left[u_j(k)\right] \cdot y_i(k) \tag{3-31}$$

Analogicky k *LMS algoritmu* je definována modifikace váhy spojení w_{ji} – vztah (3-32).

$$\Delta w_{ji}(k) = -\alpha \cdot \left. \frac{\partial E(w_{ji})}{\partial w_{ji}} \right|_{w_{ji}=w_{ji}(k)} \tag{3-32}$$

Ovšem při použití *BPG algoritmu* se pro modifikaci váhy w_{ji} používá vztah (3-33).

$$\Delta w_{ji}(k) = \alpha \cdot \delta_j(k) \cdot y_i(k) \tag{3-33}$$

Tedy oprava váhy spojení mezi neurony *i* a *j* je rovna součinu lokálního gradientu neuronu *j* $\delta_j(k)$ a výstupu z neuronu *i* $y_i(k)$, který je zároveň vstupem do neuronu *j*. Lokální gradient neuronu *j* je definován vztahem (3-34).

$$\delta_j(k) = -\left. \frac{\partial E(u_j)}{\partial u_j} \right|_{u_j=u_j(k)} \tag{3-34}$$

Důkaz ekvivalence vztahů (3-32) a (3-33) je možno provést rozepsáním vztahu (3-34) do vztahu (3-35).

$$\delta_j(k) = -\frac{\partial E}{\partial e_j} \cdot \frac{\partial e_j}{\partial y_j} \cdot \frac{\partial y_j}{\partial u_j} \tag{3-35}$$

Porovnáním vztahu (3-35) a vztahů (3-27) až (3-29) se získá vztah (3-36).

$$\delta_j(k) = e_j(k) \cdot f'\left[u_j(k)\right] \tag{3-36}$$

Vztah (3-36) s přihlédnutím ke vztahu (3-31) dokazuje ekvivalenci vztahů (3-32) a (3-33), tedy správnost vztahu (3-33) pro výpočet opravy váhy spojení w_{ji} v čase *k*.

Právě lokální gradient je hnací silou pro změny hodnot vah spojení v trénované neuronové síti. Ze vztahu (3-36) je zřejmé, že pro výpočet lokálního gradientu je nutné znát chybu na výstupu z neuronu *j*. A zde mohou nastat dva případy:

- Neuron *j* je obsažen ve výstupní vrstvě
- Neuron *j* je obsažen v některé ze skrytých vrstev

Pokud je neuron *j* výstupní vektor, je výpočet jeho lokálního gradientu jednoduchý. Pro výpočet chyby $e_j(k)$ se použije vztah (3-20).

V případě, že je neuron *j* obsažen ve skryté vrstvě, neexistuje žádná předem daná žádaná hodnota c_j na výstup tohoto neuronu. Žádaná hodnota lze být ovšem stanovena rekurzivně ze všech výstupních chyb neuronů, na které je daný neuron *j* svým výstupem přímo napojen.

V dalším textu tedy bude neuron *j* považován za neuron obsažený ve skryté vrstvě. Neuron *l* pak bude neuron výstupní vrstvy přímo napojený na neuron *j*. Je nutno vyjádřit lokální gradient neuronu *j* v závislosti na známých upravených parametrech neuronu *l*, aby mohly být upraveny i váhy spojení vedoucí do neuronu *j*.

Rozepsáním rovnice (3-34) může být lokální gradient definován také vztahem (3-37).

$$\delta_j(k) = -\left.\frac{\partial E}{\partial y_j}\right|_{y_j=y_j(k)} \cdot \left.\frac{\partial y_j}{\partial u_j}\right|_{u_j=u_j(k)} \tag{3-37}$$

S přihlédnutím ke vztahu (3-29) lze vztah (3-37) upravit na vztah (3-38).

$$\delta_j(k) = -\left.\frac{\partial E}{\partial y_j}\right|_{y_j=y_j(k)} \cdot f'\left[u_j(k)\right] \tag{3-38}$$

Pro vyjádření derivace $\left.\frac{\partial E}{\partial y_j}\right|_{y_j=y_j(k)}$ se upraví vztah (3-22) pro současné podmínky – vztah (3-39).

$$E(k) = \frac{1}{2}\sum_l e_l^{\,2}(k) \quad \ldots \text{ neuron } l \text{ je ve výstupní vrstvě} \tag{3-39}$$

Derivací rovnice (3-39) podle y_j se obdrží vztah (3-40).

$$\left.\frac{\partial E(y_j)}{\partial y_j}\right|_{y_j=y_j(k)} = \sum_l e_l(k) \cdot \left.\frac{\partial e_l(y_j)}{\partial y_j}\right|_{y_j=y_j(k)} \tag{3-40}$$

Rovnici (3-40) lze přepsat na vztah (3-41).

$$\left.\frac{\partial E(y_j)}{\partial y_j}\right|_{y_j=y_j(k)} = \sum_l e_l(k) \cdot \left.\frac{\partial e_l(u_l)}{\partial u_l}\right|_{u_l=u_l(k)} \cdot \left.\frac{\partial u_l(y_j)}{\partial y_j}\right|_{y_j=y_j(k)} \tag{3-41}$$

Ovšem úpravou rovnice (3-20) s přihlédnutím ke vztahu (3-25) lze psát vztah (3-42) resp. (3-43).

$$e_l(k) = y_{Sl}(k) - y_l(k) \tag{3-42}$$

$$e_l(k) = y_{Sl}(k) - f\left[u_l(k)\right] \tag{3-43}$$

Derivací obou stran rovnice (3-43) podle u_l bude získán vztah (3-44).

$$\left.\frac{\partial e_l(u_l)}{\partial u_l}\right|_{u_l=u_l(k)} = -f'\left[u_l(k)\right] \tag{3-44}$$

Je třeba si také uvědomit, že analogicky podle vztahu (3-24) platí vztah (3-45).

$$u_l(k) = \sum_{j=0}^{m} w_{lj}(k) \cdot y_j(k) \tag{3-45}$$

Derivací rovnice (3-45) podle y_j se získá vztah (3-46).

$$\left.\frac{\partial u_l(y_j)}{\partial y_j}\right|_{y_j = y_j(k)} = w_{lj}(k) \tag{3-46}$$

Použitím rovnic (3-44) a (3-46) do vztahu (3-41) se obdrží vztah (3-47).

$$\left.\frac{\partial E(y_j)}{\partial y_j}\right|_{y_j = y_j(k)} = -\sum_l e_l(k) \cdot f'[u_l(k)] \cdot w_{lj}(k) \tag{3-47}$$

Pomocí definice lokálního gradientu (3-36) lze rovnici (3-47) upravit na tvar (3-48).

$$\left.\frac{\partial E(y_j)}{\partial y_j}\right|_{y_j = y_j(k)} = -\sum_l \delta_l(k) \cdot w_{lj}(k) \tag{3-48}$$

Nakonec, pokud se vztah (3-48) aplikuje na rovnici (3-38), získá se vztah pro lokální gradient zpětného šíření chyby (3-49).

$$\delta_j(k) = f'\left[u_j(k)\right] \cdot \sum_l \delta_l(k) \cdot w_{lj}(k) \tag{3-49}$$

kde neuron *j* je obsažen ve skryté vrstvě, neuron *l* je obecně o jednu vrstvu blíže výstupu.

Nyní je možno určit lokální gradienty pro neurony ve skrytých vrstvách, je tedy možno pro každou váhu spojení v neuronové sítí použít vztah (3-33).

Předchozí výklad tedy stručně popisuje odvození vztahů (3-33), (3-35) a (3-49), které tvoří páteř algoritmu zpětného šíření chyby. Podrobnější rozbor uvádí zejména [Haykin, 1999].

Postup výpočtu korekcí vah spojení Δw_{ji} může být proveden v zásadě dvěma způsoby:

- Skupinový (off-line)
- Postupný (on-line)

Ve skupinovém módu jsou váhy spojení aktualizovány až po využití celé tréninkové množiny dat (tzn. po jedné epoše), zatímco v postupném módu jsou aktualizovány po průchodu každého prvku tréninkové množiny dat.

Celkový postup BPG algoritmu:

1) Úvodní inicializace vah spojení a prahů
2) Volba vhodné tréninkové množiny dat
3) Dopředný výpočet:
   Vrstvu po vrstvě se nechá vstupní signál projít sítí a použitím vztahů (3-24) a (3-25) se vypočte vektor výstupů sítě. Dále se pomocí vztahu (3-20) vypočte chyba na výstupu.
4) Zpětný výpočet:
   Vypočtou se lokální gradienty sítě definované vztahy (3-36) (pro neuron ve výstupní vrstvě) resp. (3-49) (pro neuron ve skryté vrstvě). Dále se pomocí vztahu (3-33) a (3-11) vypočtou nové váhy spojení. Parametr α (koeficient rychlosti učení) se volí tak, aby chybová funkce co nejrychleji konvergovala ke svému minimu.
5) Iterační výpočet:
   Body 3) a 4) se opakují tak dlouho, dokud se nedosáhne požadovaných podmínek.

BPG algoritmus způsobil přelom ve vývoji neuronových sítí a právě díky němu byly vyvráceny skeptické názory ohledně neuronových sítí (např. práce matematiků Minskeho a Paperta) a nastala renesance zájmu o problematiku umělých neuronových sítí v osmdesátých letech. Od doby svého vzniku byl mnohokrát modifikován a zrychlen, byly také navrženy nové architektury neuronových sítí a algoritmy jejich učení. Dvě modifikace budou uvedeny v následujících odstavcích.

3.5.3.4 Kvazi-Newtonova metoda

Kvazi-Newtonova metoda využívá *Newtonův optimalizační algoritmus*, jehož základním vztahem aplikovaným na problematiku trénování vícevrstvých dopředných neuronových sítí je rovnice (3-50), přičemž i zde platí vztah (3-11).

$$\Delta \boldsymbol{w}(k) = -\boldsymbol{H}^{-1}(k) \cdot \boldsymbol{g}(k) \qquad (3\text{-}50)$$

Vztah (3-50) je obdržen následujícím postupem:

Nechť $E_{AV}(\boldsymbol{w})$ je průměrná kriteriální funkce za celou tréninkovou množinu definovaná vztahem (3-23). Pak se hledá odhad vektoru gradientu $\frac{\partial E_{AV}(\boldsymbol{w})}{\partial \boldsymbol{w}}$. (Použití průměrné kriteriální funkce vyžaduje použití skupinového módu učení). Pak lze tuto funkci podle *Taylorova polynomu* rozepsat do druhého stupně vztahem (3-51) , jak uvádí Haykin, [1999].

$$E_{AV}\left(\boldsymbol{w}(k)+\Delta\boldsymbol{w}(k)\right)=E_{AV}\left(\boldsymbol{w}(k)\right)+\boldsymbol{g}^{\mathrm{T}}(k)\Delta\boldsymbol{w}(k)+\frac{1}{2}\Delta\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{H}(k)\Delta\boldsymbol{w}(k) \quad (3\text{-}51)$$

kde: $$\boldsymbol{g}(k)=\left.\frac{\partial E_{AV}(\boldsymbol{w})}{\partial\boldsymbol{w}}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} \quad (3\text{-}52)$$

$$\boldsymbol{H}(k)=\left.\frac{\partial^2 E_{AV}(\boldsymbol{w})}{\partial\boldsymbol{w}^2}\right|_{\boldsymbol{w}=\boldsymbol{w}(k)} \quad (3\text{-}53)$$

Z rovnice (3-51) pak lze derivací odvodit, že optimální hodnota (minimum) účelové funkce E_{AV} pro vektor změn vah spojů je dána vztahem (3-50).

Zatímco *BPG algoritmus* používá pouze lineární aproximaci kriteriální funkce v okolí pracovního bodu $\boldsymbol{w}(k)$, v tomto případě se použije aproximace vyššího řádu.

Takto použitá *Newtonova metoda* má ovšem řadu omezení:

- Je požadován výpočet inverze *Hessovy matice* $\boldsymbol{H}^{-1}$, což může být výpočetně náročné.
- Aby mohla být matice $\boldsymbol{H}^{-1}$ vypočtena, matice $\boldsymbol{H}$ musí být regulární, což není zaručeno.
- Pokud není kriteriální funkce E_{AV} kvadratickou funkcí, nemusí *Newtonova metoda* konvergovat.

Proto se v praxi při trénování umělých neuronových sítí používá zjednodušená *Newtonova metoda – kvazi-Newtonova metoda,* která některé z těchto problémů řeší. Nepočítá druhé derivace a inverzi matice $\boldsymbol{H}$ analyticky, ale iterativně jako funkce gradientu. V samotné matici $\boldsymbol{H}$ také zanedbává nediagonální prvky. Tato opatření vedou ke zvýšení rychlosti a stability výpočtu.

Podrobnější popis a odvození je uvedeno zejména v [Haykin, 1999].

3.5.3.5 Levenbergův-Marquardtův algoritmus

Učení umělých neuronových sítí *metodou Levenbergovou-Marquardtovou* (*LM algoritmus*) je jakousi kombinací *BPG algoritmu* a *kvazi-Newtonovy metody*. Předpokládá účelovou funkci ve tvaru sumy čtverců, což je pro trénování dopředných vícevrstvých umělých neuronových sítí typické, a za tohoto předpokladu může být *Hessova matice* $\boldsymbol{H}$ aproximována pomocí *Jacobiho matice***,** jak je uvedeno v rovnici (3-54).

$$\boldsymbol{H}=\boldsymbol{J}^{\mathrm{T}}\boldsymbol{J} \quad (3\text{-}54)$$

Celý algoritmus pak lze podobně jako *kvazi-Newtonovu metodu* vyjádřit za platnosti rovnice (3-11) rovnicí (3-55).

$$\Delta \boldsymbol{w}(k) = -\left[\boldsymbol{J}^{\mathrm{T}}(k)\boldsymbol{J}(k) + \mu \boldsymbol{I}(k) \right]^{-1} \boldsymbol{g}(k) \tag{3-55}$$

Kde:

$$\boldsymbol{g}(k) = \left. \frac{\partial E_{AV}(\boldsymbol{w})}{\partial \boldsymbol{w}} \right|_{\boldsymbol{w}=\boldsymbol{w}(k)} = \boldsymbol{J}^{\mathrm{T}}(k)\boldsymbol{e}(k) \tag{3-56}$$

Pokud je velikost skalární hodnoty μ nulová, přechází *Levenbergův-Marquardtův algoritmus* na *kvazi-Newtonovu metodu*, zatímco když je vysoká, blíží se *BPG algoritmu* s nízkým koeficientem rychlosti učení. Během učení se parametr μ adaptivně mění takovým způsobem, že po každé úspěšné iteraci (hodnota kriteriální funkce se sníží) se hodnota parametru μ sníží a zvýší se, pouze pokud byla v aktuální iteraci hodnota kriteriální funkce zvýšena.

Levenbergův-Marquardtův algoritmus bývá často ze všech tří zmiňovaných nejrychlejší, ovšem je velice náročný na paměť, proto se hodí k trénovaní sítí o jednodušších topologiích.

3.5.4 Celkový postup návrhu topologie a natrénování neuronové sítě

Celkový postup probíhá v několika bodech.

1) **Zisk trénovací a testovací množiny**

   Pro získání trénovací a testovací množiny je třeba provést experiment, jehož výsledkem jsou data, která popisují charakter úkolu v rámci celé pracovní oblasti. Již v této části jsou potřeba učinit některá klíčová rozhodnutí, v případě identifikace modelu například volba délky periody vzorkování.

   Získaná data se pak většinou rozdělí náhodně do dvou stejně početných množin, přičemž jedna bude považována za trénovací množinu dat a druhá za testovací množinu dat. V žádném případě by se však neměla používat shodná trénovací a testovací množina dat.

2) **Volba architektury neuronové sítě z hlediska průchodu informací neuronovou sítí a z hlediska hodnot parametrů v neuronových sítích**

   Tato volba se provádí na počátku řešení daného úkolu podle jeho charakteru a požadavků. Je tedy třeba zvážit, jestli je třeba zvolit rekurentní neuronovou síť či postačí pouze dopředná síť a jestli charakteru úkolu odpovídá síť se statickými parametry či bude výhodnější použít síť s dynamicky se měnícími parametry. Zde se také volí agregační a aktivační funkce jednotlivých neuronů.

3) **Volba architektury neuronové sítě z hlediska počtu vrstev neuronů a uspořádání neuronů v jednotlivých vrstvách a zhodnocení dané volby**

   V této části se volí určitý počet vrstev a neuronů v neuronové síti a provede se trénování neuronové sítě. Kvalita dané neuronové sítě se hodnotí podle

konečné hodnoty účelové funkce na konci trénování. Porovnávají se různé topologie sítí.

V praxi se postupuje v zásadě dvěma způsoby. Buď se nejprve zvolí redundantní neuronová síť – síť s vysokým počtem vrstev a neuronů. Tato síť se pak postupně zjednodušuje až do takového stavu, kdy co nejjednodušší síť poskytuje ještě uspokojivé výsledky.

Druhou možností je počáteční volba velmi jednoduché topologie sítě a tato se postupně obohacuje o další neurony a vrstvy tak dlouho, dokud ještě ve výkonu sítě nastává zlepšení.

4) Testování neuronové sítě

U vítězné neuronové sítě z předchozího kroku se provede testování platnosti. Na vstup sítě se přivádí testovací vstup a porovnává se výstup sítě s požadovaným výstupem. Vyhodnocení se zpravidla provádí pomocí sumy kvadrátů odchylek požadovaného a skutečného výstupu, případně pomocí grafického zobrazení absolutních chyb. Velmi kvalitní zhodnocení platnosti natrénované neuronové sítě je možno obdržet pomocí různých korelačních testů, které se ale vzhledem k jejich náročnosti používají jen zřídka.

Pokud neuronová síť při testování uspěje, prohlásí se za platnou a postup se ukončí, pokud neuspěje, postup se vrátí k jednomu z předchozích bodů.

Častými důvody neúspěchu bývá nalezení pouze lokálního minima účelové funkce při trénování (je pak třeba opakovat proces trénování), špatná volba aktivačních funkcí v jednotlivých neuronech a v nejhorším případě nedostatečná trénovací množina, kdy je pak nutno opakovat experimentální zisk dat.

3.5.5 Použití neuronových sítí

Jednou z největších výhod neuronových sítí je schopnost učit se, tedy možnost získávání vědomostí pomocí množiny vzorů bez nutnosti explicitní znalosti algoritmu řešení. Každá správně navržená a natrénovaná neuronová síť má pak také schopnost generalizace, tedy schopnost vhodně zpracovat i ty signály, které neobsahovala tréninková množina dat. Z hlediska provozu je velká výhoda neuronových sítích ta, že při výpadku malé části neuronové sítě tato porucha způsobí pouze jisté zkreslení výsledku, nenastane porucha celého zařízení.

Mezi nevýhody je možno zařadit obtížnou volbu optimální architektury sítě, velikost a složitost sítí, dobu potřebnou k natrénování a obtížné zjištění, zda síť správně generalizuje. Stále se musí jako určité omezení brát i výkon použité výpočetní techniky.

Neuronové sítě se s úspěchem používají tam, kde je třeba aproximovat požadované funkční hodnoty, při kontrole a řízení různých fyzikálních procesů, při neznalosti algoritmu řešení, při složitém matematickém popisu, při neúplných, nepřesných či neurčitých informacích, všude tam, kde klasické metody neposkytují uspokojivé výsledky.

Nevhodné jsou neuronové sítě jako prostředek pro přesné výpočty, či pro řešení úloh s jednoduchým matematickým popisem.

4 Popis bioreaktoru

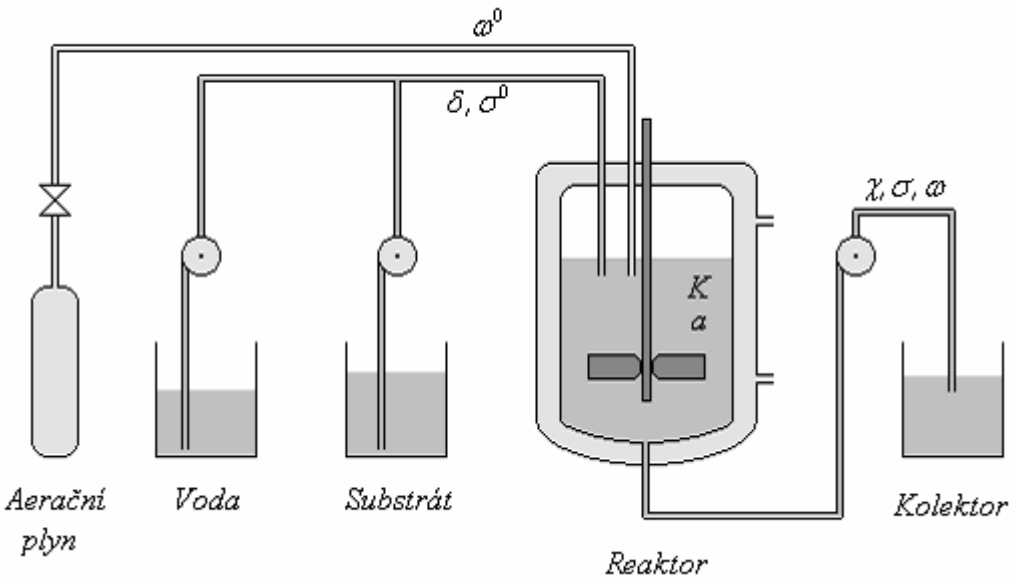
4.1 Úvod

Bioreaktor je zařízení, ve kterém dochází k přeměně látek biochemickou reakcí pomocí enzymů nebo mikroorganismů (tzv. fermentace). Fermentačně se získávají například antibiotika (penicilin, streptomycin), kyselina citrónová, kyselina mléčná, vitaminy (kyselina askorbová, riboflavin), steroidy (kortison, hydrokortison), či alkoholy (ethanol, butanol). Dále se používá pro biologické čistění odpadních vod.

Jedním z cílů této práce je navržení různých statických a dynamických modelů průtočného reaktoru pomocí neuronové sítě a využití těchto modelů při řízení, k čemuž postačují pouze hodnoty vstupů a výstupů soustavy, ovšem tyto hodnoty byly získány z matematicko-fyzikálního modelu, jehož popis bude stručně uveden v následujících odstavcích.

4.2 Popis průtočného bioreaktoru

Kontinuální bioreaktor je válcová nádoba vertikálně rozdělená na několik pater. Vstupem je substrát zředěný vodou, který se probublává aeračním plynem, produktem je pak biomasa kultivovaná uvnitř bioreaktoru. Schéma bioreaktoru je uvedeno na obrázku 4.1.



Obr. 4.1 – Technologické schéma bioreaktoru

Aerační plyn na vstupu i plyny na výstupu jsou sterilizovány průchodem přes membrány. Zvláštním způsobem se musí zpracovávat i jiné odpadní látky zvláště kvůli obsahu mikroorganismů a substrátu, které by se mohly rozšířit do životního prostředí.

Pracovní proces bioreaktoru (hodnota koncentrace biomasy na výstupu) se reguluje teplotou bioreaktoru, průtokem aeračního plynu, změnou *pH* prostředí a regulací otáček míchadla.

4.3 Matematicko-fyzikální model průtočného bioreaktoru

4.3.1 Úvod

V literatuře je uvedena celá řada různých postupů tvorby modelu průtočného bioreaktoru od poměrně jednoduchých, které jsou ovšem značně idealizované, po velmi složité modely. Vzhledem k tomu, že cílem této práce není ani tak přiblížit se co nejvíce reálné dynamice některého skutečného bioreaktoru, jako demonstrovat sílu modelování pomocí neuronových sítí, byl v literatuře [Hyklová, 2007] vybrán jeden dynamický model, který byl pro zisk trénovacích a testovacích množin použit.

4.3.2 Výchozí podmínky a vztahy

Podmínky platnosti modelu:

- Předpokládá se růst buněk pouze jedné populace
- Uvažuje se kontinuální bioreaktor s ideálním mícháním
- Předpokládá se fáze růstu, nikoliv odumírání mikroorganismů
- Proces probíhá za konstantní teploty, tlaku a *pH* kultivační látky
- Ostatní příměsi jsou oproti množství živné látky zanedbatelné

Za těchto podmínek lze proces fermentace v bioreaktoru popsat rovnicemi (4-1) až (4-3).

$$\frac{dX}{dt} = -DX + \mu_m \cdot \frac{S}{K_S + S} \cdot \frac{C}{K_C + C} \cdot X \tag{4-1}$$

$$\frac{dS}{dt} = D \cdot (S_0 - S) - \frac{1}{Y_S} \cdot \mu_m \cdot \frac{S}{K_S + S} \cdot \frac{C}{K_C + C} \cdot X \tag{4-2}$$

$$\frac{dC}{dt} = K_L a \cdot (C^R - C) - \frac{1}{Y_C} \cdot \mu_m \cdot \frac{S}{K_S + S} \cdot \frac{C}{K_C + C} \cdot X \tag{4-3}$$

Zjednodušenou slovní interpretaci rovnic (4-1) až (4-3) lze shrnout do následujících slov:

- Přírůstek množství biomasy v objemu kultivační tekutiny (*dX*) za dobu *dt* je roven množství biomasy v objemu kultivační tekutiny v současném okamžiku zmenšenému o množství biomasy z reaktoru odebrané za dobu *dt*.

- Přírůstek množství substrátu v objemu kultivační tekutiny (*dS*) za dobu *dt* je roven množství substrátu přiváděného do reaktoru zmenšenému o množství substrátu odváděného z reaktoru a množství substrátu spotřebovaného populací buněk za dobu *dt*.

- Přírůstek koncentrace kyslíku v bioreaktoru (*dC*) za dobu *dt* je roven množství kyslíku přicházejícího s aeračním plynem zmenšenému o množství kyslíku spotřebovaného populací buněk za dobu *dt*.

4.3.3 Převod rovnic na vztahy bezrozměrných veličin

Z chemicko-inženýrského hlediska je vhodné převést rovnice (4-1) až (4-3) na rovnice využívající výhradně bezrozměrné veličiny, čehož se dosáhne zavedením následujících substitucí. Je vhodné dbát na to, aby zavedené substituce měly fyzikální smysl.

$$a = \frac{K_S Y_S}{K_C Y_C}$$ Fyziologický koeficient charakterizující biochemickou reakci (4-4)

$$K = \frac{K_L a}{\mu_m}$$ Objemový koeficient výměny hmoty charakterizující reaktor (4-5)

$$\omega = \frac{C}{K_C}$$ Průběžná koncentrace kyslíku v tekutině (4-6)

$$\omega^0 = \frac{C^R}{K_C}$$ Rovnovážná koncentrace kyslíku (4-7)

$$\chi = \frac{X}{K_S Y_S}$$ Průběžná koncentrace biomasy v tekutině (4-8)

$$\sigma = \frac{S}{K_S}$$ Průběžná koncentrace substrátu v tekutině (4-9)

$$\sigma^0 = \frac{S_0}{K_S}$$ Počáteční koncentrace substrátu (4-10)

$$\delta = \frac{D}{\mu_m}$$ Průtok suroviny (4-11)

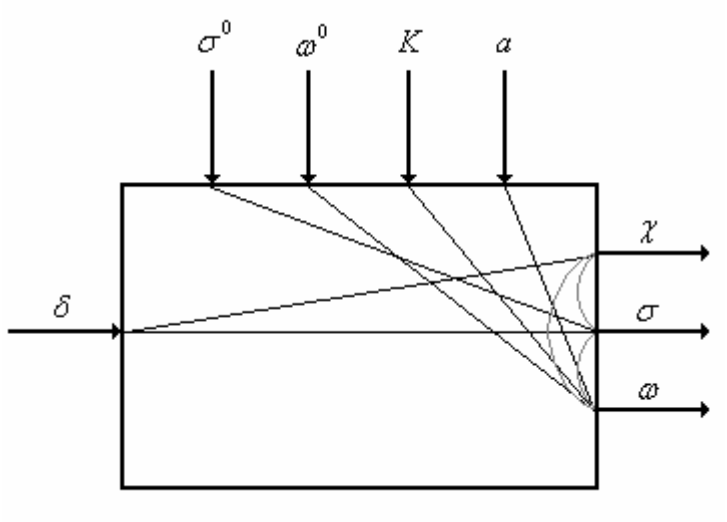
Pokud se takto zavedené substituce využijí při úpravách rovnic (4-1) až (4-3), budou nakonec obdrženy vztahy (4-12) až (4-14).

$$\frac{d\chi}{d\tau} = -\delta \cdot \chi + \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \quad (4\text{-}12)$$

$$\frac{d\sigma}{d\tau} = \delta \cdot \left(\sigma^0 - \sigma\right) - \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \quad (4\text{-}13)$$

$$\frac{d\omega}{d\tau} = K \cdot \left(\omega^0 - \omega\right) - a \cdot \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \quad (4\text{-}14)$$

Uvedené rovnice tedy popisují dynamický režim průtočného bioreaktoru. Je patrné, že popis je vícerozměrový a nelineární, proto je vhodné (za předpokladu, že je soustava rovnic (4-12) až (4-14) chápána jako na reálný proces) modelovat ho pomocí neuronové sítě. Blokové schéma procesu popsaného rovnicemi (4-12) až (4-14) je znázorněno na obrázku 4.2.



Obr. 4.2 – Blokové schéma bioreaktoru

4.3.4 Statický režim průtočného bioreaktoru

Statický režim bioreaktoru odpovídá ustálenému stavu, tedy se proměnné v soustavě rovnic (4-12) až (4-14) nemění s časem, ale pouze každá v závislosti na hodnotách ostatních proměnných. Rovnice (4-12) až (4-14) v ustáleném stavu přejdou na tvary (4-15) až (4-17).

$$0 = -\delta \cdot \chi + \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \tag{4-15}$$

$$0 = \delta \cdot \left(\sigma^0 - \sigma\right) - \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \tag{4-16}$$

$$0 = K \cdot \left(\omega^0 - \omega\right) - a \cdot \frac{\sigma}{1+\sigma} \cdot \frac{\omega}{1+\omega} \cdot \chi \tag{4-17}$$

Triviálním řešením rovnic (4-15) až (4-17) je případ, kdy platí soustava rovnic (4-18).

$$\chi = 0\,,\ \sigma = \sigma^0\,,\ \omega = \omega^0 \tag{4-18}$$

Pokud se bioreaktor při reálném provozu ustálí právě při hodnotách vyjádřených soustavou rovnic (4-18), jeho stav odpovídá režimu vymývání biomasy. Tento neproduktivní jev nastává při rychlostech průtoku suroviny δ větších nebo minimálně rovných rychlosti vymývání δ_V, která je definovaná vztahem (4-19).

$$\delta_V = \frac{\sigma^0}{1+\sigma^0} \cdot \frac{\omega^0}{1+\omega^0} \tag{4-19}$$

Velký průtok suroviny způsobuje krátkou dobu přítomnosti mikroorganismů v reaktoru, mikroorganismy se nestačí množit a pouze se vymývají.

Naopak režim tvorby biomasy nastává při platnosti rovnice (4-20), což způsobí neplatnost soustavy rovnic (4-18). Statický model tohoto režimu je přirozeně z hlediska identifikace zajímavější.

$$\delta < \delta_V \tag{4-20}$$

Vzhledem k tomu, že ze tří výstupů bioreaktoru je zajímavá zejména hodnota koncentrace biomasy na výstupu χ, budou další úpravy rovnic (4-15) až (4-17) směřovat ke tvaru (4-21).

$$\chi = f(\delta, \sigma, \omega, a, K) \tag{4-21}$$

Součtem rovnic (4-15) a (4-16) a následnou úpravou lze za předpokladu $\delta \neq 0$ obdržet rovnici (4-22).

$$\sigma = -\chi + \sigma^0 \tag{4-22}$$

Dále vynásobením rovnice (4-15) výrazem *a*, přičtením výsledné rovnice k rovnici (4-17) a následnou úpravou lze obdržet tvar (4-23).

$$\omega = -\frac{a}{K} \cdot \delta \cdot \chi + \omega^0 \tag{4-23}$$

Dosazením substitucí (4-22) a (4-23) do rovnice (4-15) lze získat z pohledu proměnné χ kvadratickou rovnici (4-24), jejíž řešení je dáno výrazem (4-25).

$$\delta = \frac{-\chi + \sigma^0}{1 - \chi + \sigma^0} \cdot \frac{-\frac{a}{K} \cdot \delta \cdot \chi + \omega^0}{1 - \frac{a}{K} \cdot \delta \cdot \chi + \omega^0} \cdot \chi \tag{4-24}$$

$$\chi_{1,2} = \frac{1}{2} \cdot \left[\left(\sigma^0 - \frac{\delta}{1-\delta} \right) + \frac{K}{a \cdot \delta} \cdot \left(\omega^0 - \frac{\delta}{1-\delta} \right) \right] \pm$$
$$\pm \frac{1}{2} \cdot \sqrt{\left[\left(\sigma^0 - \frac{\delta}{1-\delta} \right) + \frac{K}{a \cdot \delta} \cdot \left(\omega^0 - \frac{\delta}{1-\delta} \right) \right]^2 - \frac{4 \cdot K}{a \cdot \delta \cdot (1-\delta)} \cdot (1 + \sigma_0)(1 + \omega^0)(\delta_V - \delta)} \tag{4-25}$$

Ze dvou řešení je vybrána fyzikálně možná hodnota, což je hodnota ležící v intervalu $\chi \in (0; \sigma^0)$. Lze ukázat, že této hodnotě odpovídá část výrazu (4-25) se znaménkem mínus, tedy platí řešení (4-26).

$$\chi = \frac{1}{2} \cdot \left[\left(\sigma^0 - \frac{\delta}{1-\delta} \right) + \frac{K}{a \cdot \delta} \cdot \left(\omega^0 - \frac{\delta}{1-\delta} \right) \right] -$$
$$- \frac{1}{2} \cdot \sqrt{\left[\left(\sigma^0 - \frac{\delta}{1-\delta} \right) + \frac{K}{a \cdot \delta} \cdot \left(\omega^0 - \frac{\delta}{1-\delta} \right) \right]^2 - \frac{4 \cdot K}{a \cdot \delta \cdot (1-\delta)} \cdot (1 + \sigma^0)(1 + \omega^0)(\delta_V - \delta)} \tag{4-26}$$

Ostatní výstupní veličiny se dále vypočtou ze vztahů (4-22) a (4-23).

5 Tvorba statického modelu bioreaktoru pomocí umělé neuronové sítě

5.1 Úvod

Cílem tohoto odstavce je vytvořit pomocí umělé neuronové sítě sérii statických modelů bioreaktoru v režimu tvorby biomasy. Protože není k dispozici reálné zařízení, bude pro zisk trénovací množiny použit matematicko-fyzikální model odvozený v odstavci 4.3.4. Z rovnice (4-26) je patrno, že obecný statický model má šest vstupních veličin, ovšem mnohé z nich se za určitých podmínek během experimentu nemění, jsou tedy konstantní. Proto budou vytvořeny tři různé neuronové modely vyjádřené následujícími rovnicemi.

$$\chi = f(\delta) \qquad \text{předp. } K,\ a,\ \omega^0,\ \sigma^0,\ \delta_V \text{ konstantní} \qquad (5\text{-}1)$$

$$\chi = f(\delta, K) \qquad \text{předp. } a,\ \omega^0,\ \sigma^0,\ \delta_V \text{ konstantní} \qquad (5\text{-}2)$$

$$\chi = f(\delta, a) \qquad \text{předp. } K,\ \omega^0,\ \sigma^0,\ \delta_V \text{ konstantní} \qquad (5\text{-}3)$$

Modely budou navrženy pomocí všech popsaných algoritmů učení a jednotlivé algoritmy učení budou zhodnoceny.

Na základě předchozích zkušeností byly pro všechny použité neuronové sítě použity agregační i aktivační funkce fixně. Agregační funkce vždy jako lineární bazické funkce definované vztahem (3-1), aktivační funkce neuronů ve skrytých vrstvách jako hyperbolické tangenty, ve výstupní vrstvě pak jako lineární aktivační funkce.

5.2 SISO model $\chi = f(\delta)$

5.2.1 Generování trénovací a testovací množiny dat

Pro zisk trénovací množiny byla použita rovnice (4-26), přičemž konstantní parametry byly zvoleny na základě informací v literatuře [Hyklová, 2007] následujícím způsobem.

$\sigma^0 = 10$ $\qquad$ $K = 250$

$\omega^0 = 10$ $\qquad$ $a = 4000$

Aby proces v bioreaktoru probíhal v režimu tvorby biomasy, je třeba, aby byla splněna podmínka (4-20), tedy $\delta < \delta_V$. Rychlost vymývání δ_V, která je definována vztahem (4-19), pro konkrétní konstantní parametry nabývá hodnoty $\delta_V = 0{,}826446$. Aby tedy byl dodržen režim tvorby biomasy, nesmí být v žádném případě průtok suroviny δ vyšší než δ_V.

Nezávislá proměnná δ pro trénovací množinu byla tedy volena rovnoměrně na intervalu $(0 \div 0.826446)$ tak, aby byl dosažen vektor o 100 hodnotách. Z rovnice (4-26) pak ke každé hodnotě δ byla vypočtena výstupní hodnota χ.

Testovací množina byla získána analogicky, pouze byla nezávisle proměnná δ volena rovnoměrně na intervalu $(0 \div 0.82)$, čímž byla zajištěna rozdílnost trénovací a testovací množiny dat.

5.2.2 Hledání optimální topologie pro základní BPG algoritmus

Hledání optimální topologie proběhlo ve dvou krocích:

a) Pro předem daný koeficient rychlosti učení byly trénovány sítě o různých topologiích a pro další užití byla vybrána síť s nejmenším kriteriem střední kvadratické chyby.
b) Neuronová síť vybraná předchozím bodem byla trénována pro různé koeficienty rychlosti učení, zkoumala se rychlost konvergence.

Ad a)

Koeficient rychlosti učení je lépe volit pro začátek dostatečně nízký, aby vůbec algoritmus konvergoval. V tomto kroku byl zvolen koeficient rychlosti učení:

$$\alpha = 0.01$$

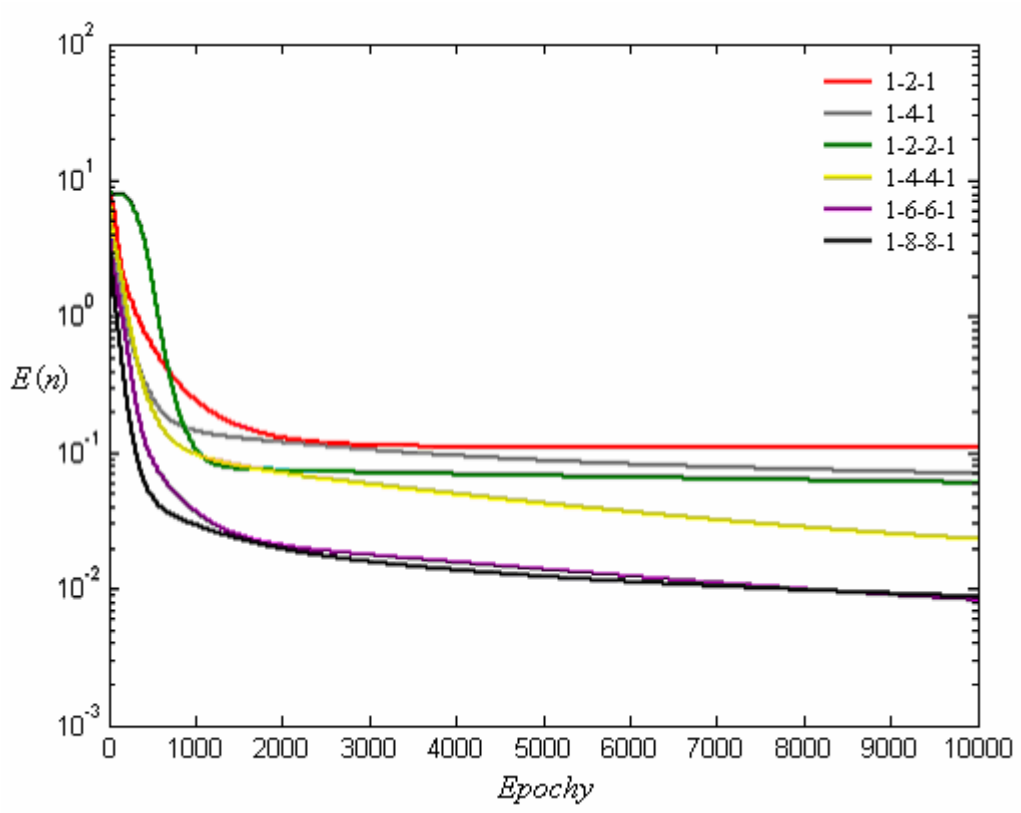
Pro tento koeficient rychlosti učení byly postupně pomocí *BPG algoritmu* trénovány sítě se stále složitějšími topologiemi do té doby, dokud se konečná hodnota kriteria střední kvadratické chyby snižovala. Výsledky jsou shrnuty v tabulce 5.1.

Tab. 5.1 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	1–2–1	1–4–1	1–2–2–1	1–4–4–1	1–6–6–1	1–8–8–1
$E(N)$	0,1081	0,06993	0,06025	0,02295	0,008260	0,008735

Jednotlivé průběhy kriteriálních funkcí (hodnot $E(n)$ v závislosti na počtu epoch) jsou znázorněny na obrázku 5.1.

Z tabulky 5.1 i obrázku 5.1 je patrné, že průběhy kriteriálních funkcí jednotlivých topologií jsou příznivější s rostoucí složitostí topologie až přibližně do topologie 1 – 6 – 6 – 1, s dalším přidáváním neuronů se průběh již zásadně neliší. Proto bude do dalších výpočtů použita topologie 1 – 6 – 6 – 1.



Obrázek 5.1 – Průběhy kriteriálních funkcí pro jednotlivé topologie

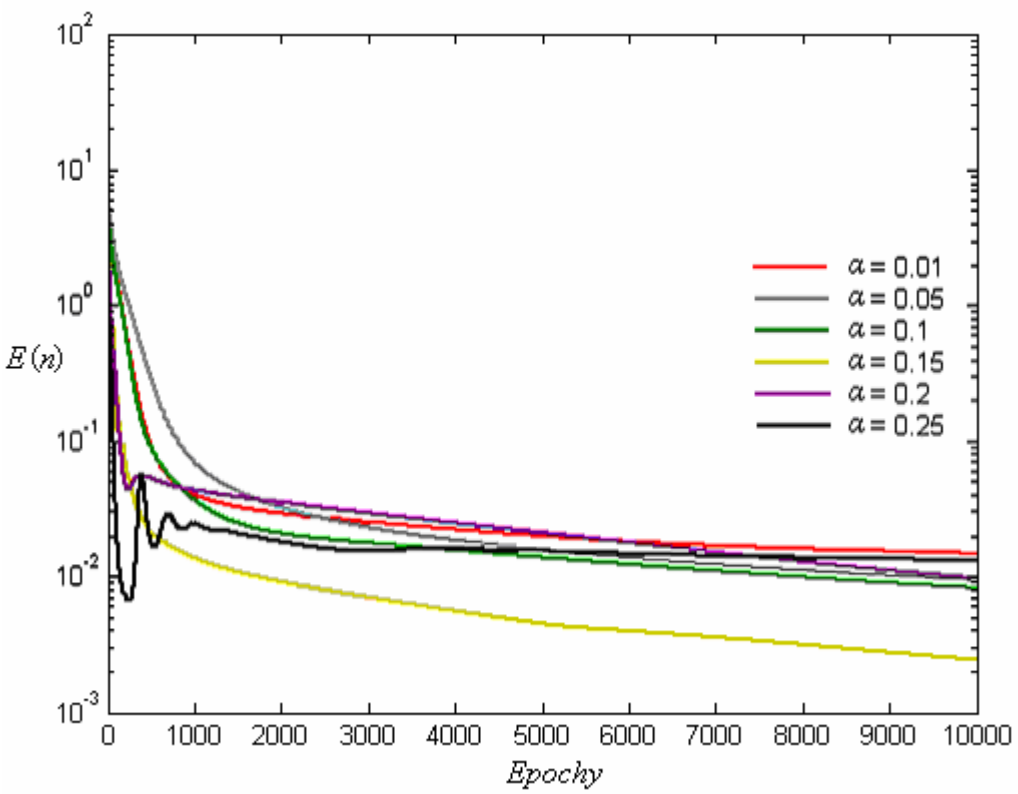
Ad b)

Koeficient rychlosti učení u *BPG algoritmu* ovlivňuje na jedné straně rychlost konvergence algoritmu, na druhé straně pak jeho stabilitu. Proto je nutné vybrat jeho optimální hodnotu. Neuronová síť o topologii 1 – 6 – 6 – 1 byla pomocí *BPG algoritmu* trénována pro různé koeficienty rychlosti učení a sledována byla rychlost konvergence i konečná hodnota kriteria střední kvadratické chyby. Konečné hodnoty kriteria střední kvadratické chyby jsou shrnuty v tabulce 5.2.

Tab. 5.2 – Hodnoty $E(N)$ pro hodnoty koeficientu rychlosti učení

α	0,01	0,05	0,10	0,15	0,20	0,25
$E(N)$	0,01469	0,009481	0,008260	0,002415	0,009683	0,01317

Jednotlivé průběhy kriteriálních funkcí jsou znázorněny na obrázku 5.2.



Obr. 5.2 – Průběhy kriteriálních funkcí pro koeficienty rychlosti učení

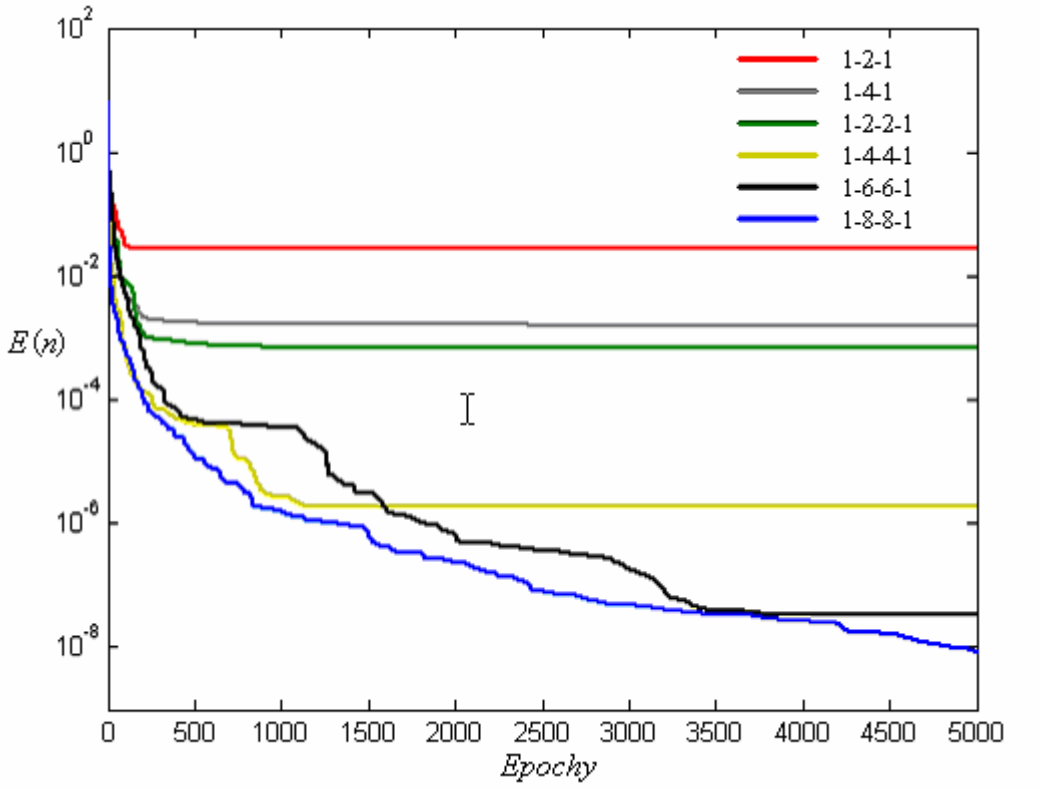
Z obrázku 5.2 je patrné, že vliv koeficientu rychlosti učení na průběh trénování je takový, že se zvyšujícím se koeficientem rychlosti učení je počáteční průběh kriteriální funkce strměji klesající, ovšem hrozí kmitavý průběh a s dalším zvyšováním nestabilita trénovacího algoritmu. Nejpříznivější průběh kriteriální funkce vykazovala pro koeficient rychlosti učení $\alpha = 0.15$. Takto natrénovaná síť bude použita do konečného porovnání pod názvem *BPGnet I.*

5.2.3 Hledání optimální topologie pro trénování pomocí kvazi-Newtonovy metody

Použití *kvazi-Newtonovy metody* pro trénování dopředné vícevrstvé umělé neuronové umožňuje v *Neural Network Toolboxu* funkce *trainBFG*, která již má řadu přednastavených hodnot. Změněn byl pouze počet epoch trénování na 5000. Byly trénovány sítě o různých topologiích a zkoumána byla hodnota kriteriální funkce v závislosti na počtu epoch. Konečné hodnoty kriteriálních funkcí jsou seřazeny v tabulce 5.3, průběhy pak na obrázku 5.3.

Tab. 5.3 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	1–2–1	1–4–1	1–2–2–1	1–4–4–1	1–6–6–1	1–8–8–1
$E(N)$	0,02877	0,001613	$6{,}938.\ 10^{-4}$	$1{,}785.10^{-6}$	$3{,}380.10^{-8}$	$8{,}5258.10^{-9}$



Obr 5.3 – Průběhy kriteriálních funkcí pro jednotlivé topologie

Jak je patrno v tabulce 5.3 i na obrázku 5.3, pro složitější topologie poskytuje *kvazi-Newtonova metoda* velmi uspokojivé výsledky. Vzhledem k tomu, že rozdíl výkonů mezi topologií 1 – 6 – 6 – 1 a 1 – 8 – 8 – 1 je již nevelký, pro další porovnání byla vybrána natrénovaná síť s topologií 1 – 6 – 6 – 1. V dalším textu bude vedena pod označením *BFGnet I*.

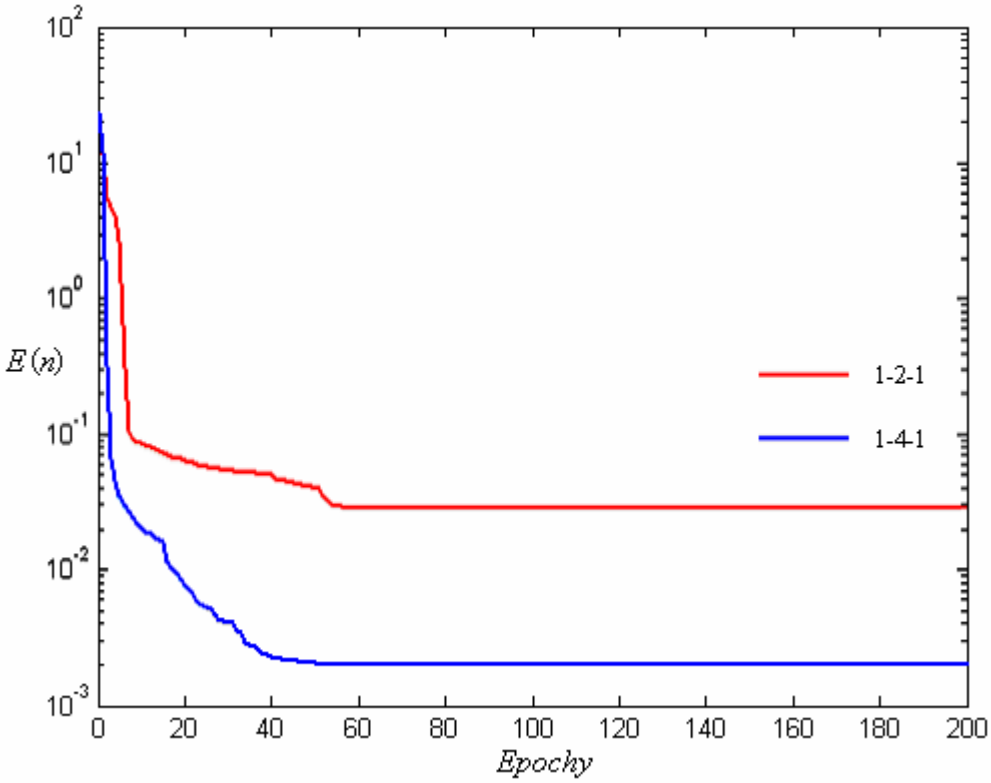
5.2.4 Hledání optimální topologie pomocí Levenbergovy-Marquardtovy metody

Vzhledem k tomu, že trénovací algoritmus vycházející z *Levenbergovy-Marquardtovy metody* tak, jak jej nabízí *Neural Network Toolbox* (funkce *trainLM*), umožňuje z parametrů volit pouze počáteční hodnotu adaptivního parametru μ a omezující podmínky změn μ v každé epoše, výkon učícího algoritmu závisí zejména na topologii neuronové sítě a zhodnocení lze provést v jednom kroku. Algoritmus byl aplikován na neuronové sítě o různých topologiích a byly vynášeny průběhy kriteriálních funkcí. Vzhledem k obecně rychlejší konvergenci byl maximální počet epoch zvolen N = 5000, ovšem pro mnohé topologie algoritmus ke svému minimu konvergoval mnohem dříve. Konečné hodnoty kriteria střední kvadratické chyby jsou seřazeny v tabulce 5.4.

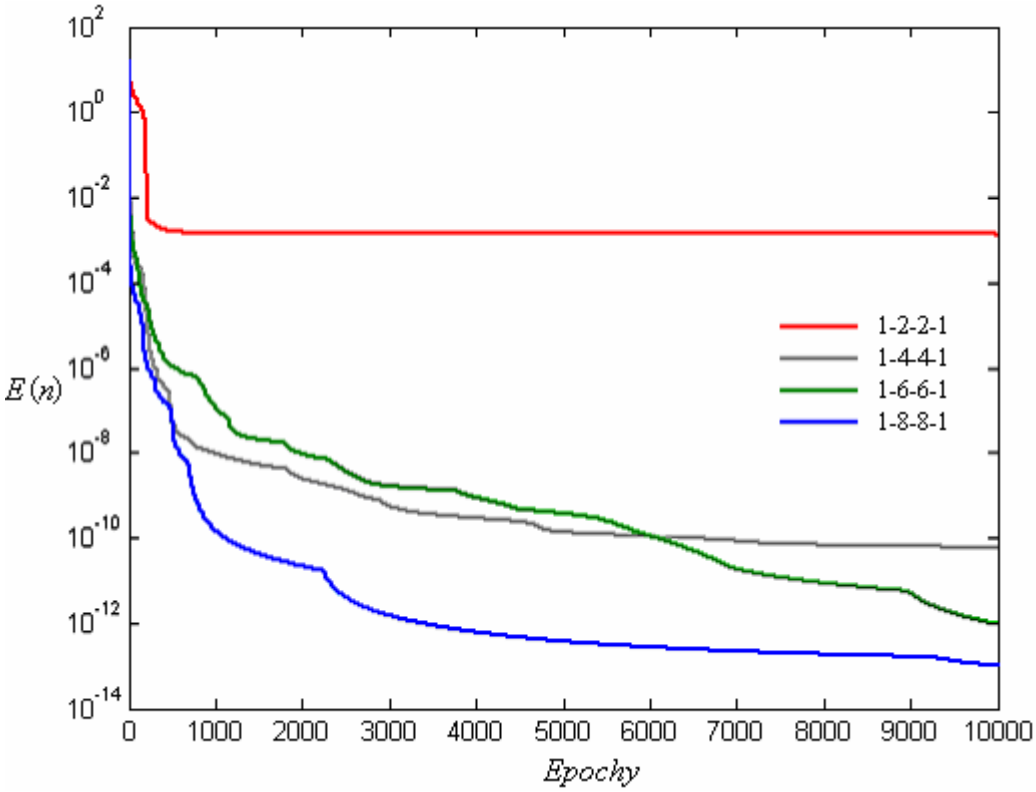
Tab. 5.4 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	1–2–1	1–4–1	1–2–2–1	1–4–4–1	1–6–6–1	1–8–8–1
$E(N)$	0,02877	0,001983	0,001211	$5{,}969.10^{-11}$	$9{,}3545.10^{-13}$	$9{,}6740.10^{-13}$

Jednotlivé průběhy kriteriálních funkcí pro sítě s jednou skrytou vrstvou jsou znázorněny na obrázku 5.4, průběhy kriteriálních funkcí pro sítě se dvěma skrytými vrstvami jsou znázorněny na obrázku 5.5.



Obr. 5.4 – Průběhy kriteriálních funkcí pro sítě s jednou skrytou vrstvou



Obr. 5.5 – Průběhy kriteriálních funkcí pro sítě se dvěma skrytými vrstvami

Jak je vidět z obrázků 5.4 a 5.5, výsledky tréninku jsou s rostoucí složitostí stále příznivější, ovšem pro použití v praxi by bohatě dostačovala síť s topologií 1 – 4 – 4 – 1

s konečnou hodnotou kriteria střední kvadratické chyby rovnu $5{,}969.10^{-11}$. Tato neuronová síť proto bude pod označením *LMnet I* použita pro další srovnání.

5.2.5 Porovnání a zhodnocení jednotlivých algoritmů učení pro SISO model $\chi = f(\delta)$

V tomto odstavci budou porovnány výkony umělých neuronových sítí natrénovaných v minulých oddílech. Vybrané sítě jsou seřazeny v tabulce 5.5.

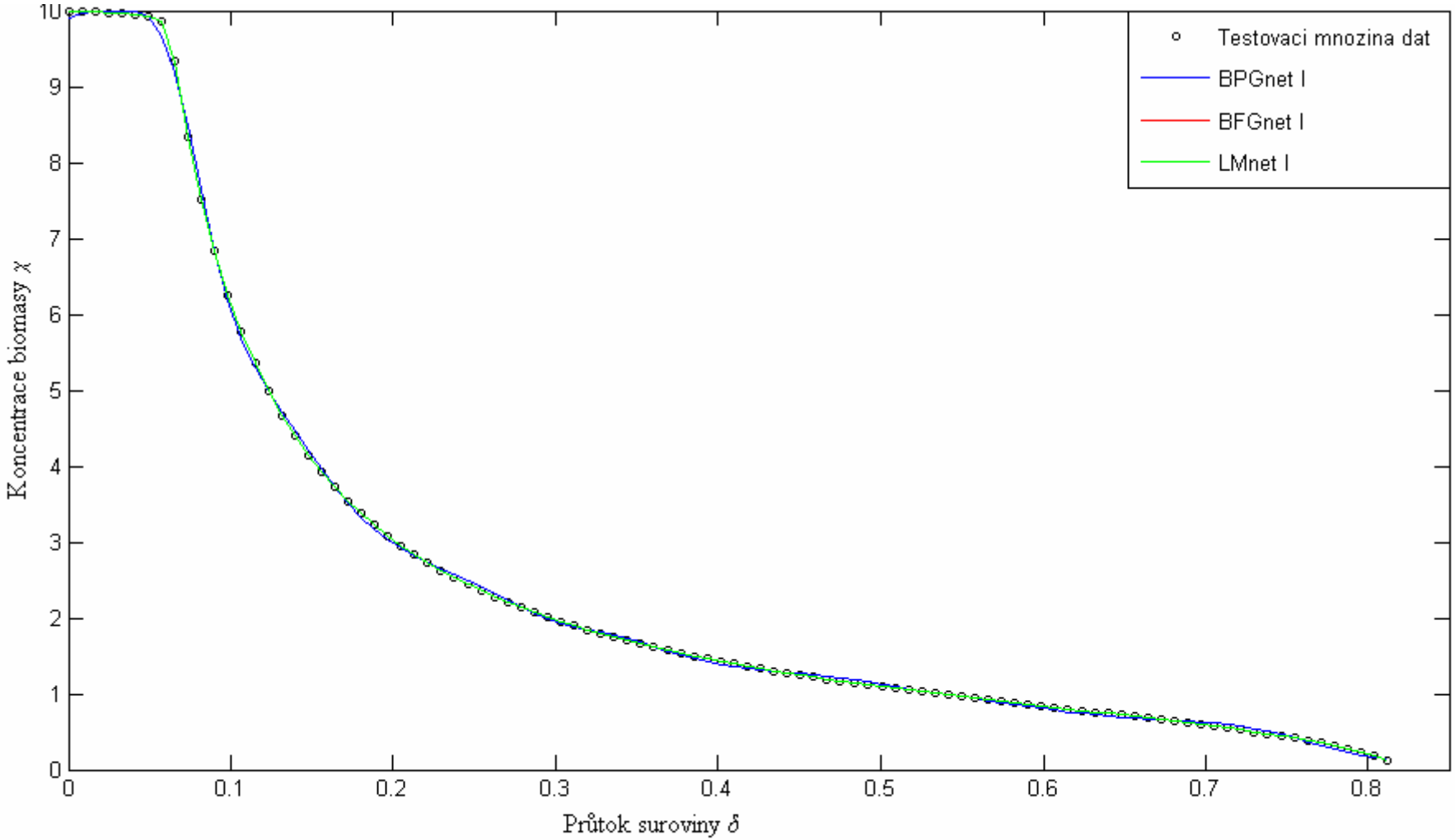
Tab. 5.5 – Seznam sítí pro zhodnocení

Označení sítě	Topologie	Algoritmus trénování	$E(N)$
BPGnet I	1-6-6-1	BPG algoritmus	0,002415
BFGnet I	1-6-6-1	kvazi-Newtonova metoda	$3{,}380.10^{-8}$
LMnet I	1-4-4-1	LM algoritmus	$5{,}969.10^{-11}$

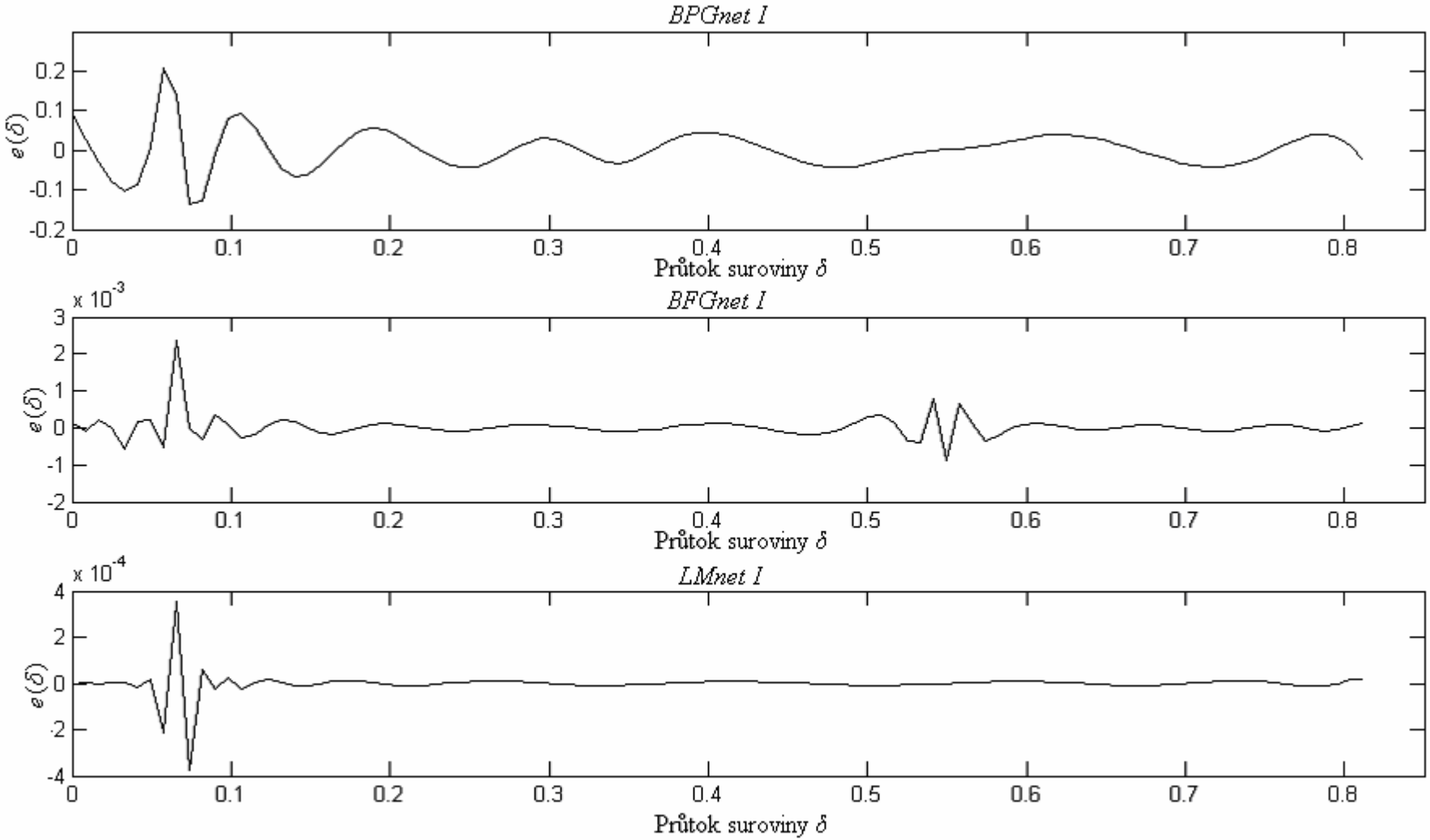
Pro tyto vybrané natrénované umělé neuronové sítě bylo provedeno testování pro testovací množinu dat získanou podle odstavce 5.2.1. Výsledné průběhy společně s testovací množinou dat byly vyneseny do grafu na obrázku 5.6. Do série grafů na obrázku 5.7 pak byly vyneseny odchylky výstupů jednotlivých umělých neuronových sítí od žádaných hodnot podle vztahu (5-1).

$$e(\delta) = \chi(\delta) - \chi_M(\delta) \tag{5-1}$$

Obrázek 5.6 není moc průkazný, neboť na něm všechny křivky téměř splývají, ovšem obrázek 5.7 již jasně prokazuje, který trénovací algoritmus se ukázal pro SISO model jako nejvhodnější. Nejhůře dopadl podle očekávání *BPG algoritmus*, jehož chyba na výstupu se pohybuje v řádu kolem 10^{-1}, zatímco *kvazi-Newtonova metoda* poskytuje výsledky přibližně o dva řády přesnější a *Levenbergův-Marquardtův algoritmus* poskytuje výsledky ještě přesnější i za handicapu jednodušší topologie. Dále je možné si na obrázku 5.7 povšimnout, že největší chyby se u všech algoritmů učení vyskytují kolem $\delta \approx 0{,}08$, což z obrázku 5.6 odpovídá největšímu poklesu hodnoty koncentrace biomasy χ.



Obr. 5.6 - Porovnání výsledků testování neuronových sítí pro SISO model



Obr. 5.7 - Porovnání chyb neuronových sítí pro SISO model

5.3 MISO model $\chi = f(\delta, K)$

5.3.1 Generování trénovací a testovací množiny dat

Tréninková i testovací množina byly získány analogicky podle odstavce 5.2.1. Konstantní parametry byly zvoleny takto:

$$\sigma^0 = 10$$
$$\omega^0 = 10$$
$$a = 4000$$

Opět bylo třeba dodržet podmínku (4-20), proto byla nezávislá proměnná δ pro trénovací množinu volena rovnoměrně na intervalu $(0 \div 0.826446)$ tak, aby byl dosažen vektor o 100 hodnotách a nezávislá proměnná K byla volena na intervalu $(100 \div 400)$ s krokem 50. Takto získané vektory byly sestaveny do matice 2×700 tak, že ve sloupcích byly umístěny uspořádané dvojice [δ; K] stylem každá s každou. Testovací data byla získána analogicky, pouze byla nezávisle proměnná δ volena rovnoměrně na intervalu $(0 \div 0.82)$ a hodnota K byla volena na intervalu $(125 \div 425)$, čímž byla zajištěna rozdílnost trénovací a testovací množiny dat.

5.3.2 Hledání optimální topologie pro základní BPG algoritmus

Hledání optimální topologie proběhlo analogicky k odstavci 5.2.2 ve dvou krocích:

a) Pro předem daný koeficient rychlosti učení byly trénovány sítě o různých topologiích a pro další užití byla vybrána síť s nejmenším kriteriem střední kvadratické chyby.
b) Neuronová síť vybraná předchozím bodem byla trénována pro různé koeficienty rychlosti učení, zkoumala se rychlost konvergence.

Vzhledem ke stejnému postupu jako v odstavci 5.2.2 budou prezentovány pouze výsledné hodnoty a průběhy.

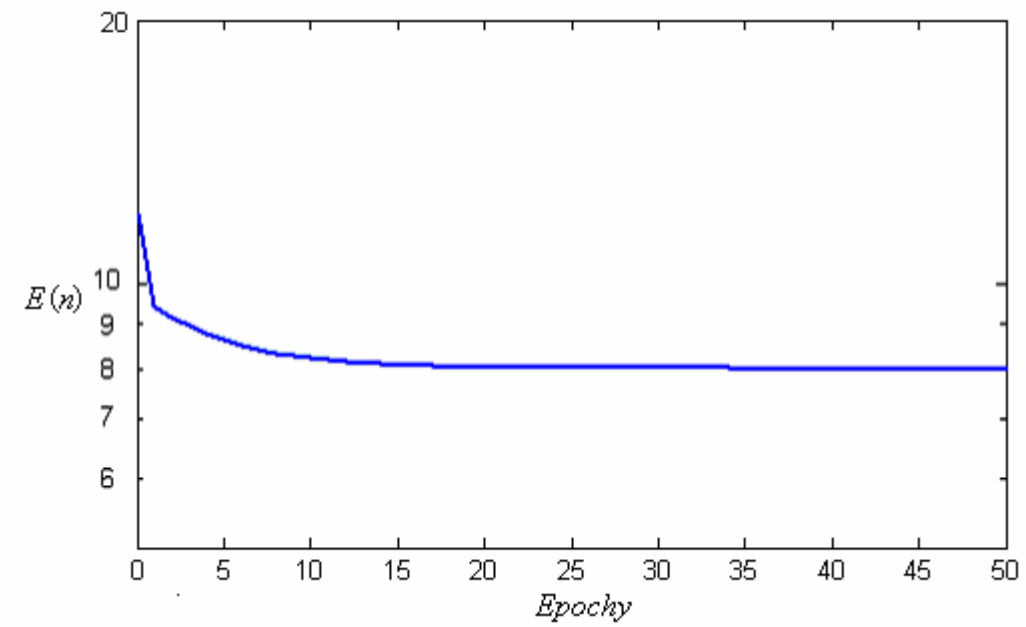
Ad a)
$\alpha = 0.01$

Tab. 5.6 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-20-20-1
$E(N)$	8,011	8,011	8,011	8,011	8,011	8,011

Je patrné, že *BPG algoritmus* tak, jak jej nabízí *Neural Network Toolbox*, si s tímto MISO modelem nedokázal poradit.

Průběhy kriteriálních funkcí byly všechny téměř shodné, ke svému minimu konvergovaly již po několika desítkách iterací. Průběh jednoho zástupce je znázorněn na obrázku 5.8.



Obr. 5.8 – Shodný průběh kriteriálních funkcí

Pro výpočet koeficientu rychlosti učení byla víceméně náhodně vybrána topologie 1 – 8 – 8 – 1.

Ad b)

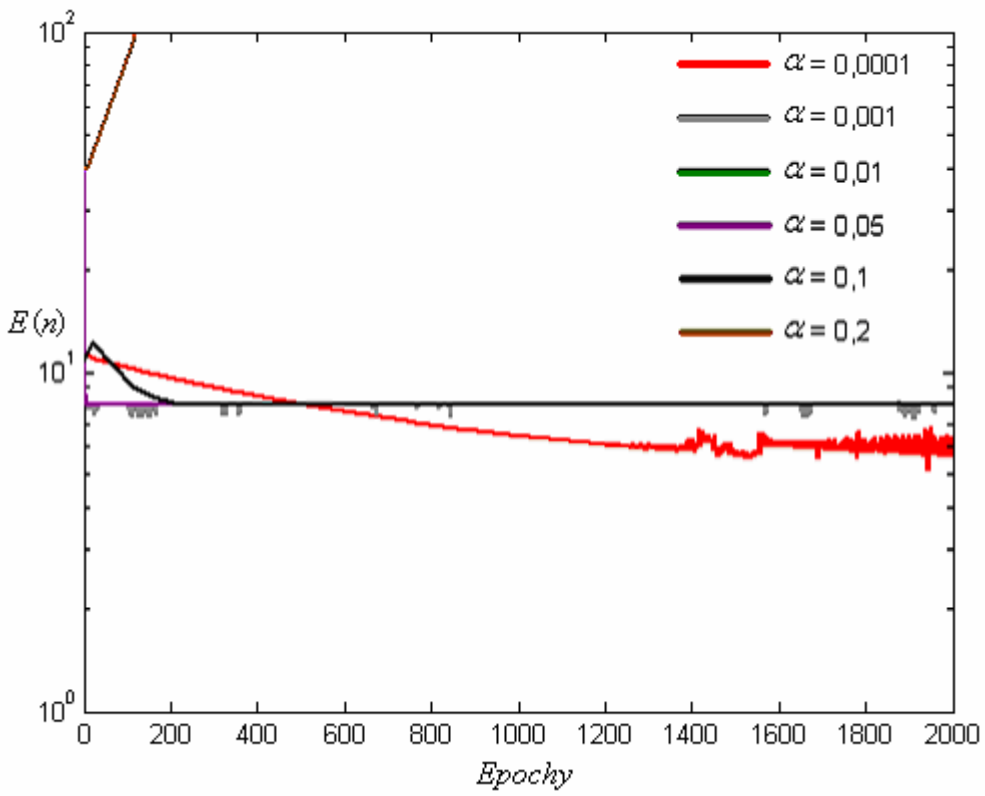
Neuronová síť 1 – 8 – 8 – 1 byla *BPG algoritmem* trénována pro různé koeficienty rychlosti učení, výsledné hodnoty kriteria střední kvadratické chyby jsou shrnuty v tabulce 5.7 a průběhy kriteriální funkce jsou vyneseny do grafu na obrázku 5.9.

Tab. 5.7 – Hodnoty $E(N)$ pro hodnoty koeficientu rychlosti učení

α	0,0001	0,001	0,10	0,15	0,20	0,25
$E(N)$	6,283	7,956	8,011	8,011	8,011	∞

Průběhy kriteriálních funkcí na obrázku 5.9 kromě krajních hodnot koeficientů rychlosti učení splývají v jednu křivku s konečnou hodnotou kriteria střední kvadratické chyby rovnu 8,011, což potvrzuje domněnku, že základní BPG algoritmus tak, jak jej nabízí *Neural Network Toolbox*, nedokáže daný problém uspokojivě vyřešit.

Pro konečné porovnání byla vybrána neuronová síť trénovaná s koeficientem rychlosti učení $\alpha = 0{,}0001$. V dalším textu bude označena názvem *BPGnet II.*



Obr. 5.9 – Průběhy kriteriálních funkcí pro koeficienty rychlosti učení

5.3.3 Hledání optimální topologie pomocí kvazi-Newtonovy metody

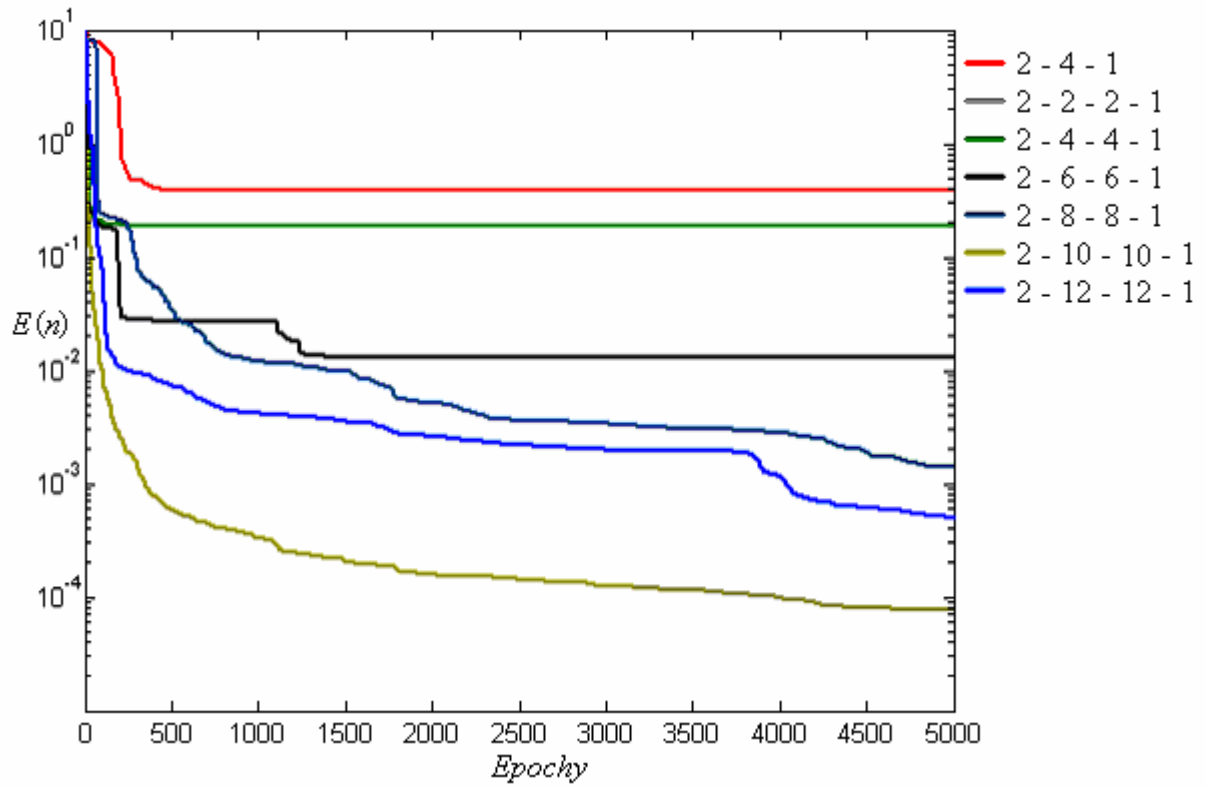
Hledání optimální topologie proběhlo analogicky k odstavci 5.2.3. Výsledné hodnoty kriteria střední kvadratické chyby jsou uvedeny v tabulce 5.8.

Tab. 5.8 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-10-10-1	2-12-12-1
$E(N)$	0,3827	0,1884	0,1861	0,01288	0,001436	$7{,}664.10^{-5}$	$5{,}008.10^{-4}$

Průběhy jednotlivých kriteriálních funkcí jsou uvedeny na obrázku 5.10.

Jak je patrné z tabulky 5.8 a obrázku 5.10, přijatelné hodnoty kriteriální funkce poskytuje *kvazi-Newtonova metoda* až při poměrně složitých topologiích. Vzhledem k tomu, že nejlepších výsledků dosáhla topologie 2 – 10 – 10 – 1, bude do konečného porovnání vybrána tato. Označena v dalším textu bude názvem *BFGnet II*.



Obr. 5.10 – Průběhy kriteriálních funkcí

5.3.4 Hledání optimální topologie pomocí Levenbergovy-Marquardtovy metody

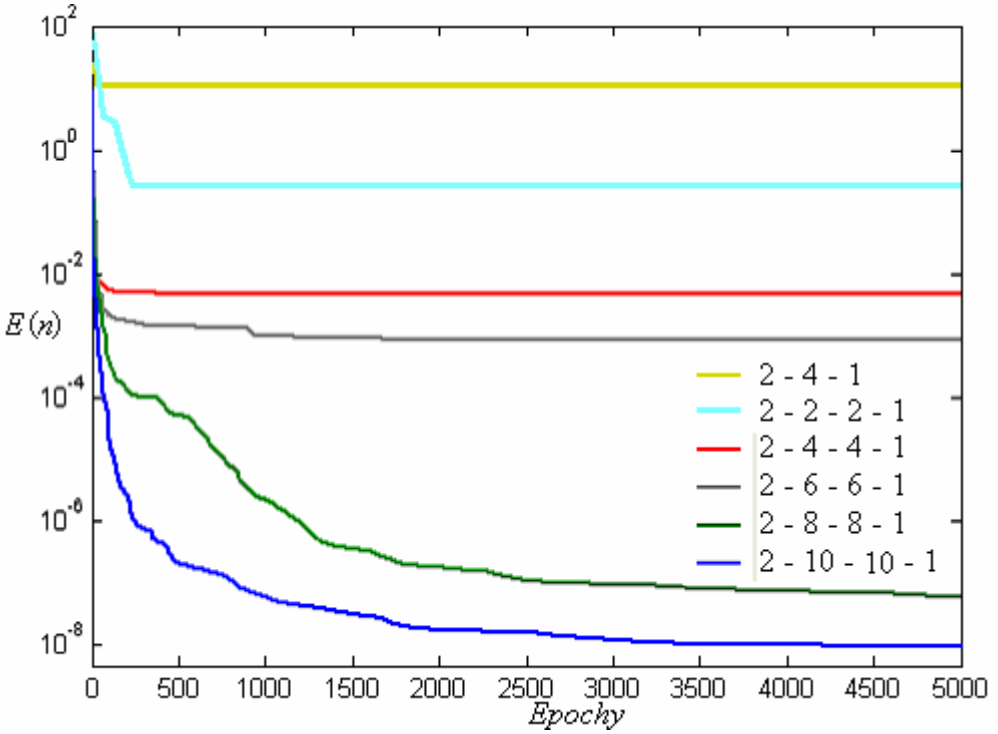
Hledání optimální topologie proběhlo analogicky k odstavci 5.2.4. Výsledné hodnoty kriteria střední kvadratické chyby jsou uvedeny v tabulce 5.9.

Tab. 5.9 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-10-10-1
$E(N)$	0,3823	8,011	0,004697	$8{,}595.10^{-4}$	$6{,}091.10^{-8}$	$9{,}375.10^{-9}$

Průběhy jednotlivých kriteriálních funkcí jsou vyneseny do grafu na obrázku 5.11.

Z tabulky 5.9 a z obrázku 5.11 patrné, že *LM algoritmus* učení si s daným problémem poradí pro vyšší složitosti sítě poměrně úspěšně, přičemž pro praktické účely plně dostačuje natrénovaná síť o topologii 2 – 8 – 8 – 1, proto byla do konečného zhodnocení vybrána tato neuronová síť, která v dalším textu ponese označení *LMnet II*.



Obr. 5.11 – Průběhy kriteriálních funkcí

5.3.5 Hledání Porovnání a zhodnocení jednotlivých algoritmů učení pro MISO model $\chi = f(\delta, K)$

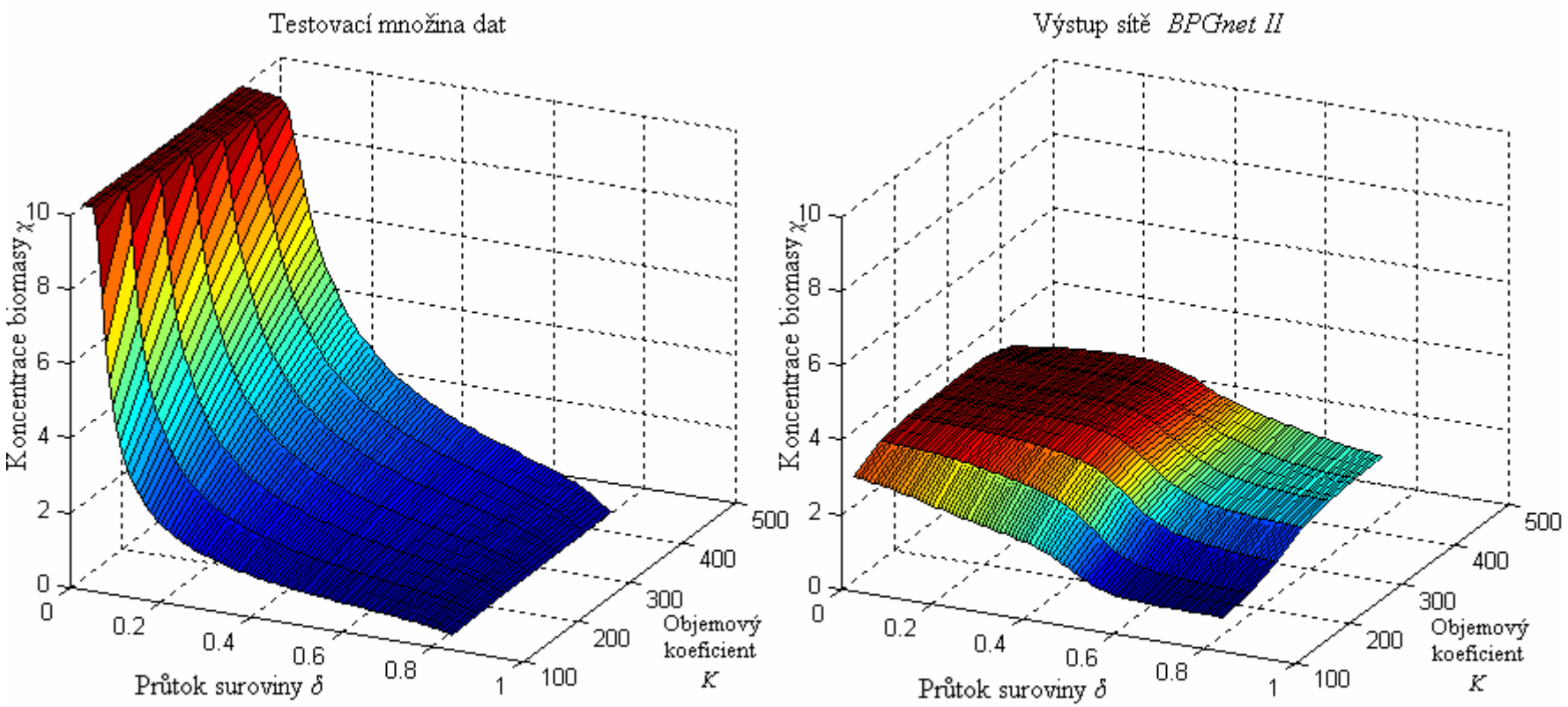
Vybrané sítě jsou seřazeny v tabulce 5.10.

Tab. 5.10 – Seznam sítí pro zhodnocení

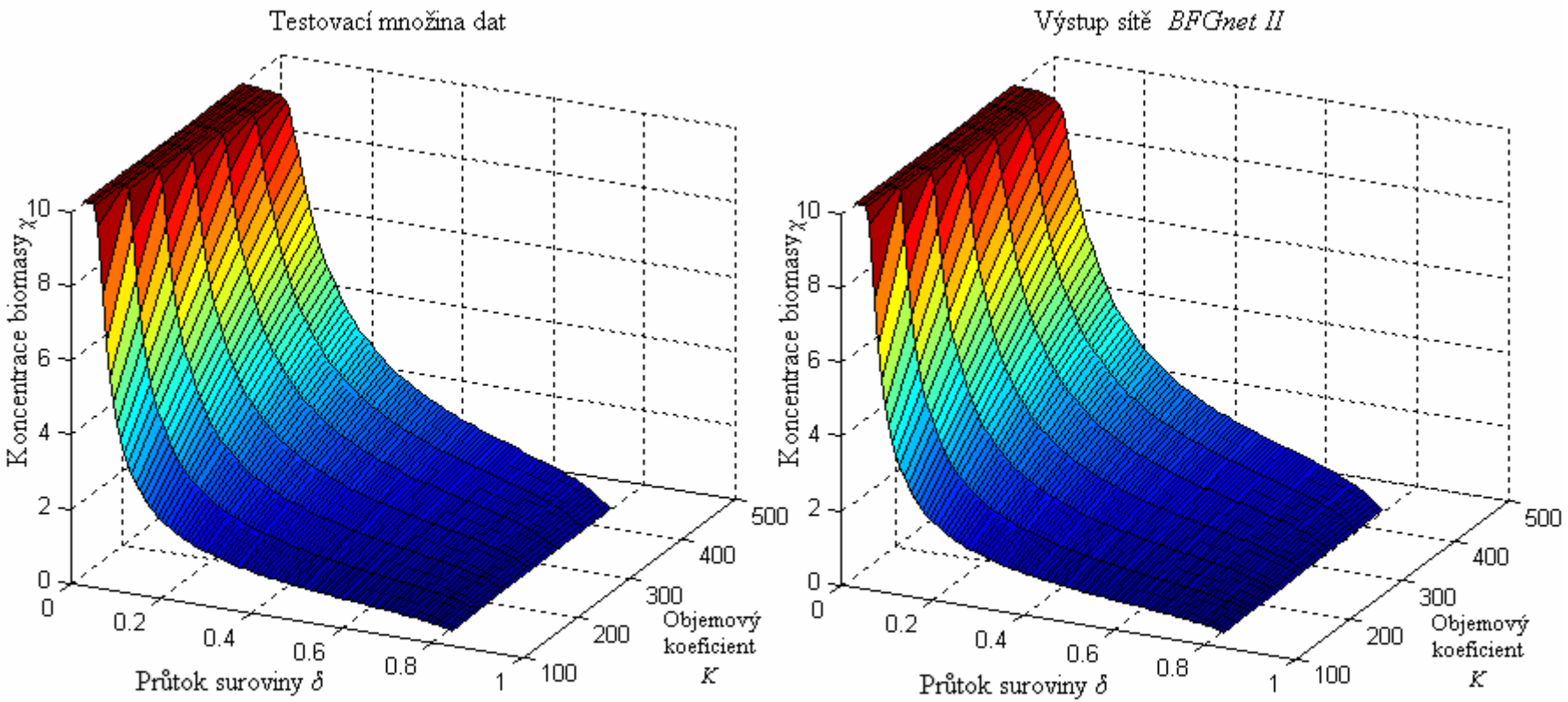
Označení sítě	Topologie	Algoritmus trénování	$E(N)$
BPGnet II	2 – 8 – 8 – 1	BPG algoritmus	6,283
BFGnet II	2 – 10 – 10 – 1	kvazi-Newtonova metoda	$7{,}664.10^{-5}$
LMnet II	2 – 8 – 8 – 1	LM algoritmus	$6{,}091.10^{-8}$

Pro vybrané natrénované umělé neuronové sítě bylo provedeno testování s testovací množinou dat získanou podle odstavce 5.3.1. Zobrazené průběhy jsou vzhledem k charakteru modelu třírozměrné, aby tedy byly grafy aspoň v rámci možností přehledné, byly zobrazeny po dvojicích vedle sebe, vždy žádaný průběh proti výstupu jedné z neuronových sítí. Grafy jsou zobrazeny na obrázcích 5.12 až 5.14. Dále byla na obrázcích 5.15 až 5.17 zobrazena rozložení absolutních chyb na výstupu v závislosti na vstupních proměnných.

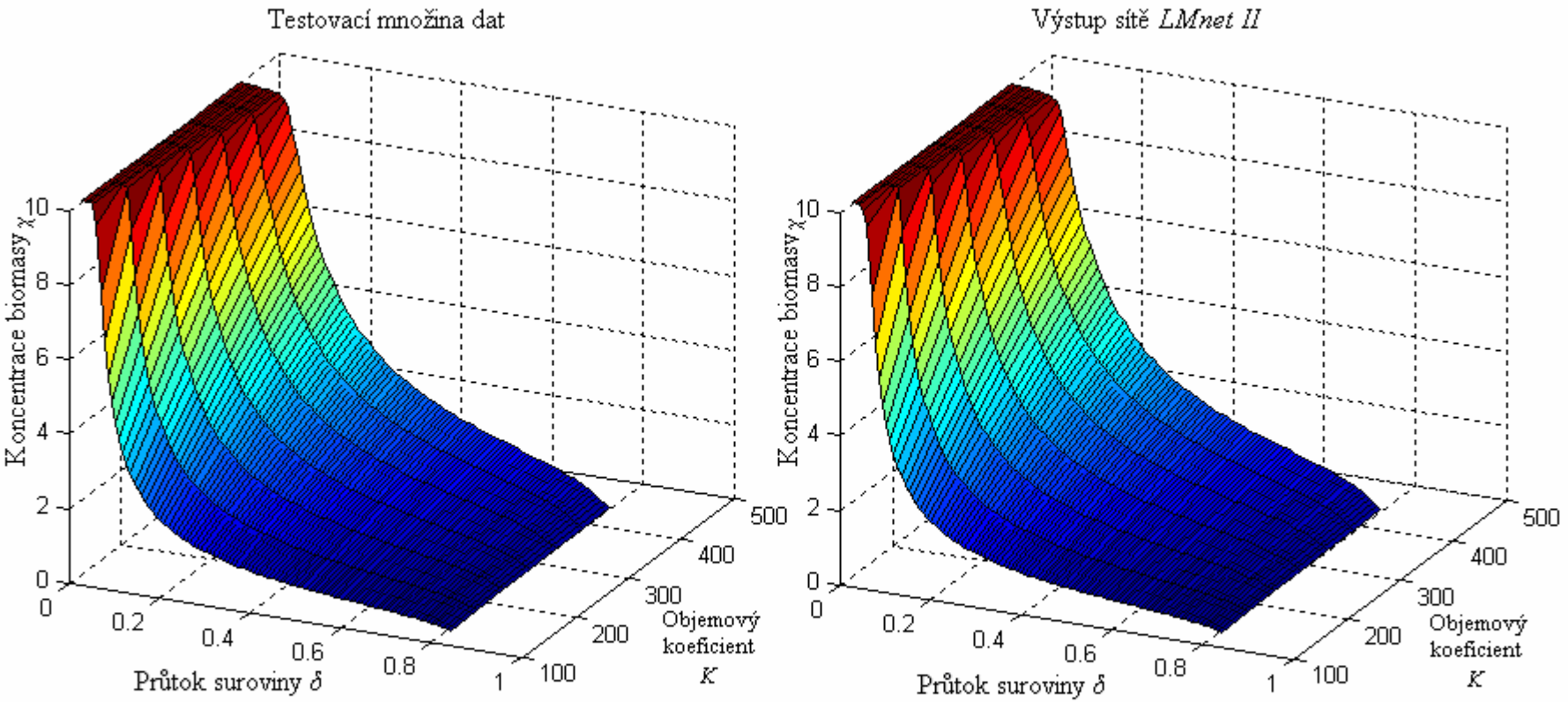
Obrázky 5.12 a 5.15 potvrzují domněnku, že základní *BPG algoritmus* si s tímto MISO modelem nedokázal rozumně poradit. Naproti tomu testování sítí natrénovaných pomocí *kvazi-Newtonovy metody* a *Levenbergova-Marquardtova algoritmu* je možno považovat za úspěšné, přičemž lépe se opět jevil *LM-algoritmus*, který prokázal nejlepší výkon za postačující nižší topologie než *kvazi-Newtonova metoda.* Z rozložení absolutních chyb lze usuzovat, že podobně jako u SISO modelu měly všechny algoritmy největší problém aproximovat strmý pád hodnoty koncentrace biomasy χ při průtoku suroviny kolem hodnoty cca 0,1 (v závislosti na hodnotě objemového koeficientu K).



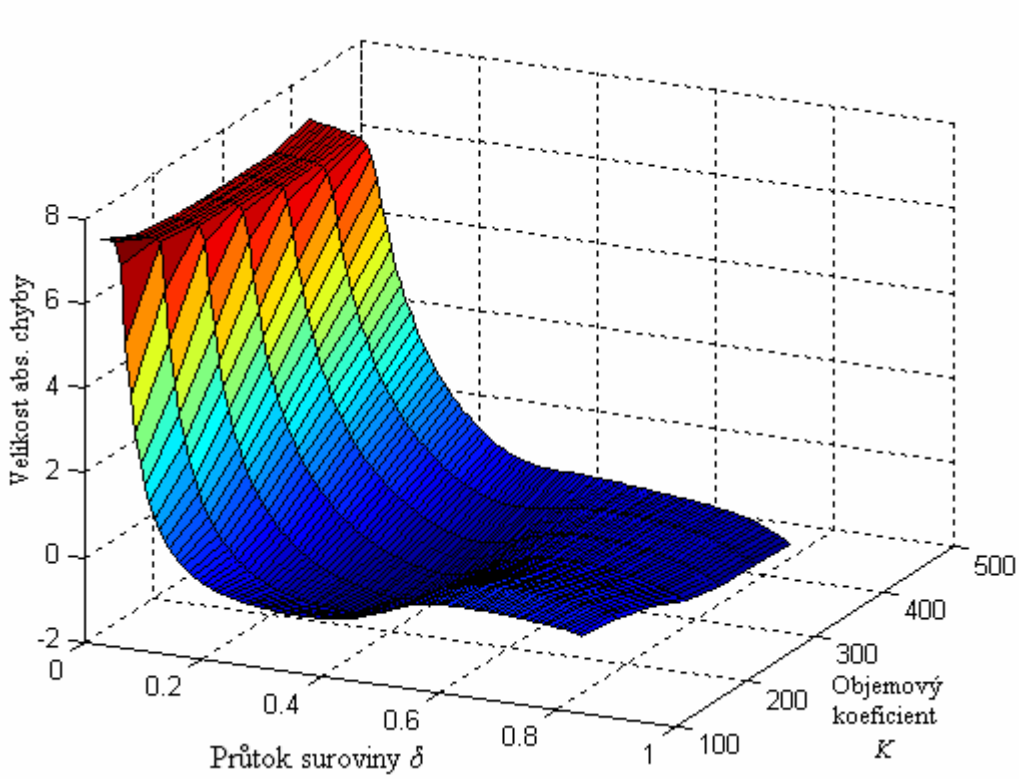
Obr. 5.12 - Porovnání výstupu sítě *BPGnet II* s žádaným výstupem



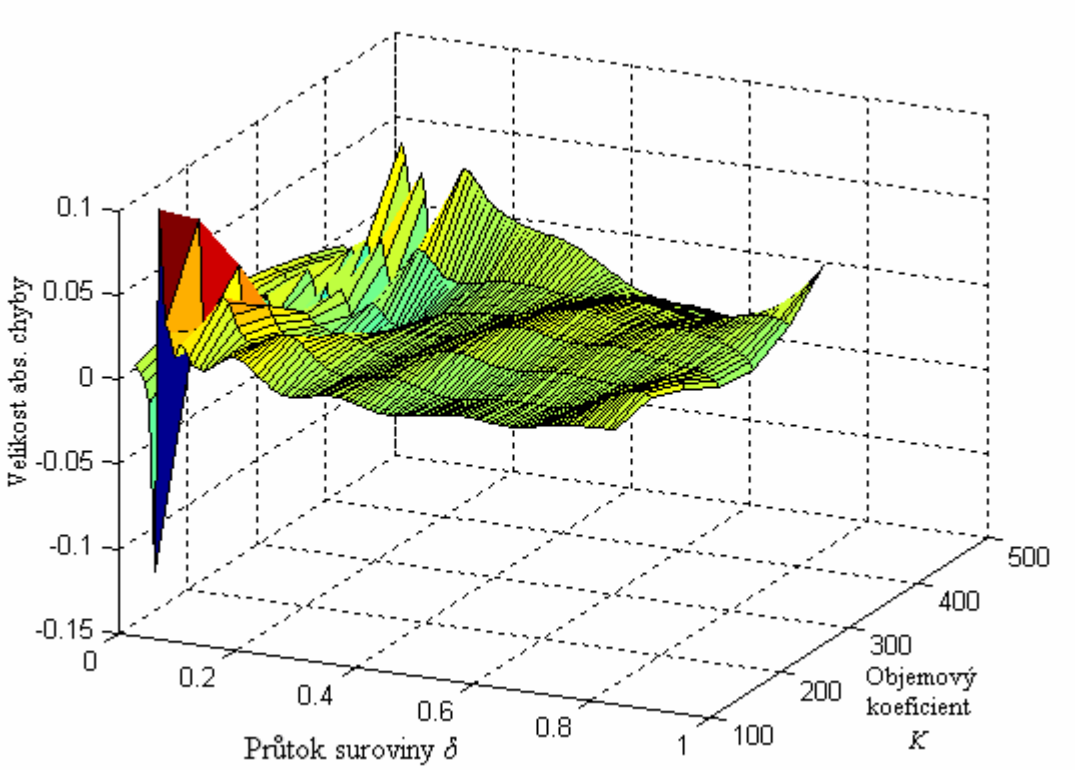
Obr. 5.13 - Porovnání výstupu sítě *BFGnet II* s žádaným výstupem



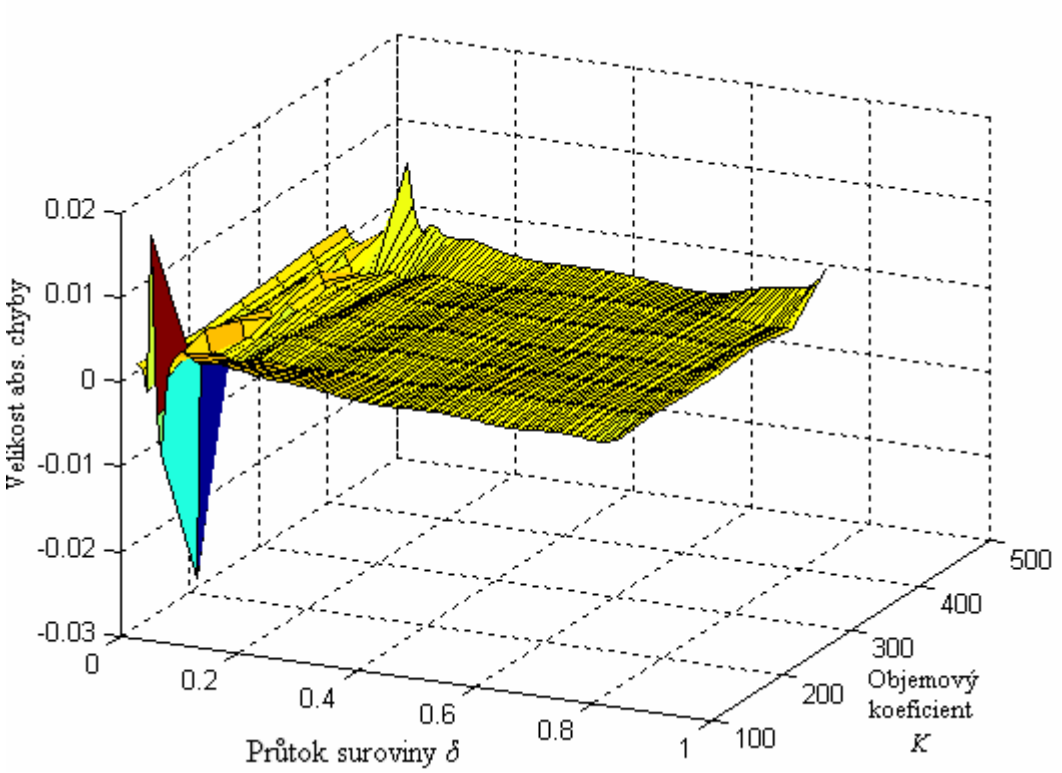
Obr. 5.14 - Porovnání výstupu sítě *LMnet II* s žádaným výstupem



Obr. 5.15 – Rozložení chyby na výstupu sítě *BPGnet II*



Obr. 5.16 – Rozložení chyby na výstupu sítě *BFGnet II*



Obr. 5.17 – Rozložení chyby na výstupu sítě *LMnet II*

5.4 MISO model $\chi = f(\delta, a)$

5.4.1 Generování trénovací a testovací množiny dat

Pro zisk trénovací množiny byla použita opět rovnice (4-26), přičemž parametry byly zvoleny takto:

$$\sigma^0 = 10$$
$$\omega^0 = 10$$
$$K = 100$$

Nezávislá proměnná δ pro trénovací množinu byla volena rovnoměrně na intervalu $(0 \div 0.826446)$ tak, aby byl dosažen vektor o 100 hodnotách, nezávislá proměnná a byla volena na intervalu $(2500 \div 6000)$ s krokem 500. Takto získané vektory byly sestaveny do matice 2×800 tak, že ve sloupcích byly umístěny uspořádané dvojice (δ; a) stylem každá s každou.

Testovací data byla získána analogicky, pouze byla nezávisle proměnná δ volena rovnoměrně na intervalu $(0 \div 0.82)$ a hodnota a byla volena na intervalu $(2600 \div 6100)$, čímž byla zajištěna rozdílnost trénovací a testovací množiny dat.

5.4.2 Hledání optimální topologie pro základní BPG algoritmus

Hledání optimální topologie proběhlo analogicky k odstavci 5.2.2 ve dvou krocích:

a) Pro předem daný koeficient rychlosti učení byly trénovány sítě o různých topologiích a pro další užití byla vybrána síť s nejmenším kriteriem střední kvadratické chyby.
b) Neuronová síť vybraná předchozím bodem byla trénována pro různé koeficienty rychlosti učení, zkoumala se rychlost konvergence.

Vzhledem ke stejnému postupu jako v odstavci 5.2.2 budou prezentovány pouze výsledné hodnoty a průběhy.

Ad a)

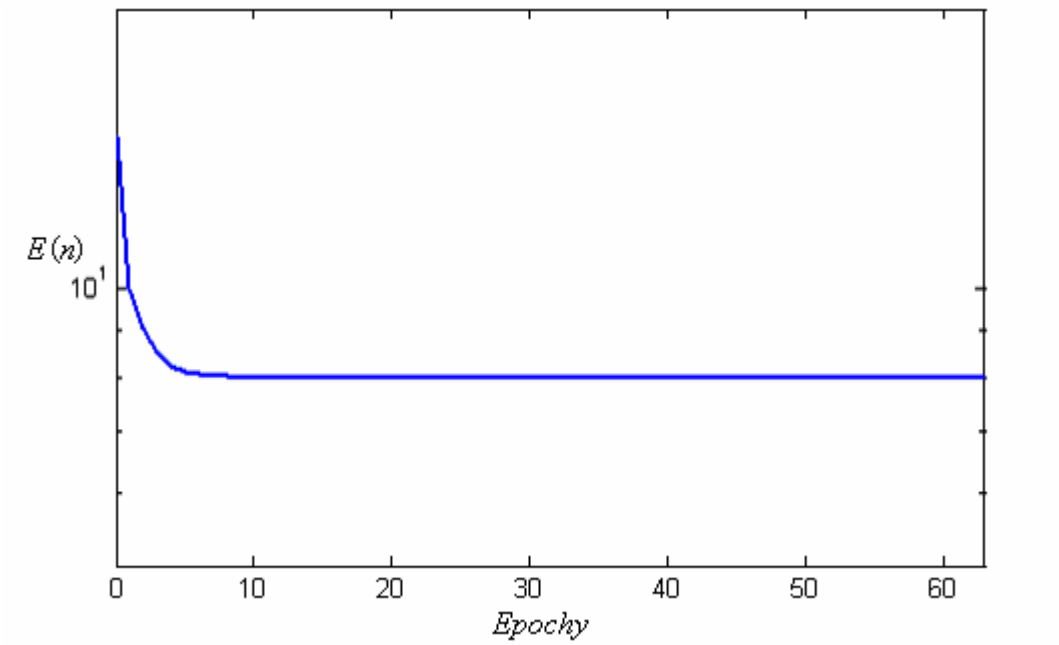
$\alpha = 0.01$

Tab. 5.11 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-20-20-1
$E(N)$	7,997	7,997	7,997	7,997	7,997	7,997

Je patrné, že BPG algoritmus tak, jak jej nabízí *Neural Network Toolbox*, si s ani s tímto MISO modelem nedokázal poradit.

Průběhy kriteriálních funkcí byly opět téměř totožné, k lokálnímu minimu konvergovaly již po několika desítkách iterací a ani po mnoha nových inicializacích se z vlivu toho lokálního minima nedokázaly vymanit. Průběh jednoho zástupce je znázorněn na obrázku 5.18.



Obr. 5.18 – Shodný průběh kriteriálních funkcí

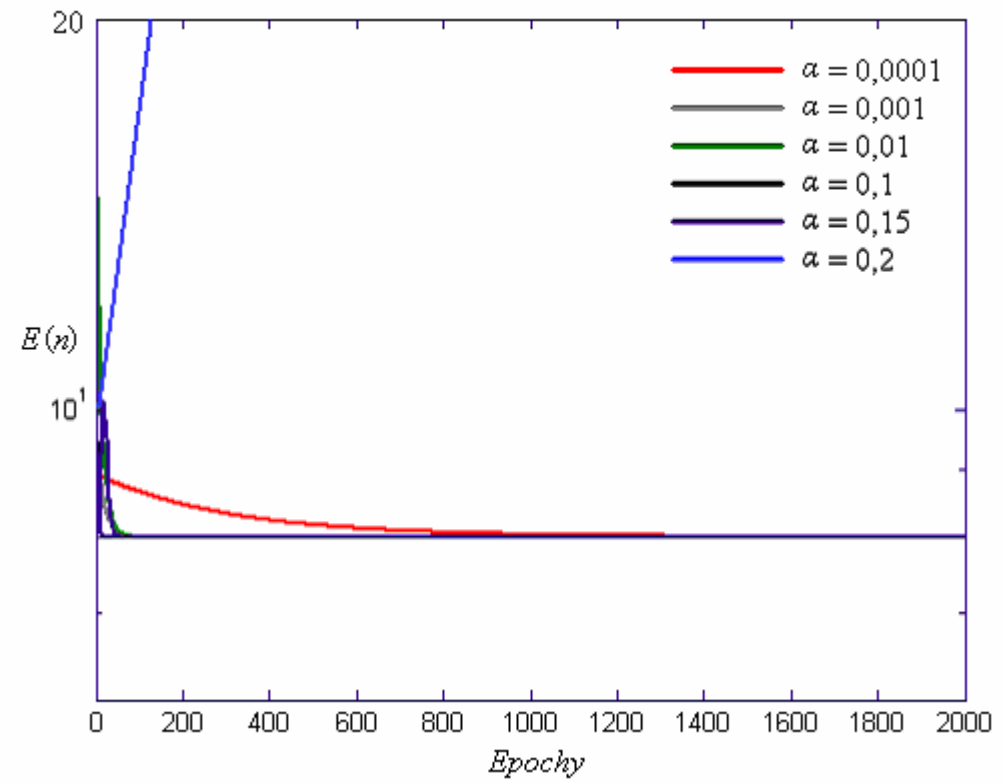
Pro další výpočet byla vybrána topologie 2 – 8 – 8 – 1.

Ad b)

Neuronová síť 2 – 8 – 8 – 1 byla *BPG algoritmem* trénována pro různé koeficienty rychlosti učení, výsledné hodnoty kriteria střední kvadratické chyby jsou shrnuty v tabulce 5.12 a průběhy kriteriální funkce jsou vyneseny do grafu na obrázku 5.19.

Tab. 5.12 – Hodnoty $E(N)$ pro hodnoty koeficientu rychlosti učení

α	0,0001	0,001	0,10	0,15	0,20
$E(N)$	7,997	7,997	7,997	7,997	∞



Obr. 5.19 – Průběhy kriteriálních funkcí pro různé koeficienty rychlosti učení

Jak je patrno z tabulky 5.12 a obrázku 5.19, ani změny koeficientu rychlosti učení nepřinesly zlepšení výsledků. Jediné rozdíly mezi jednotlivými průběhy kriteriálních funkcí (kromě nestabilní krajní hodnoty $\alpha = 0.2$) jsou v rychlosti dosažení lokálního minima. V konečném porovnání bude takto natrénovaná síť označena pojmem *BPGnet III*.

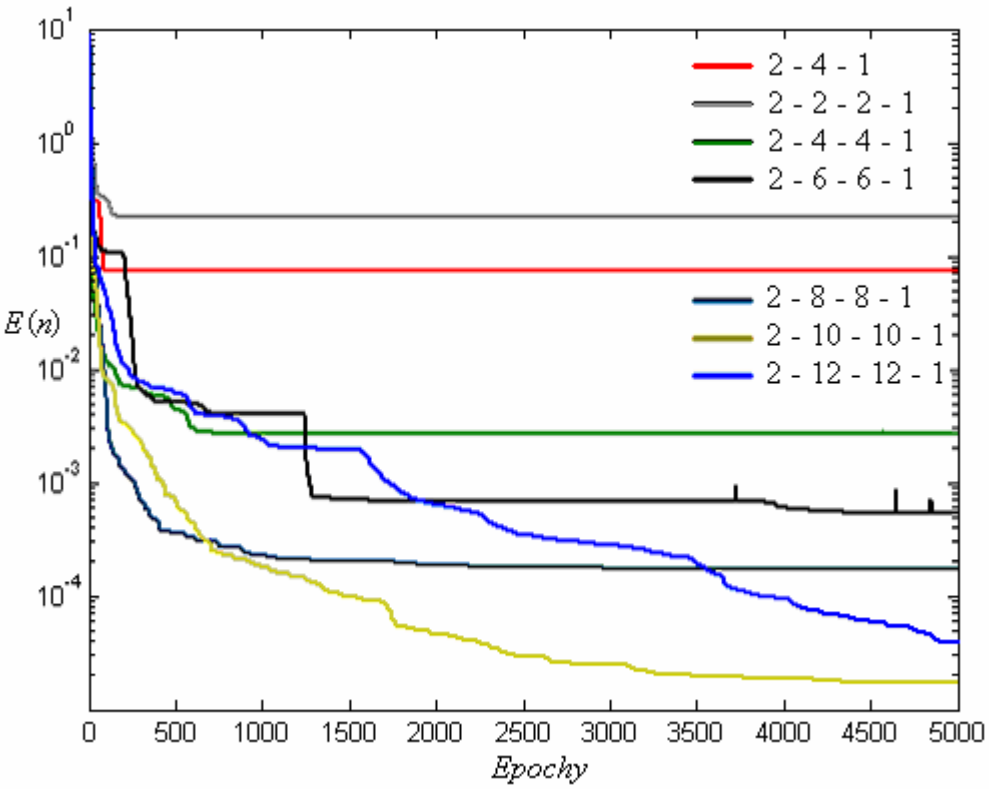
5.4.3 Hledání optimální topologie pomocí kvazi-Newtonovy metody

Hledání optimální topologie proběhlo analogicky k odstavci 5.2.3. Výsledné hodnoty kriteria střední kvadratické chyby jsou uvedeny v tabulce 5.13.

Tab. 5.13 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-10-10-1	2-12-12-1
$E(N)$	0,07264	0,2179	0,002703	$5{,}421.10^{-4}$	$1{,}733.10^{-4}$	$1{,}712.10^{-5}$	$3{,}868.10^{-5}$

Průběhy jednotlivých kriteriálních funkcí jsou uvedeny na obrázku 5.20.



Obr. 5.20 – Průběhy kriteriálních funkcí

Jak je patrné z tabulky 5.13 a obrázku 5.20, již od topologie 2-6-6-1 poskytuje pro tento MISO model *kvazi-Newtonova metoda* poměrně přijatelné výsledky. Nejnižší hodnotu kvadratického kriteria poskytla síť o topologii 2-10-10-1, zatímco topologie 2-12-12-1 již zlepšení nepřinesla, proto bude do konečného srovnání použita síť o topologii 2-10-10-1. Označena v dalším textu bude názvem *BFGnet III.*

5.4.4 Hledání optimální topologie pomocí Levenbergovy-Marquardtovy metody

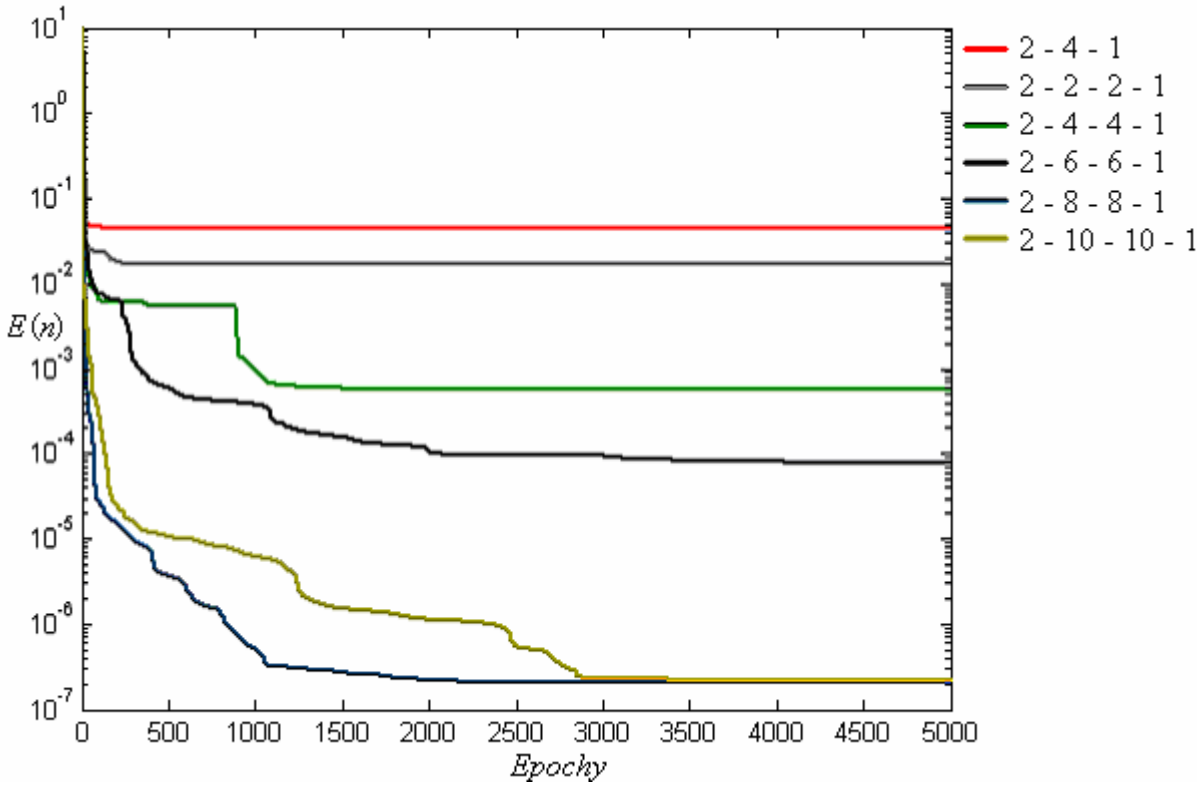
Hledání optimální topologie proběhlo analogicky k odstavci 5.2.4. Výsledné hodnoty kriteria střední kvadratické chyby jsou uvedeny v tabulce 5.14.

Tab. 5.14 – Hodnoty $E(N)$ pro jednotlivé topologie

Topologie	2-4-1	2-2-2-1	2-4-4-1	2-6-6-1	2-8-8-1	2-10-10- 1
$E(N)$	0,04567	0,01652	$5{,}550.10^{-4}$	$7{,}856.10^{-5}$	$2{,}026.10^{-7}$	$2{,}1464.10^{-7}$

Průběhy kriteriálních funkcí jsou znázorněny na obrázku 5.21.

Tabulka 5.14 i obrázek 5.21 ukazují, že *LM algoritmus* si poradil i s tímto MISO modelem. Se složitější topologií poskytuje vždy lepší výsledky a to až do topologie 2-8-8-1. Následující neuronová síť již nevykazuje zlepšení, proto bude do zhodnocení vybrána natrénovaná umělá neuronová síť s topologií 2-8-8-1, která ponese v dalším textu označení *LMnet III.*



Obr. 5.21 – Průběhy kriteriálních funkcí

5.4.5 Porovnání a zhodnocení jednotlivých algoritmů učení pro MISO model $\chi = f(\delta, K)$

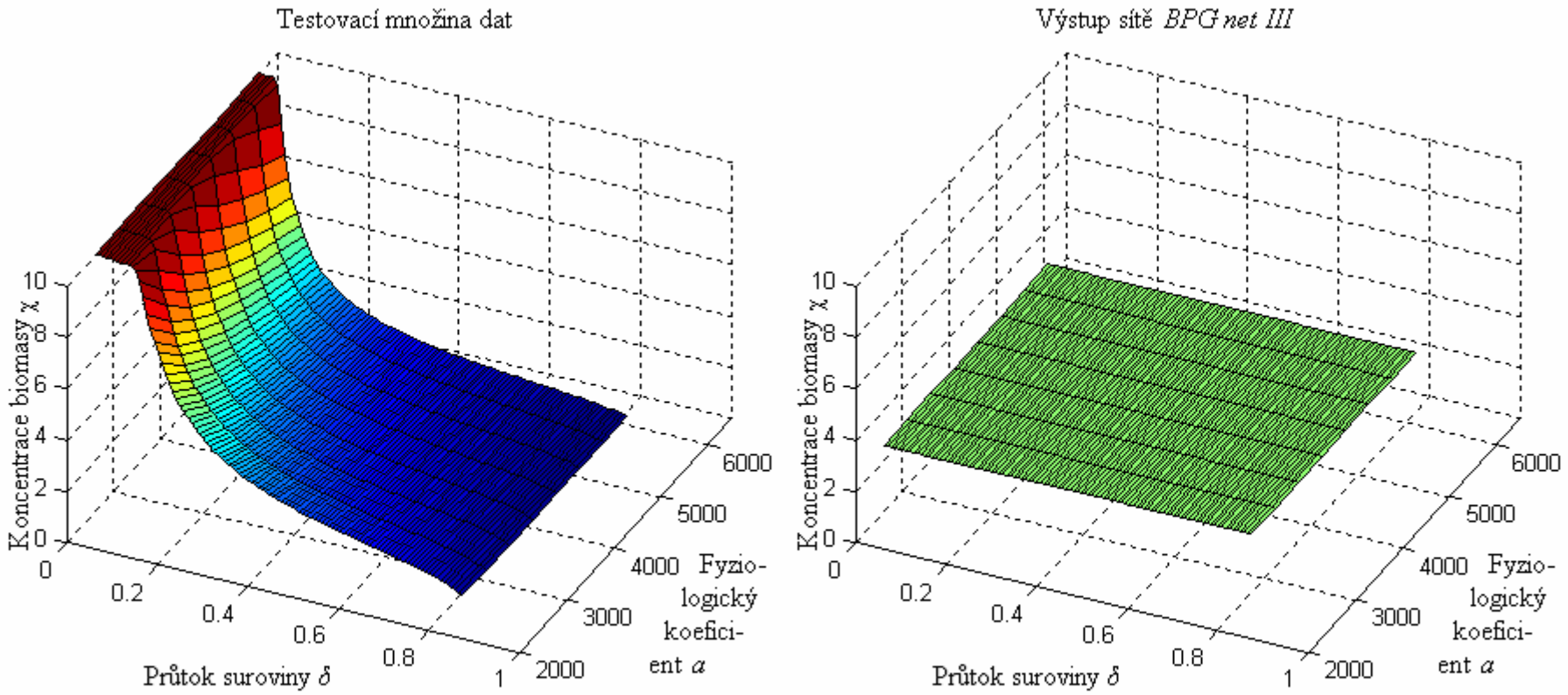
Vybrané sítě jsou seřazeny v tabulce 5.15.

Tab. 5.15 – Seznam sítí pro zhodnocení

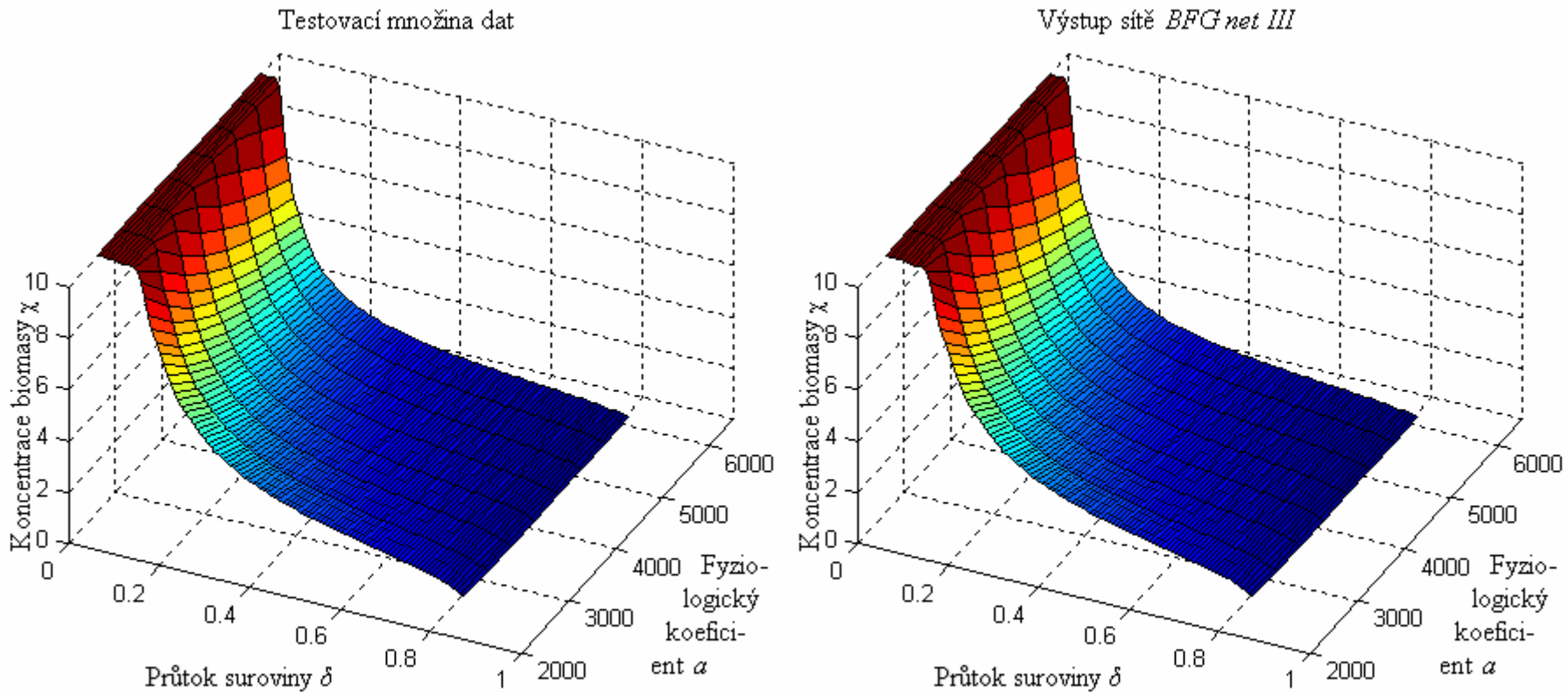
Označení sítě	Topologie	Algoritmus trénování	$E(N)$
BPGnet III	2 – 8 – 8 – 1	BPG algoritmus	7,997
BFGnet III	2 – 10 – 10 – 1	kvazi-Newtonova metoda	$1{,}712.10^{-5}$
LMnet III	2 – 8 – 8 – 1	LM algoritmus	$2{,}026.10^{-7}$

Pro tyto vybrané natrénované umělé neuronové sítě bylo provedeno testování s testovací množinou dat získanou podle odstavce 5.4.1. Výsledné grafy jsou opět třírozměrné, zobrazení tedy proběhlo analogicky podle odstavce 5.3.5. Porovnání požadovaných výstupu se skutečnými je zobrazeno na obrázcích 5.22 až 5.24, zobrazení chyb na výstupu pak na obrázcích 5.25 až 5.27.

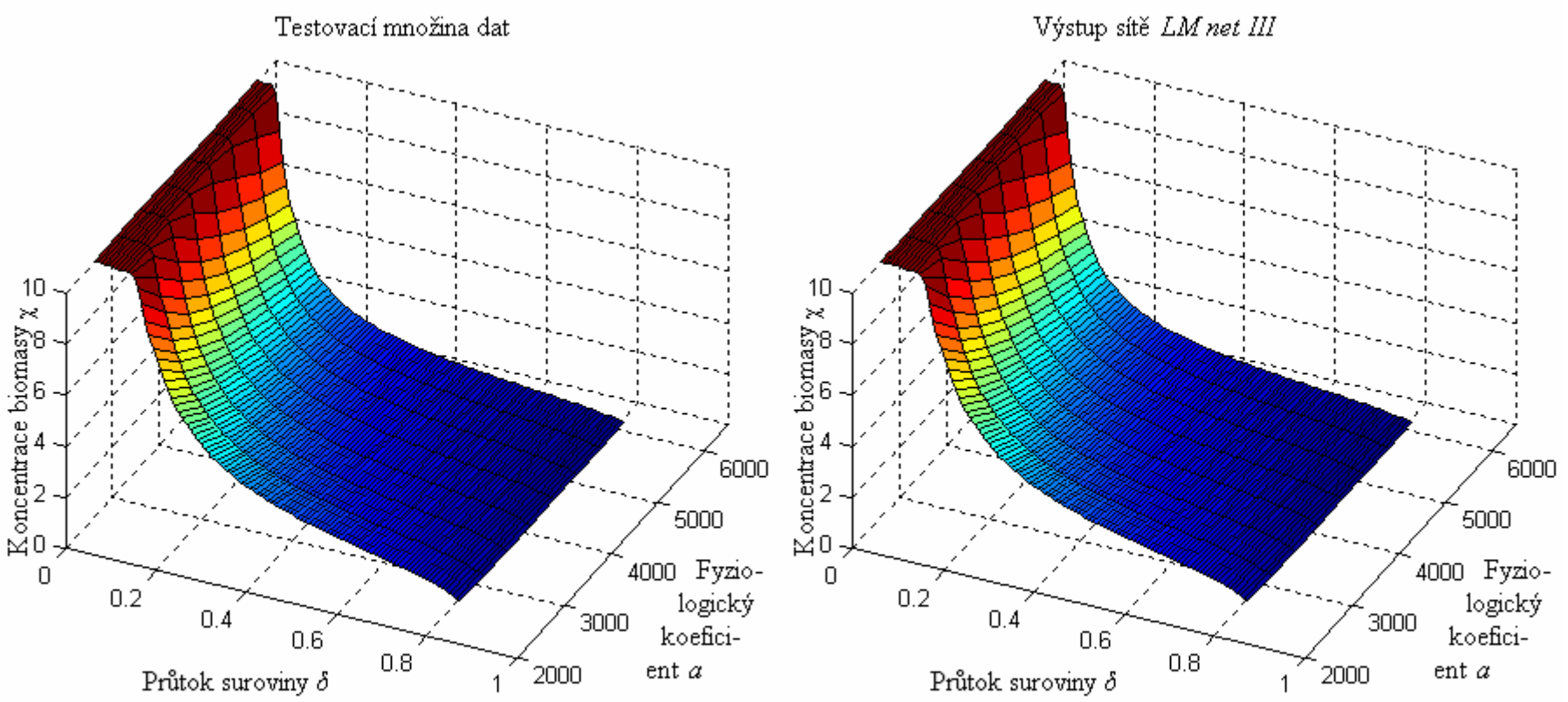
Obrázky 5.22 s 5.25 opět dokazují, že základní *BPG algoritmus* tak, jak jej poskytuje funkce *trainGD Neural Network Toolboxu*, si se zkoumaným MISO modelem naprosto nedokázal poradit, zatímco *kvazi-Newtonova metoda* i *Levenbergův-Marquardtův algoritmus* poskytují výsledky, jejichž rozdíl je vizuálně nerozeznatelný (obrázky 5.23 a 5.24). Z rozložení chyb na obrázcích 5.26 s 5.27 lze říci, že *LM-algoritmus* i v tomto třetím případě poskytl nejpřesnější výsledky a opět se potvrdil předpoklad, že největší chyby tyto neuronové sítě poskytují kolem strmého poklesu hodnoty koncentrace biomasy x při průtoku suroviny kolem hodnoty cca 0,1.



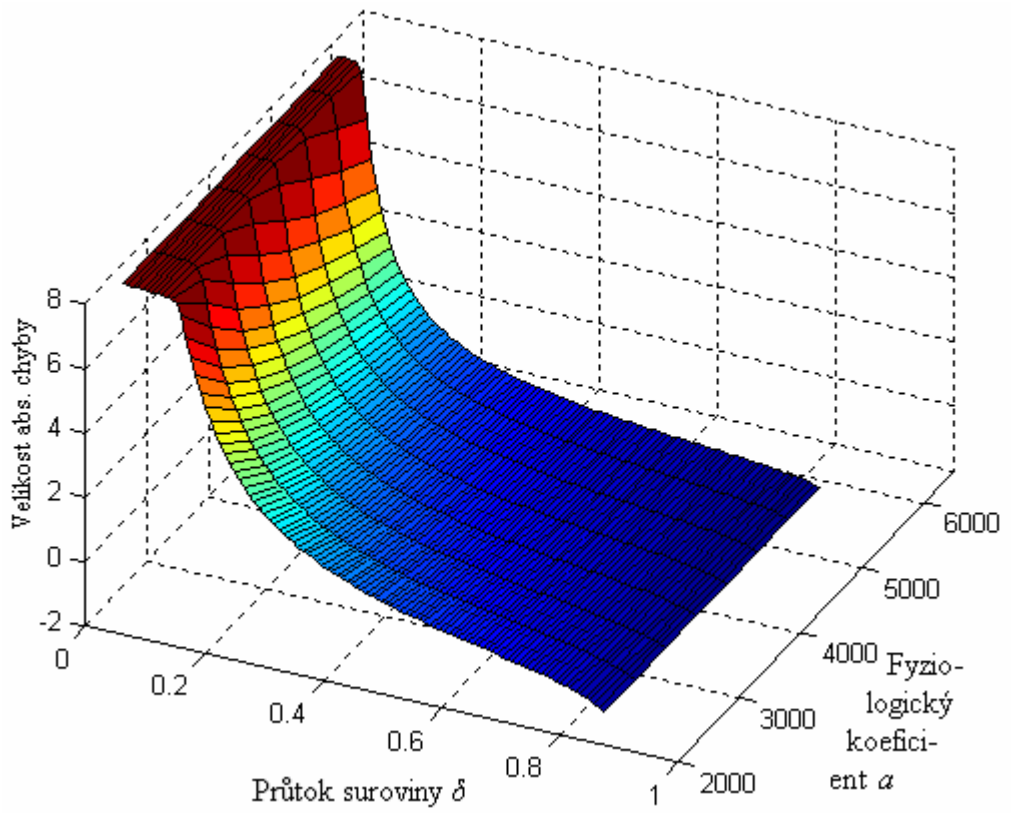
Obr. 5.22 - Porovnání výstupu sítě *BPG net III* s žádaným výstupem



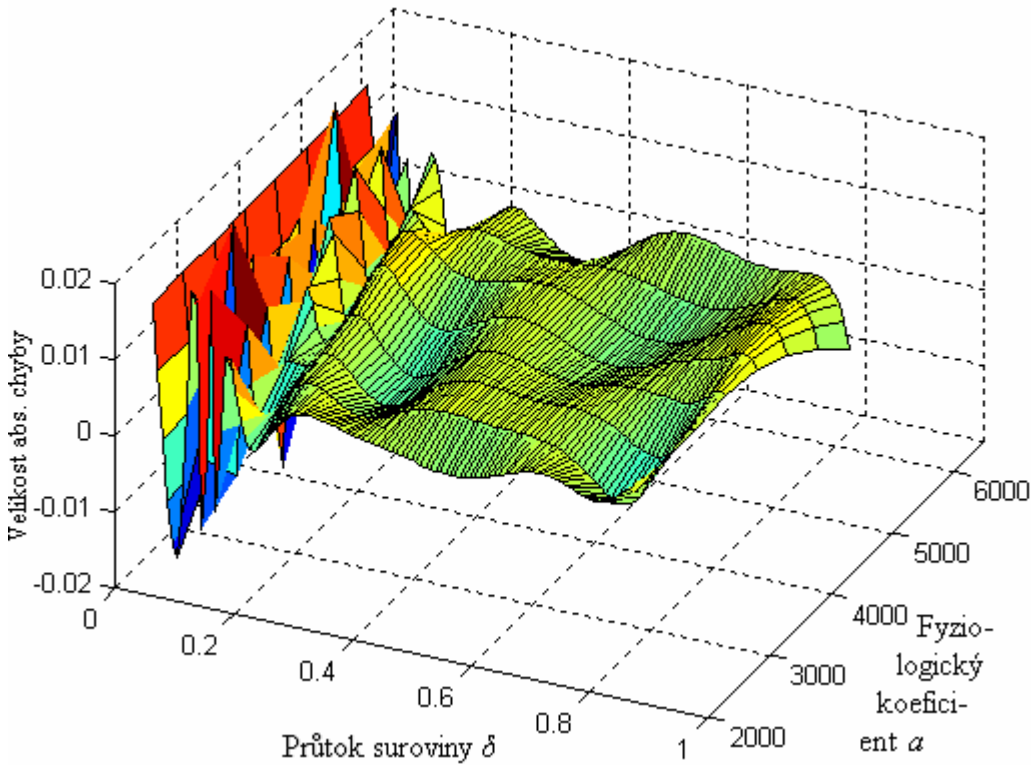
Obr. 5.23 - Porovnání výstupu sítě *BFGnet III* s žádaným výstupem



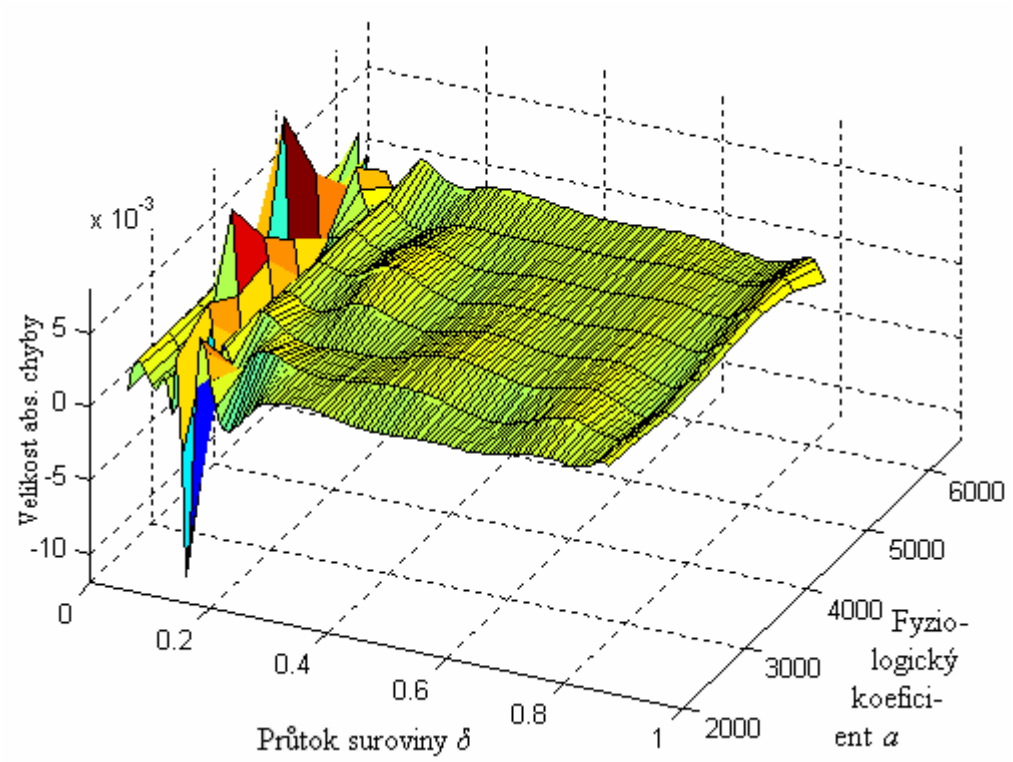
Obr. 5.24 - Porovnání výstupu sítě *LM net III* s žádaným výstupem



Obr. 5.25 - Rozložení chyby na výstupu sítě *BPGnet III*



Obr. 5.26 - Rozložení chyby na výstupu sítě *BFGnet III*



Obr. 5.27 - Rozložení chyby na výstupu sítě *LMnet III*

5.5 Zhodnocení tvorby statického modelu bioreakce

Byly provedeny experimenty se třemi modely založenými na statické charakteristice bioreakce popsané vztahem (4-26). V každém experimentu byla provedena syntéza umělé neuronové sítě pomocí tří algoritmů učení (*BPG algoritmus, kvazi-Newtonova metoda a LM algoritmus*). Výsledné hodnoty byly diskutovány vždy na konci úseků věnovaných jednotlivým modelům, je však možno konstatovat, že kvalitativní pořadí zkoumaných algoritmů učení bylo vždy shodné a dopadlo podle očekávání. Výpočetně nejjednodušší *BPG algoritmus* poskytoval nejméně přesné výsledky, zatímco výpočetně nejnáročnější *LM algoritmus* poskytoval výsledky nejpřesnější. To ovšem neznamená, že *Levenbergův-Marquardtův algoritmus* je nejvýhodnější použít v každém případě. Má několik nevýhod, z nichž ta nejviditelnější je časová náročnost výpočtu. V dnešní době relativně rychlých počítačů se může zdát tato nevýhoda překonaná, ovšem stále ještě může vyplynout na povrch při použití v systémech s požadavkem odezvy v reálném čase. Naproti tomu učení *BPG algoritmu*, ač vyžadovalo více epoch tréninku, časově proběhlo daleko rychleji. Bohužel však v případě MISO modelů *BPG algoritmus* naprosto zklamal. Je možné, že například jinou volbou trénovací množiny by bylo dosaženo lepších výsledků, ovšem za daných podmínek stejných pro všechny algoritmy *BPG algoritmus* neuspěl, zatímco *kvazi-Newtonova metoda* i *LM algoritmus* poskytly přijatelné výsledky.

Další zajímavou vlastností algoritmů učení umělých neuronových sítí, která byla během experimentů pozorována, byla schopnost překonat lokální minima. jak je vidět z hodnot v tabulkách 5.6 či 5.11, zvláště při trénování neuronových sítí použitých pro aproximaci MISO modelů se vyskytovala velmi nevýhodná lokální minima, která se vyznačovala vysokými hodnotami kriteriálních funkcí (8,011 pro MISO model $\chi = f(\delta, K)$ a 7,997 pro MISO model $\chi = f(\delta, a)$). Přesto k těmto lokálním minimům algoritmy učení velmi často konvergovaly. U *BPG algoritmu* nepomohla ani opakovaná náhodná inicializace vah spojení a prahů na počátku trénování. *Kvazi-Newtonova metoda* k těmto minimům

konvergovala také poměrně často, ovšem při opakovaném experimentu s nově inicializovanými vahami spojení byla tato lokální minima překonána. Naproti tomu *LM algoritmus* k těmto nevýhodným lokálním minimům konvergoval jen sporadicky a pouze při jednoduchých topologiích.

Závěrem lze tedy říci, že nejpříznivější výsledky při učení umělých neuronových sítí na zkoumaných tréninkových množinách poskytuje *Levenbergův-Marquardtův algoritmus.* V případě použití takovým způsobem, že se neuronová natrénuje pouze jednou na začátku a pak se již nebudou váhy spojení upravovat, je *LM algoritmus* nejvhodnější. V případě opakovaného trénování během pracovního procesu modelu již není doporučení *LM algoritmu* tak jednoznačné a je třeba vzít v úvahu přesnost, jaká je požadována.

6 Tvorba dynamického modelu bioreaktoru pomocí umělé neuronové sítě

6.1 Teoretický popis tvorby dynamického modelu nelineární soustavy pomocí neuronové sítě

6.1.1 Úvod

Existuje několik možností, jak modelovat dynamický systém pomocí neuronové sítě. Asi nejpřirozenější postup spočívá v převedení modelu do diskrétního tvaru volbou vhodného intervalu vzorkování a dále postupovat analogicky metodám experimentální identifikace (např. [Drábek, 1987]), pouze místo parametrů diferenční rovnice se optimalizují hodnoty vah a prahů v neuronové síti. Tento postup klade požadavky na počet vstupů a výstupů umělé neuronové sítě a tvar trénovací a testovací množiny, ovšem samotný typ topologie neuronové sítě i algoritmus trénování zůstanou shodné s topologiemi a algoritmy použitými při tvorbě statického modelu v předchozích odstavcích.

6.1.2 Struktura nelineárního dynamického modelu

Nechť je třeba vytvořit model obecně MIMO soustavy s vektorem vstupů $\boldsymbol{u}_R$ a vektorem výstupů $\boldsymbol{y}_S$ (obr. 6.1).

Obr. 6.1 – Schéma MIMO soustavy

Pomocí zvoleného intervalu vzorkování lze chování soustavy popsat soustavou nelineárních diferenčních rovnic (6-1).

$$\begin{aligned}
y_{S1}(k) &= \varphi_1\big[y_{S1}(k-1), y_{S1}(k-2), \cdots, y_{S1}(k-n_1), y_{S2}(k-1), \cdots, y_{S2}(k-n_2), \cdots, y_{Sr}(k-1), \cdots, z_{Sr}(k-n_r),\\
&\quad u_1(k-1), u_1(k-2), \cdots, u_1(k-m_1), u_2(k-1), \cdots, u_2(k-m_2), \cdots, u_s(k-1), \cdots, u_s(k-m_s)\big]\\
y_{S2}(k) &= \varphi_2\big[y_{S1}(k-1), y_{S1}(k-2), \cdots, y_{S1}(k-n_1), y_{S2}(k-1), \cdots, y_{S2}(k-n_2), \cdots, y_{Sr}(k-1), \cdots, z_{Sr}(k-n_r),\\
&\quad u_1(k-1), u_1(k-2), \cdots, u_1(k-m_1), u_2(k-1), \cdots, u_2(k-m_2), \cdots, u_s(k-1), \cdots, u_s(k-m_s)\big]\\
&\vdots\\
y_{Sr}(k) &= \varphi_r\big[y_{S1}(k-1), y_{S1}(k-2), \cdots, y_{S1}(k-n_1), y_{S2}(k-1), \cdots, y_{S2}(k-n_2), \cdots, y_{Sr}(k-1), \cdots, z_{Sr}(k-n_r),\\
&\quad u_1(k-1), u_1(k-2), \cdots, u_1(k-m_1), u_2(k-1), \cdots, u_2(k-m_2), \cdots, u_s(k-1), \cdots, u_s(k-m_s)\big]
\end{aligned} \tag{6-1}$$

Přičemž platí:

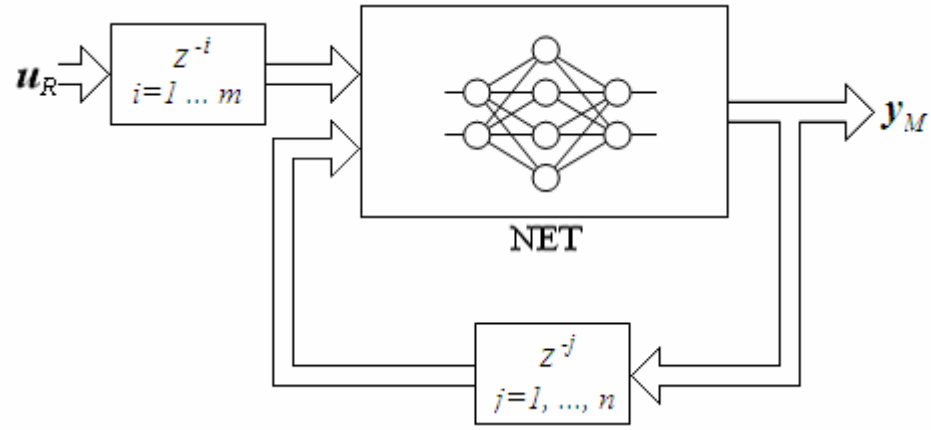
$$\boldsymbol{u}_R = [u_1, u_2, \cdots, u_s]^{\mathrm{T}}$$
$$\boldsymbol{y}_S = [y_{S1}, y_{S2}, \cdots, y_{Sr}]^{\mathrm{T}}$$

Je zřejmé, že při identifikaci je klíčové určit vektor nelineárních funkcí $\boldsymbol{\varphi} = [\varphi_1, \varphi_2, \cdots, \varphi_r]^{\mathrm{T}}$. Dopředná neuronová síť se jeví jako vhodná varianta aproximace celého vektoru funkcí $\boldsymbol{\varphi}$.

6.1.3 Volba struktury neuronové sítě a tvaru trénovací množiny

Volba topologie neuronové sítě (počet vstupů, výstupů, počet neuronů a vrstev, volba aktivačních funkcí, …) je důležitou oblastí celé identifikace, ale probíhá poměrně rutinně, neboť počet vstupů a výstupů je dán identifikovanou soustavou (u neuronové sítě budou vstupy rozšířeny o vektor předchozích výstupů, aby odpovídaly vztahům (6-1)) a volba počtu neuronů a jejich uspořádání do vrstev probíhá shodně s postupy uvedenými při tvorbě statického modelu.

Požadovaná neuronová síť je schematicky znázorněna na obrázku 6.2.



Obr. 6.2 – Schéma dynamického modelu soustavy pomocí neuronové sítě

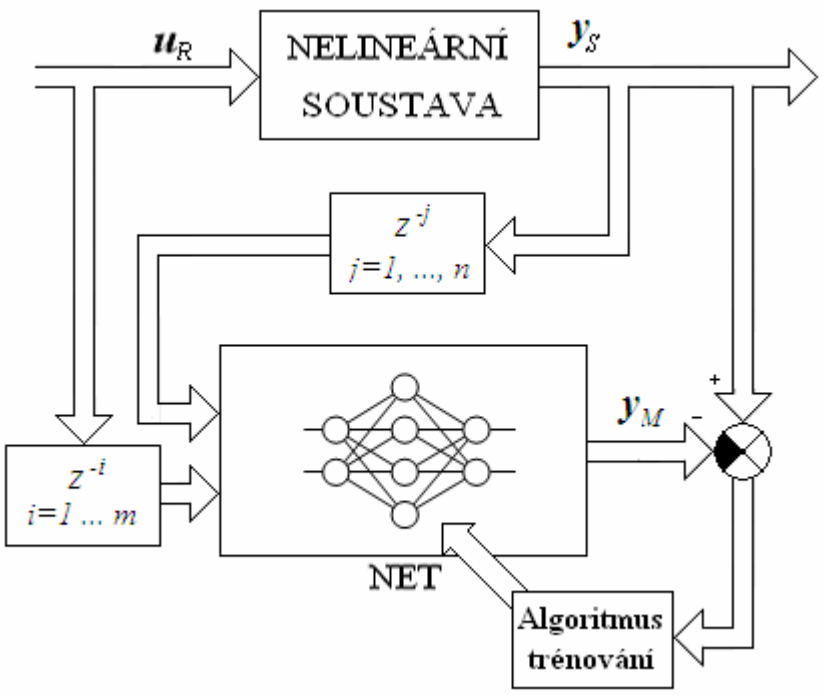
Snahou je, aby výstup neuronové sítě $\boldsymbol{y}_M$ byl na vstupní signál $\boldsymbol{u}_R$ co nejbližší výstupu $\boldsymbol{y}_S$ původní nelineární soustavy. Aby toho bylo docíleno, musí se neuronová síť trénovat pomocí vhodně seřazených hodnot trénovací množiny. Struktura trénovací množiny je uvedena v tabulce 6.1, schéma trénování pak na obrázku 6.3.

Samotný proces trénování a testování je možno provádět řadou možností, z nichž některé jsou popsány v odstavci 3.

Podrobnější popis tvorby dynamického modelu společně s mnoha modifikacemi je možno nalézt v [Haykin, 1999].

Tab. 6.1 – Trénovací množina

	k	1	2	…	N
Vstupní vzory	$u_1(k-1)$	$u_1(0)$	$u_1(1)$	…	$u_1(N-1)$
	$u_1(k-2)$	$u_1(-1)$	$u_1(0)$	…	$u_1(N-2)$
	⋮	⋮	⋮	…	⋮
	$u_1(k-m_1)$	$u_1(1-m_1)$	$u_1(2-m_1)$	…	$u_1(N-m_1)$
	$u_2(k-1)$	$u_2(0)$	$u_2(1)$	…	$u_2(N-1)$
	$u_2(k-2)$	$u_2(-1)$	$u_2(0)$	…	$u_2(N-2)$
	⋮	⋮	⋮	…	⋮
	$u_2(k-m_2)$	$u_2(1-m_2)$	$u_2(2-m_2)$	…	$u_2(N-m_2)$
	⋮	⋮	⋮	…	⋮
	$u_s(k-1)$	$u_s(0)$	$u_s(1)$	…	$u_s(N-1)$
	$u_s(k-2)$	$u_s(-1)$	$u_s(0)$	…	$u_s(N-2)$
	⋮	⋮	⋮	…	⋮
	$u_s(k-m_s)$	$u_s(1-m_s)$	$u_s(2-m_s)$	…	$u_s(N-m_s)$
	$y_{S1}(k-1)$	$y_{S1}(0)$	$y_{S1}(1)$	…	$y_{S1}(N-1)$
	$y_{S1}(k-2)$	$y_{S1}(-1)$	$y_{S1}(0)$	…	$y_{S1}(N-2)$
	⋮	⋮	⋮	…	⋮
	$y_{S1}(k-n_1)$	$y_{S1}(1-n_1)$	$y_{S1}(2-n_1)$	…	$y_{S1}(N-n_1)$
	$y_{S2}(k-1)$	$y_{S2}(0)$	$y_{S2}(1)$	…	$y_{S2}(N-1)$
	$y_{S2}(k-2)$	$y_{S2}(-1)$	$y_{S2}(0)$	…	$y_{S2}(N-2)$
	⋮	⋮	⋮	…	⋮
	$y_{S2}(k-n_2)$	$y_{S2}(1-n_2)$	$y_{S2}(2-n_2)$	…	$y_{S2}(N-n_2)$
	⋮	⋮	⋮	…	⋮
	$y_{Sr}(k-1)$	$y_{Sr}(0)$	$y_{Sr}(1)$	…	$y_{Sr}(N-1)$
	$y_{Sr}(k-2)$	$y_{Sr}(-1)$	$y_{Sr}(0)$	…	$y_{Sr}(N-2)$
	⋮	⋮	⋮	…	⋮
	$y_{Sr}(k-n_r)$	$y_{Sr}(1-n_r)$	$y_{Sr}(2-n_r)$	…	$y_{Sr}(N-n_r)$
Výstupy	$y_{S1}(k)$	$y_{S1}(1)$	$y_{S1}(2)$	…	$y_{S1}(N)$
	$y_{S2}(k)$	$y_{S2}(1)$	$y_{S2}(2)$	…	$y_{S2}(N)$
	$y_{S3}(k)$	$y_{S3}(1)$	$y_{S3}(2)$	…	$y_{S3}(N)$
	$y_{S4}(k)$	$y_{S4}(1)$	$y_{S4}(2)$	…	$y_{S4}(N)$
	⋮	⋮	⋮	…	⋮
	$y_{Sr}(k)$	$y_{Sr}(1)$	$y_{Sr}(2)$	…	$y_{Sr}(N)$



Obr. 6.3 – Schéma trénování neuronové sítě jako dynamického modelu

6.2 Identifikace dynamického modelu bioreaktoru

6.2.1 Požadavky na dynamický neuronový model bioreaktoru

Dynamický model bioreaktoru by měl v rozumné míře simulovat chování skutečného bioreaktoru, v tomto případě spojitého matematicko-fyzikálního modelu. Model bude vytvořen postupem teoreticky popsaným v odstavci 6.1**,** čímž bude obdržen diskrétní model s pevnou hodnotou intervalu vzorkování. Za vstup do modelu bude považován pouze průtok suroviny δ, výstupem bude trojice hodnot koncentrace biomasy χ, koncentrace substrátu σ a koncentrace kyslíku ω. Ostatní relevantní hodnoty (objemový koeficient *K*, fyziologický koeficient *a*, počáteční koncentrace substrátu σ^0, rovnovážná koncentrace kyslíku ω^0 a počáteční koncentrace biomasy χ^0) budou pokládány za podmínky experimentu a pro daný model budou považovány za konstanty.

Je však třeba připustit, že zvláště objemový koeficient *K* a fyziologický koeficient *a* se mohou v závislosti na podmínkách reálného provozu měnit, proto je třeba vytvořit několik dynamických modelů v závislosti na různých kombinacích hodnot koeficientů *K* a *a*. V tomto didaktickém případě byly zvoleny tři různé hodnoty obou koeficientů, bude tedy třeba vytvořit devět dynamických modelů bioreaktoru pomocí neuronové sítě. Zvolené hodnoty jsou znázorněny v tabulce 6.2.

Tab. 6.2 – Zvolené modely bioreaktoru

Název sítě	Objemový koeficient *K*	Fyziologický koeficient *a*
NNBIO1	100	2500
NNBIO2	100	4000
NNBIO3	100	6000
NNBIO4	250	2500
NNBIO5	250	4000
NNBIO6	250	6000
NNBIO7	400	2500
NNBIO8	400	4000
NNBIO9	400	6000

6.2.2 Zisk tréninkové množiny dat pro učení neuronové sítě

K zisku tréninkové množiny dat pro všechny modely bude použit matematicko-fyzikální model vytvořený v aplikaci *Matlab – Simulink*. Podrobný postup jeho tvorby je uveden v příloze. Ještě před samotnou simulací sběru dat je však třeba určit interval vzorkování. Při identifikaci lineární soustavy se doporučuje volit interval vzorkování takový, aby bylo obdrženo sedm až patnáct vzorků do doby ustálení při skokové změně na vstupu ([Drábek, 1987]). Bioreaktor je však silně nelineární soustava, jejíž doba do ustálení při skoku na vstupu silně závisí na zvoleném pracovním bodu. Za určitých podmínek má totiž bioreaktor dynamiku poměrně rychlou, zatímco jindy velmi pomalou. Proto byla provedena série experimentů pro různé intervaly vzorkování a bylo zjištěno, že solidních výsledků bude dosaženo například pro interval vzorkování T_s = 1s.

Po volbě intervalu vzorkování již mohlo být přistoupeno k samotnému sběru dat. Sběr pro každou kombinaci koeficientů *K* a *a* byl v simulačním prostředku *Simulink* proveden takovým způsobem, že na vstup byl přiveden generátor náhodných čísel s rovnoměrným rozložením s rozsahem hodnot rovným povolenému rozsahu vstupu do bioreaktoru: $\delta \in \langle 0 \div 0{,}82644 \rangle$. Byl zvláště kladen důraz na to, aby se v množině vstupů vyskytovaly i krajní hodnoty intervalu. Perioda změny vstupu byla zvolena 10 s. Během experimentu pak byly veškeré relevantní hodnoty $(\delta, \chi, \sigma, \omega)$ měřeny v dobách násobků periody vzorkování T_s = 1 s. Experiment trval 2000 vteřin.

6.2.3 Učení neuronových sítí a volba optimální topologie

6.2.3.1 Úvod

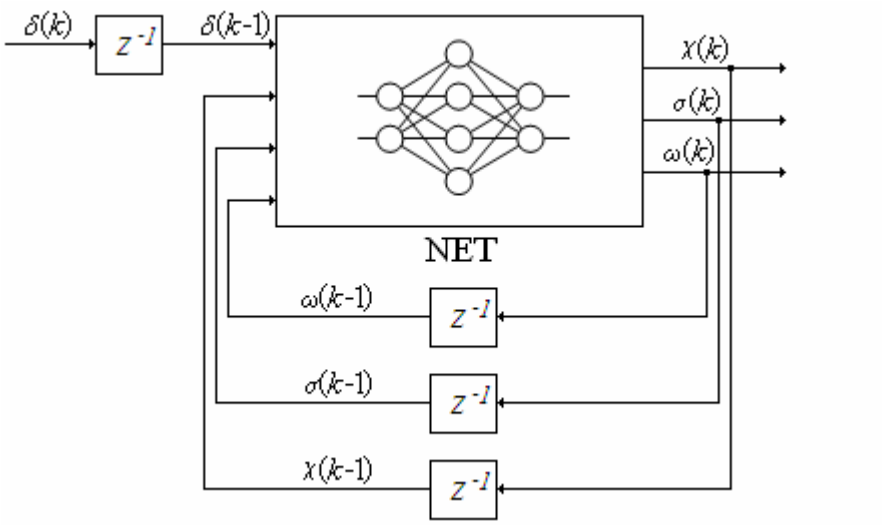
Při volbě počtu vstupů a výstupů neuronové sítě modelující bioreaktor se vycházelo z toho poznatku matematicko-fyzikální analýzy, že dynamické chování bioreaktoru je popsáno soustavou nelineárních diferenciálních rovnic prvního řádu (viz odstavec 4**).** Diferenční model bioreaktoru by pak měl znít za daných podmínek podle soustavy rovnic (6-2).

$$\begin{aligned} \chi(k) &= \varphi_1\left[\chi(k-1), \sigma(k-1), \omega(k-1), \delta(k-1)\right] \\ \sigma(k) &= \varphi_2\left[\chi(k-1), \sigma(k-1), \omega(k-1), \delta(k-1)\right] \\ \omega(k) &= \varphi_3\left[\chi(k-1), \sigma(k-1), \omega(k-1)\right] \end{aligned} \qquad (6\text{-}2)$$

No a cílem je pro každou kombinaci koeficientů *K* a *a* získat neuronovou síť, která bude aproximovat soustavu rovnic (6-2). Lze ji zapsat vztahem (6-3).

$$\begin{bmatrix} \chi(k) \\ \sigma(k) \\ \omega(k) \end{bmatrix} = \text{NET} \begin{bmatrix} \chi(k-1) \\ \sigma(k-1) \\ \omega(k-1) \\ \delta(k-1) \end{bmatrix} \tag{6-3}$$

Taková umělá neuronová síť se pak použije v zapojení znázorněném na obrázku 6.4.



Obr. 6.4 – Neuronová síť modelující bioreaktor

Pro *N* změřených hodnot každé proměnné je tedy třeba seřadit hodnoty do tvaru uvedeného v tabulce 6.3.

Tab. 6.3 – Trénovací množina dat

k	Množina vstupních dat				Množina výstupních dat		
	χ(k-1)	***σ(k-1)***	***ω(k-1)***	***δ(k-1)***	***χ(k)***	***σ(k)***	***ω(k)***
1	*χ(1)*	*σ(1)*	*ω(1)*	*δ(1)*	*χ(2)*	*σ(2)*	*ω(2)*
2	*χ(2)*	*σ(2)*	*ω(2)*	*δ(2)*	*χ(3)*	*σ(3)*	*ω(3)*
…	…	…	…	…	…	…	…
N-2	*χ(N-2)*	*σ(N-2)*	*ω(N-2)*	*δ(N-2)*	*χ(N-1)*	*σ(N-1)*	*ω(N-1)*
N-1	*χ(N-1)*	*σ(N-1)*	*ω(N-1)*	*δ(N-1)*	*χ(N)*	*σ(N)*	*ω(N)*

Takto seřazené hodnoty byly použity k určení optimálních topologií jednotlivých neuronových sítí *NNBIO1* až *NNBIO9*. Na základě předchozích zkušeností získaných při tvorbě statického modelu byl k učení umělých neuronových sítí použit *Levenbergův-Marquardtův algoritmus*, který poskytoval vždy nejlepší výsledky. Jako agregační funkce byla zvolena prostá sumace vstupů, aktivační funkce neuronů ve skrytých vrstvách byly zvoleny shodně jako hyperbolické tangenty a aktivační funkce neuronů ve výstupní vrstvě byly zvoleny lineární s jednotkovou strmostí.

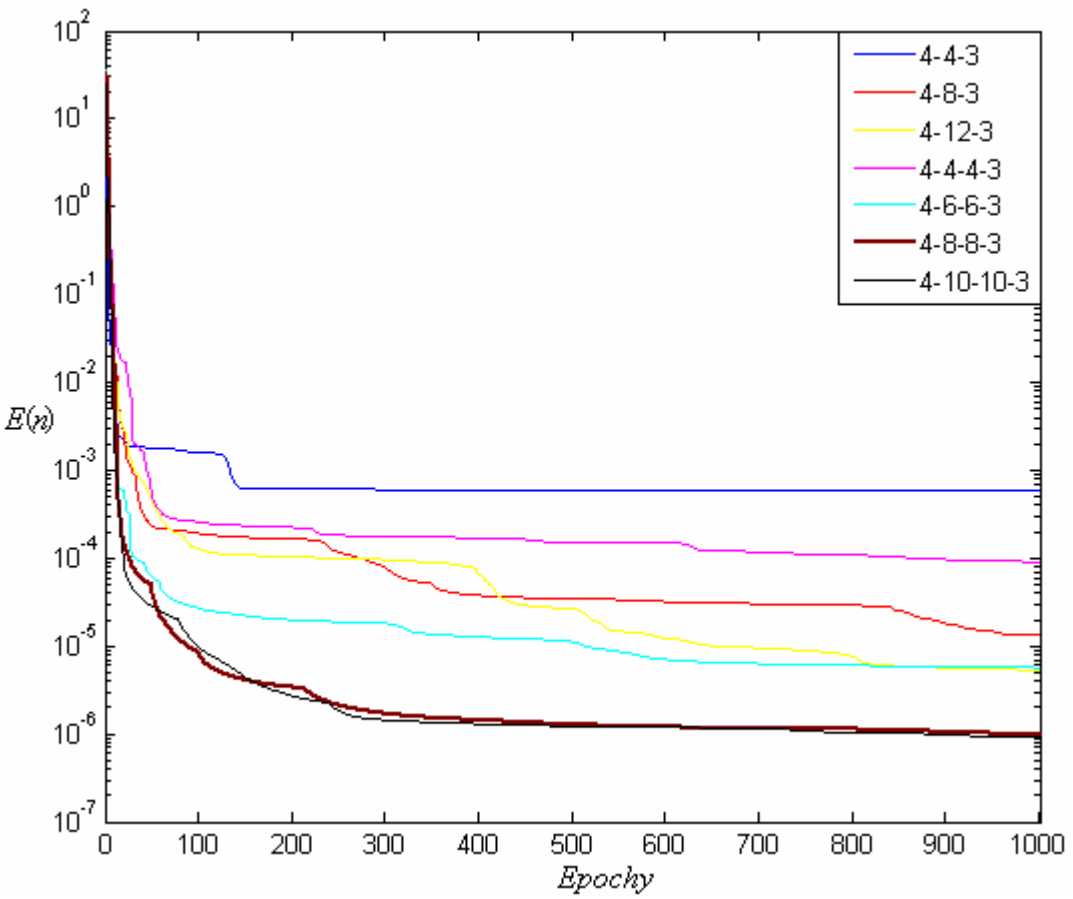
6.2.3.2 Volba optimální topologie sítě *NNBIO1*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$\begin{aligned} K &= 100 \\ a &= 2500 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \tag{6-4}$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.5, konečné hodnoty kriteriálních funkcí pak v tabulce 6.4.



Obr 6.5 – Průběhy kriteriálních funkcí

Tab. 6.4 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$5{,}994 \cdot 10^{-4}$
4 – 8 – 3	$1{,}331 \cdot 10^{-5}$
4 – 12 – 3	$5{,}181 \cdot 10^{-6}$
4 – 4 – 4 – 3	$9{,}051 \cdot 10^{-5}$
4 – 6 – 6 – 3	$5{,}665 \cdot 10^{-6}$
4 – 8 – 8 – 3	$9{,}717 \cdot 10^{-7}$
4 – 10 – 10 – 3	$9{,}108 \cdot 10^{-7}$

Jak je patrno z obrázku 6.5 a z tabulky 6.4, výkon jednotlivých sítí se více méně zlepšoval s narůstající topologií až do topologie 4 – 8 – 8 – 3. Větší složitost topologie již výrazné zlepšení nepřinesla, proto bude pro testování použita natrénovaná umělá neuronová síť o topologii 4 – 8 – 8 – 3, která v dalším textu ponese označení *NNBIO1*.

6.2.3.3 Volba optimální topologie sítě *NNBIO2*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

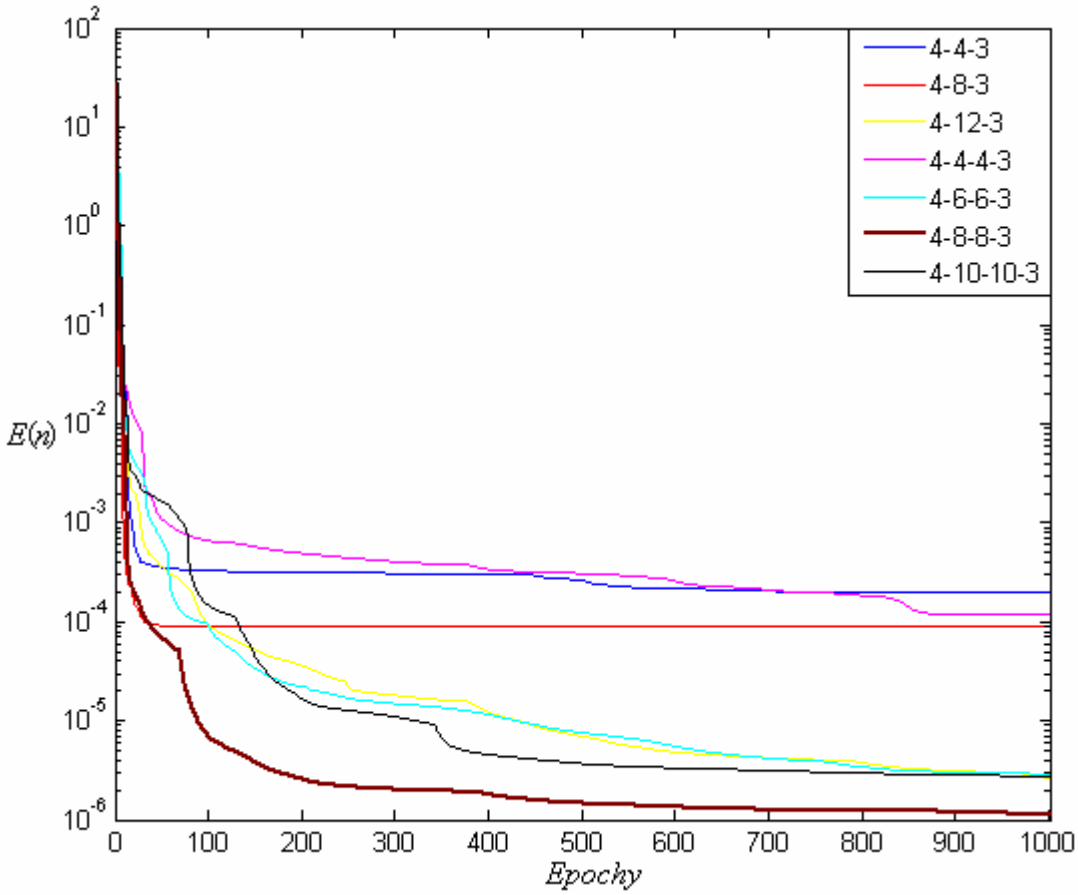
$$\begin{aligned} K &= 100 \\ a &= 4000 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \qquad (6\text{-}5)$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.6, konečné hodnoty kriteriálních funkcí pak v tabulce 6.5.

Tab. 6.5 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$1{,}969 \cdot 10^{-4}$
4 – 8 – 3	$9{,}002 \cdot 10^{-5}$
4 – 12 – 3	$2{,}590 \cdot 10^{-6}$
4 – 4 – 4 – 3	$1{,}187 \cdot 10^{-4}$
4 – 6 – 6 – 3	$2{,}905 \cdot 10^{-6}$
4 – 8 – 8 – 3	$1{,}131 \cdot 10^{-6}$
4 – 10 – 10 – 3	$2{,}698 \cdot 10^{-6}$



Obr. 6.6 – Průběhy kriteriálních funkcí

Zhodnocením obrázku 6.6 a tabulky 6.5 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO2*.

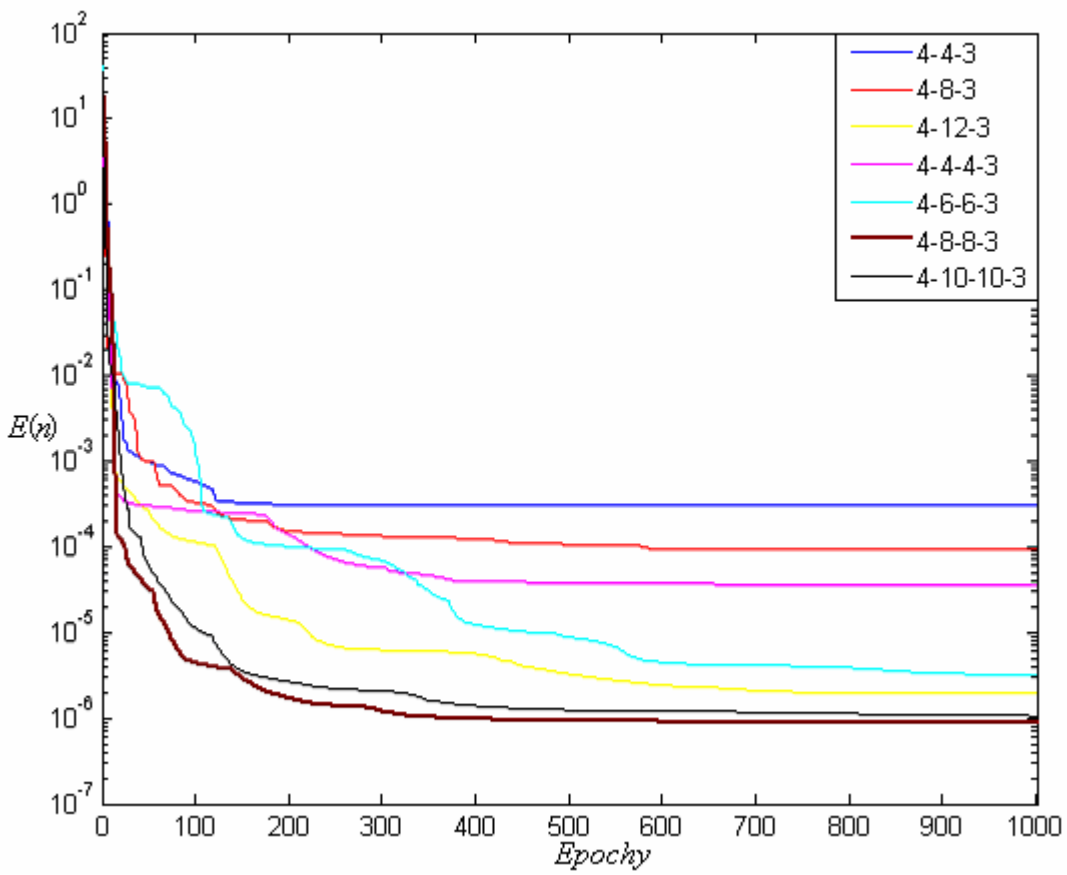
6.2.3.4 Volba optimální topologie sítě *NNBIO3*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$\begin{aligned} K &= 100 \\ a &= 6000 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \qquad (6\text{-}6)$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.7, konečné hodnoty kriteriálních funkcí pak v tabulce 6.6.



Obr. 6.7 – Průběhy kriteriálních funkcí

Tab. 6.6 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$3{,}097 \cdot 10^{-4}$
4 – 8 – 3	$9{,}177 \cdot 10^{-5}$
4 – 12 – 3	$1{,}930 \cdot 10^{-6}$
4 – 4 – 4 – 3	$3{,}525 \cdot 10^{-5}$
4 – 6 – 6 – 3	$3{,}164 \cdot 10^{-6}$
4 – 8 – 8 – 3	$8{,}644 \cdot 10^{-7}$
4 – 10 – 10 – 3	$1{,}051 \cdot 10^{-6}$

Zhodnocením obrázku 6.7 a tabulky 6.6 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO3*.

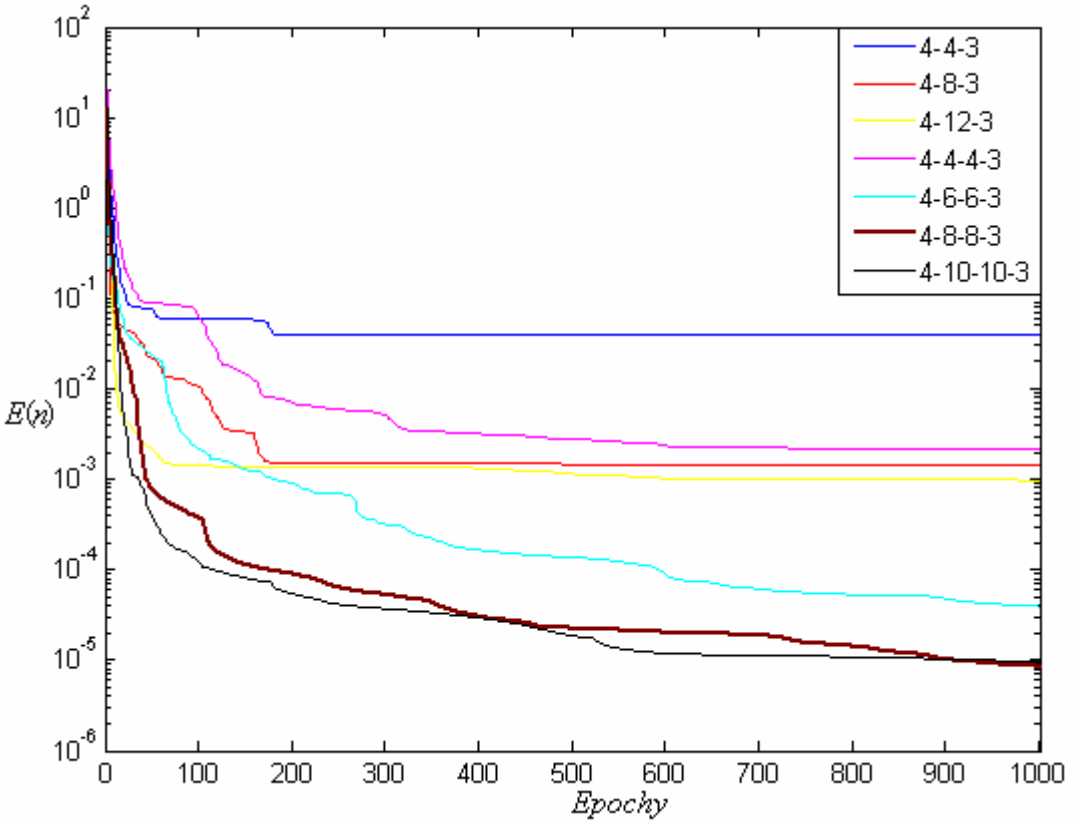
6.2.3.5 Volba optimální topologie sítě *NNBIO4*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$K = 250$$
$$a = 2500$$
$$\sigma^0 = 10 \qquad (6\text{-}7)$$
$$\omega^0 = 10$$
$$\chi^0 = 10$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.8, konečné hodnoty kriteriálních funkcí pak v tabulce 6.7.



Obr. 6.8 – Průběhy kriteriálních funkcí

Tab. 6.7 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$4{,}034 \cdot 10^{-2}$
4 – 8 – 3	$1{,}459 \cdot 10^{-3}$
4 – 12 – 3	$9{,}649 \cdot 10^{-4}$
4 – 4 – 4 – 3	$2{,}134 \cdot 10^{-3}$
4 – 6 – 6 – 3	$3{,}934 \cdot 10^{-5}$
4 – 8 – 8 – 3	$8{,}496 \cdot 10^{-6}$
4 – 10 – 10 – 3	$9{,}740 \cdot 10^{-6}$

Zhodnocením obrázku 6.8 a tabulky 6.7 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO4*.

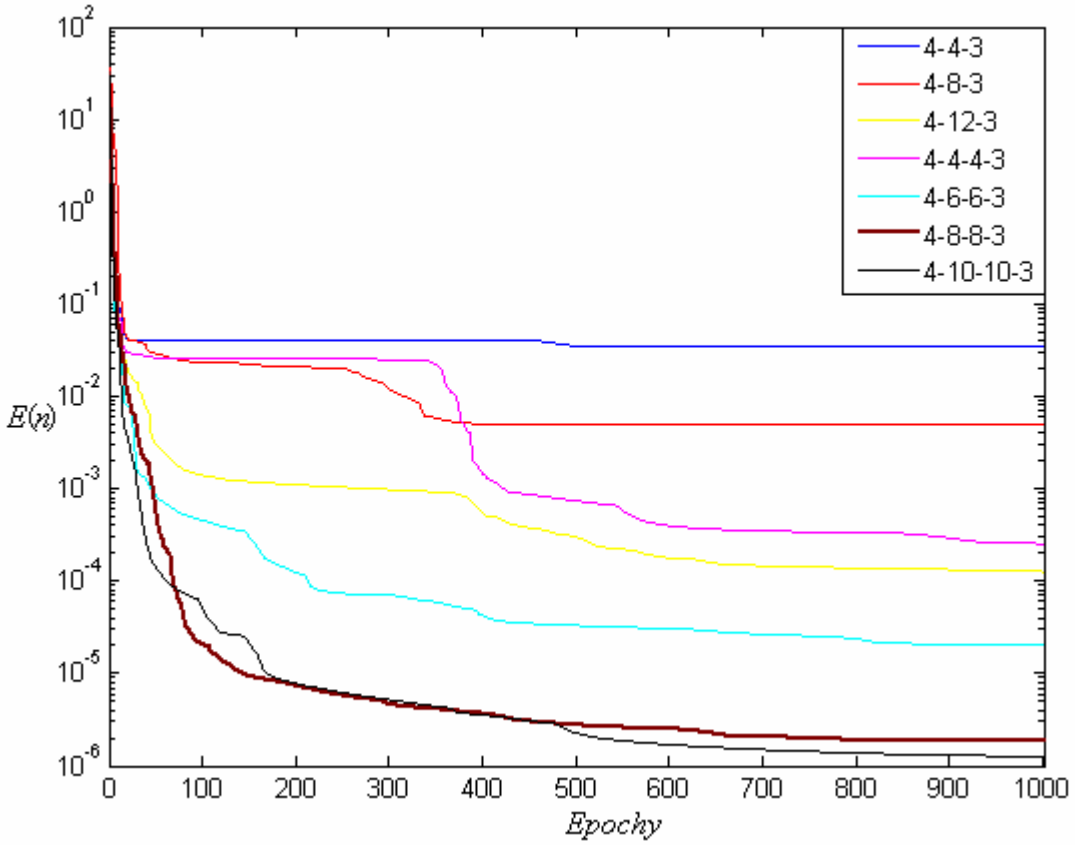
6.2.3.6 Volba optimální topologie sítě *NNBIO5*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$\begin{aligned} K &= 250 \\ a &= 4000 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \tag{6-8}$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.9, konečné hodnoty kriteriálních funkcí pak v tabulce 6.8.



Obr. 6.9 – Průběhy kriteriálních funkcí

Tab. 6.8 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$3{,}432 \cdot 10^{-2}$
4 – 8 – 3	$4{,}919 \cdot 10^{-3}$
4 – 12 – 3	$1{,}253 \cdot 10^{-4}$
4 – 4 – 4 – 3	$2{,}543 \cdot 10^{-4}$
4 – 6 – 6 – 3	$2{,}005 \cdot 10^{-5}$
4 – 8 – 8 – 3	$1{,}842 \cdot 10^{-6}$
4 – 10 – 10 – 3	$1{,}256 \cdot 10^{-6}$

Zhodnocením obrázku 6.9 a tabulky 6.8 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3, neboť zlepšení výkonu sítě s topologií 4 – 10 – 10 – 3 je zanedbatelné. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO5*.

6.2.3.7 Volba optimální topologie sítě *NNBIO6*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

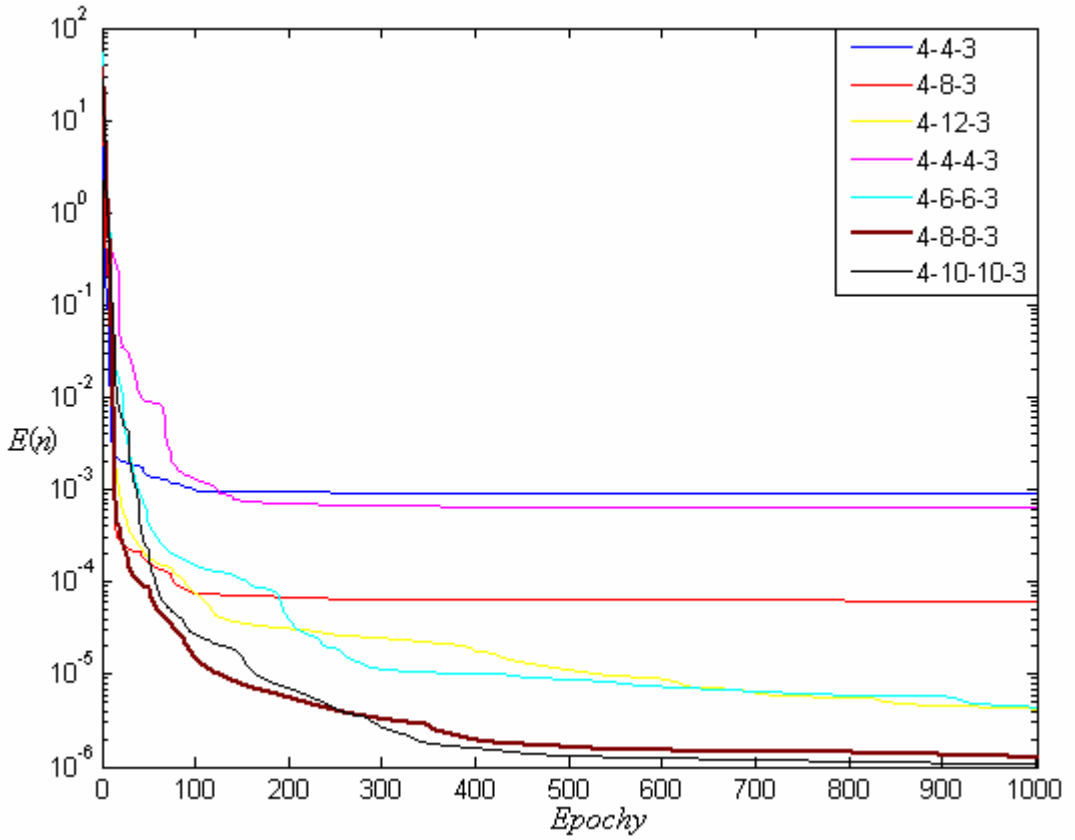
$$\begin{aligned} K &= 250 \\ a &= 6000 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \tag{6-9}$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.10, konečné hodnoty kriteriálních funkcí pak v tabulce 6.9.

Tab. 6.9 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$3{,}432 \cdot 10^{-2}$
4 – 8 – 3	$4{,}919 \cdot 10^{-3}$
4 – 12 – 3	$1{,}253 \cdot 10^{-4}$
4 – 4 – 4 – 3	$2{,}543 \cdot 10^{-4}$
4 – 6 – 6 – 3	$2{,}005 \cdot 10^{-5}$
4 – 8 – 8 – 3	$1{,}842 \cdot 10^{-6}$
4 – 10 – 10 – 3	$1{,}256 \cdot 10^{-6}$



Obr. 6.10 – Průběhy kriteriálních funkcí

Zhodnocením obrázku 6.10 a tabulky 6.9 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3, neboť zlepšení výkonu sítě s topologií 4 – 10 – 10 – 3 je zanedbatelné. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO6*.

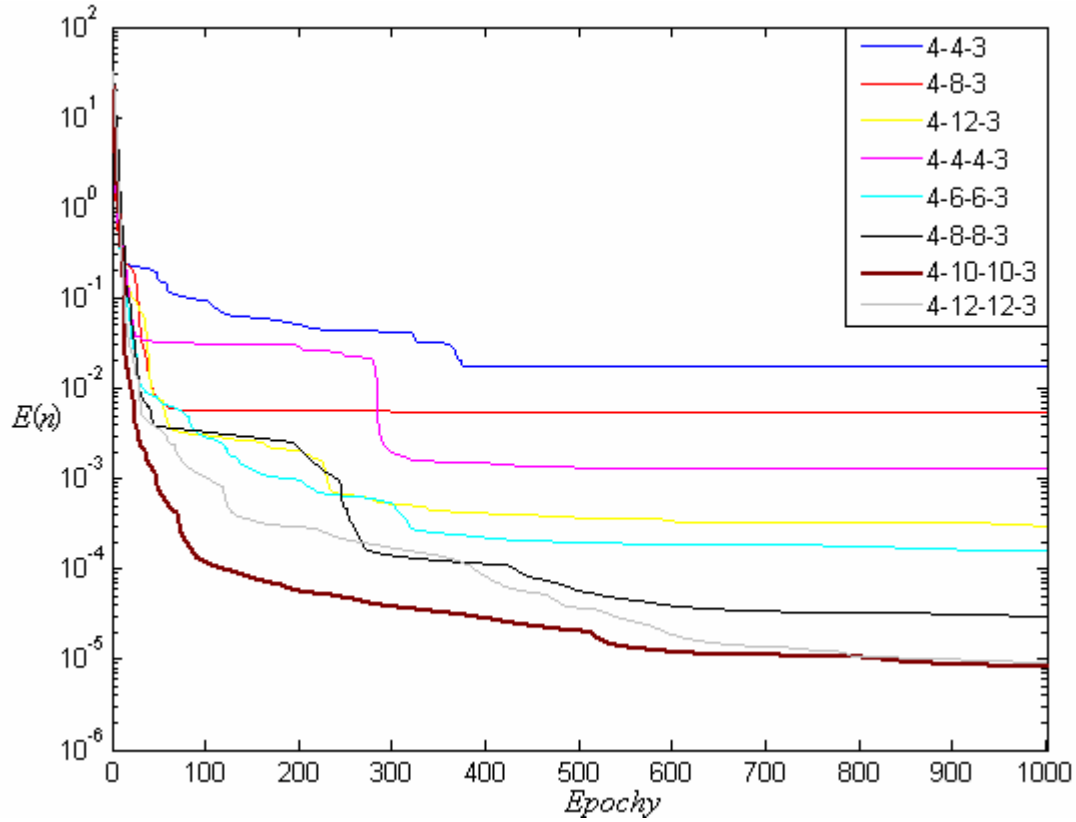
6.2.3.8 Volba optimální topologie sítě *NNBIO7*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$\begin{aligned} K &= 400 \\ a &= 2500 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \qquad (6\text{-}10)$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.11, konečné hodnoty kriteriálních funkcí pak v tabulce 6.10.



Obr. 6.11 – Průběhy kriteriálních funkcí

Tab. 6.10 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$1{,}725 \cdot 10^{-2}$
4 – 8 – 3	$5{,}541 \cdot 10^{-3}$
4 – 12 – 3	$2{,}970 \cdot 10^{-4}$
4 – 4 – 4 – 3	$1{,}274 \cdot 10^{-3}$
4 – 6 – 6 – 3	$1{,}620 \cdot 10^{-4}$
4 – 8 – 8 – 3	$2{,}970 \cdot 10^{-5}$
4 – 10 – 10 – 3	$8{,}255 \cdot 10^{-6}$
4 – 12 – 12 – 3	$9{,}187 \cdot 10^{-6}$

Zhodnocením obrázku 6.11 a tabulky 6.10 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 10 – 10 – 3, neboť zlepšení výkonu sítě s topologií 4 – 12 – 12 – 3 je zanedbatelné. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO7*.

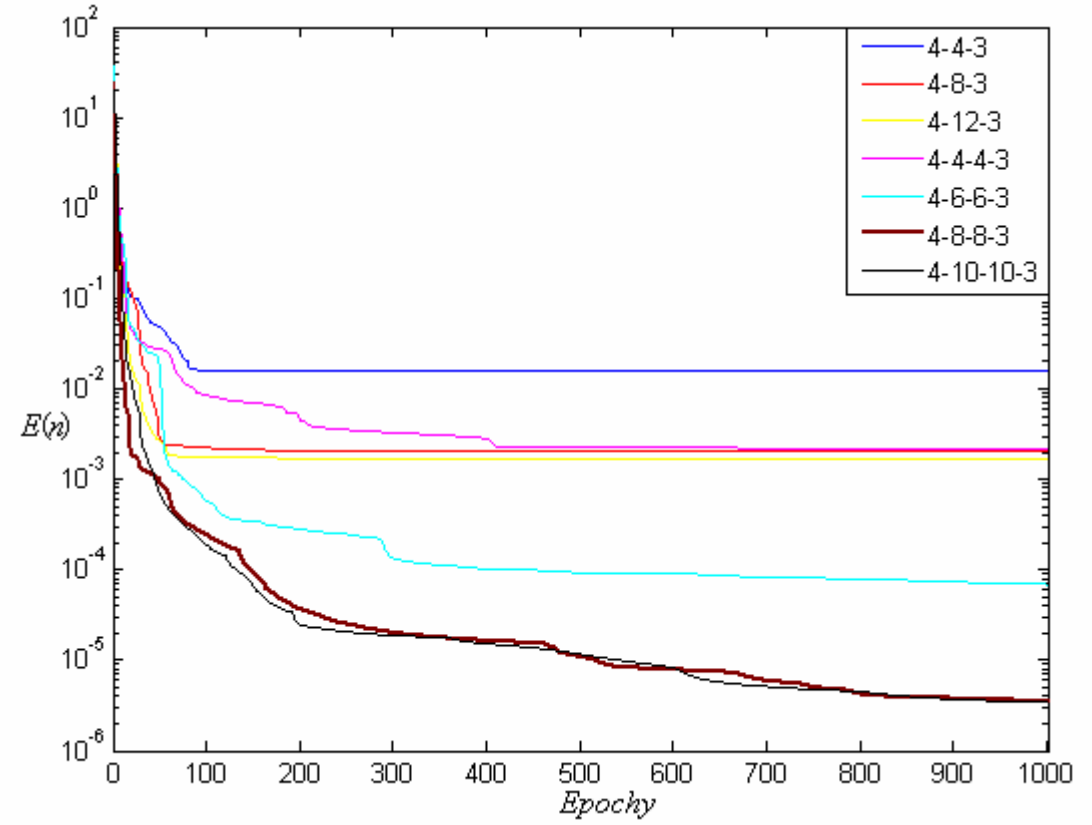
6.2.3.9 Volba optimální topologie sítě *NNBIO8*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$K = 400$$
$$a = 4000$$
$$\sigma^0 = 10 \qquad (6\text{-}11)$$
$$\omega^0 = 10$$
$$\chi^0 = 10$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.12, konečné hodnoty kriteriálních funkcí pak v tabulce 6.11.



Obr. 6.12 – Průběhy kriteriálních funkcí

Tab. 6.11 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$1{,}613 \cdot 10^{-2}$
4 – 8 – 3	$2{,}015 \cdot 10^{-3}$
4 – 12 – 3	$1{,}644 \cdot 10^{-3}$
4 – 4 – 4 – 3	$2{,}199 \cdot 10^{-3}$
4 – 6 – 6 – 3	$6{,}901 \cdot 10^{-5}$
4 – 8 – 8 – 3	$3{,}498 \cdot 10^{-6}$
4 – 10 – 10 – 3	$3{,}436 \cdot 10^{-6}$

Zhodnocením obrázku 6.12 a tabulky 6.11 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3, neboť zlepšení výkonu sítě s topologií 4 – 10 – 10 – 3 je zanedbatelné. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO8*.

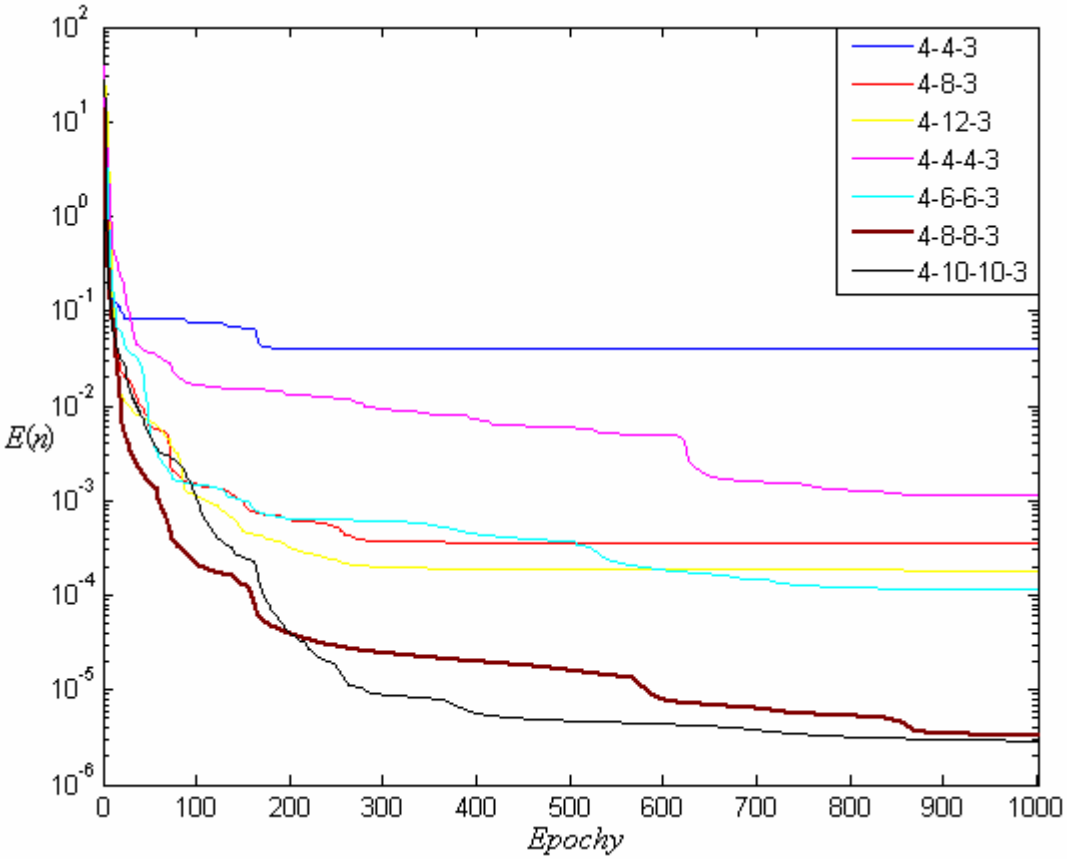
6.2.3.10 Volba optimální topologie sítě *NNBIO9*

Podle odstavce 6.2.2 byla získána trénovací množina dat, přičemž konstantní hodnoty byly voleny následovně.

$$\begin{aligned} K &= 400 \\ a &= 6000 \\ \sigma^0 &= 10 \\ \omega^0 &= 10 \\ \chi^0 &= 10 \end{aligned} \tag{6-12}$$

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Analogicky k odstavci 6.2.3.2 byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 6.13, konečné hodnoty kriteriálních funkcí pak v tabulce 6.12.



Obr. 6.13 – Průběhy kriteriálních funkcí

Tab. 6.12 – Konečné hodnoty kriteriálních funkcí

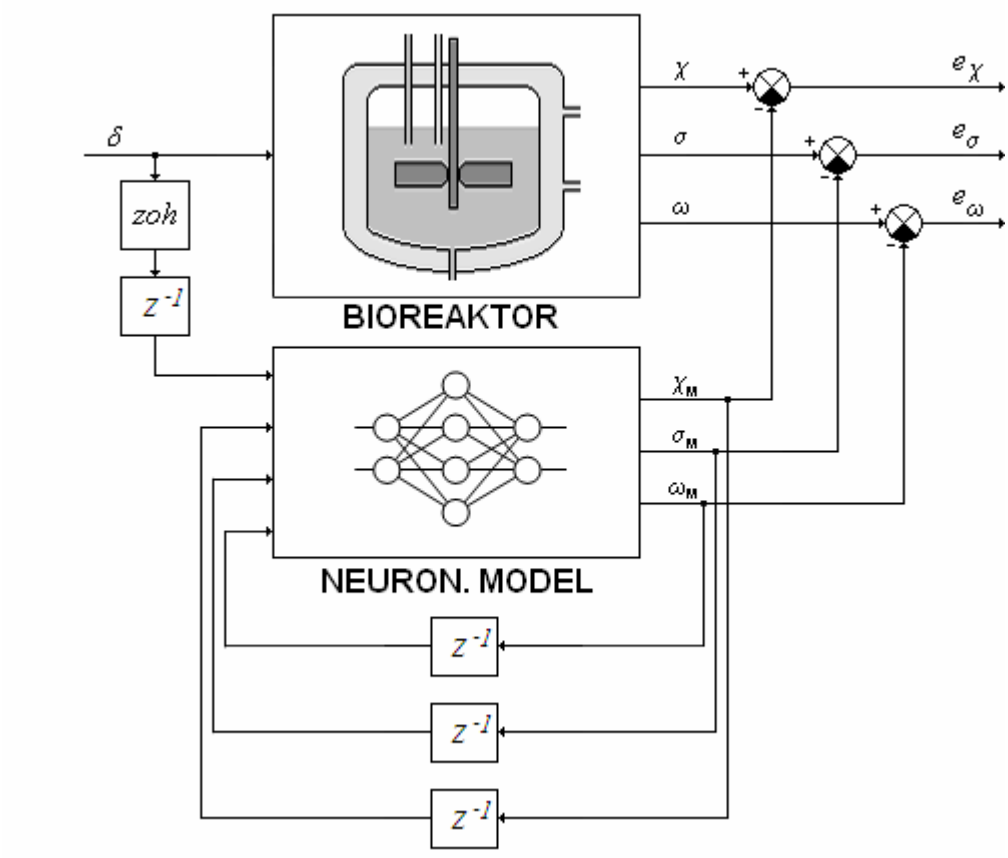
Topologie sítě	Hodnota kriteria $E(N)$
4 – 4 – 3	$4{,}015 \cdot 10^{-2}$
4 – 8 – 3	$3{,}510 \cdot 10^{-4}$
4 – 12 – 3	$1{,}831 \cdot 10^{-4}$
4 – 4 – 4 – 3	$1{,}140 \cdot 10^{-3}$
4 – 6 – 6 – 3	$1{,}135 \cdot 10^{-4}$
4 – 8 – 8 – 3	$3{,}271 \cdot 10^{-6}$
4 – 10 – 10 – 3	$2{,}877 \cdot 10^{-6}$

Zhodnocením obrázku 6.13 a tabulky 6.12 lze dojít k závěru, že ze zkoumaných topologií se jako optimální jeví topologie 4 – 8 – 8 – 3, neboť zlepšení výkonu sítě s topologií 4 – 10 – 10 – 3 je zanedbatelné. Tato topologie bude proto použita pro testování a bude označena zkratkou *NNBIO9*.

6.2.4 Testování natrénovaných umělých neuronových sítí

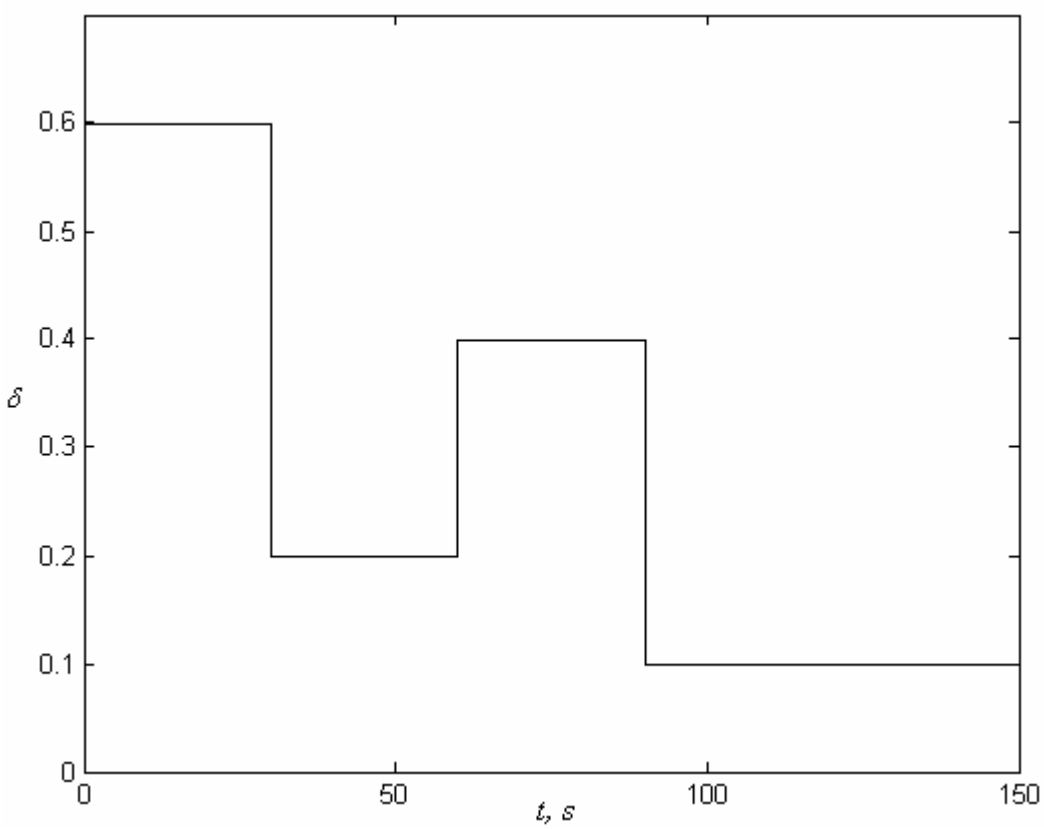
6.2.4.1 Postup testování a volba testovací množiny

Testování jednotlivých neuronových sítí bude probíhat v zapojení podle obrázku 6.14.



Obr. 6.14 – Schéma testování neuronových sítí

Jako vstup δ pro testování byla ve všech případech zvolena stupňovitá funkce znázorněná na obrázku 6.15, která poskytla dostatečné vybuzení systému k tomu, aby bylo možno posoudit správnost modelu. Testování vždy začínalo z ustáleného stavu pro vstup $\delta = 0{,}1$.

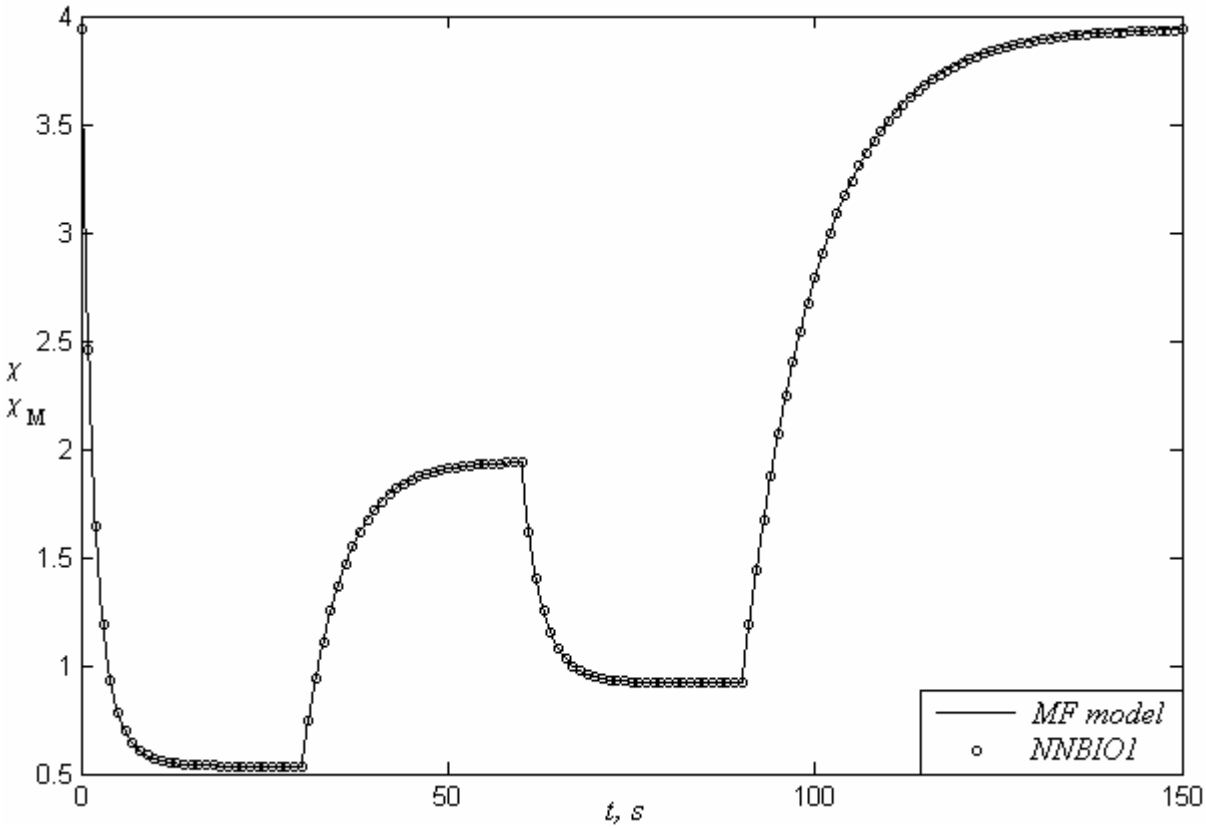


Obr. 6.15 – Průběh vstupu při testování

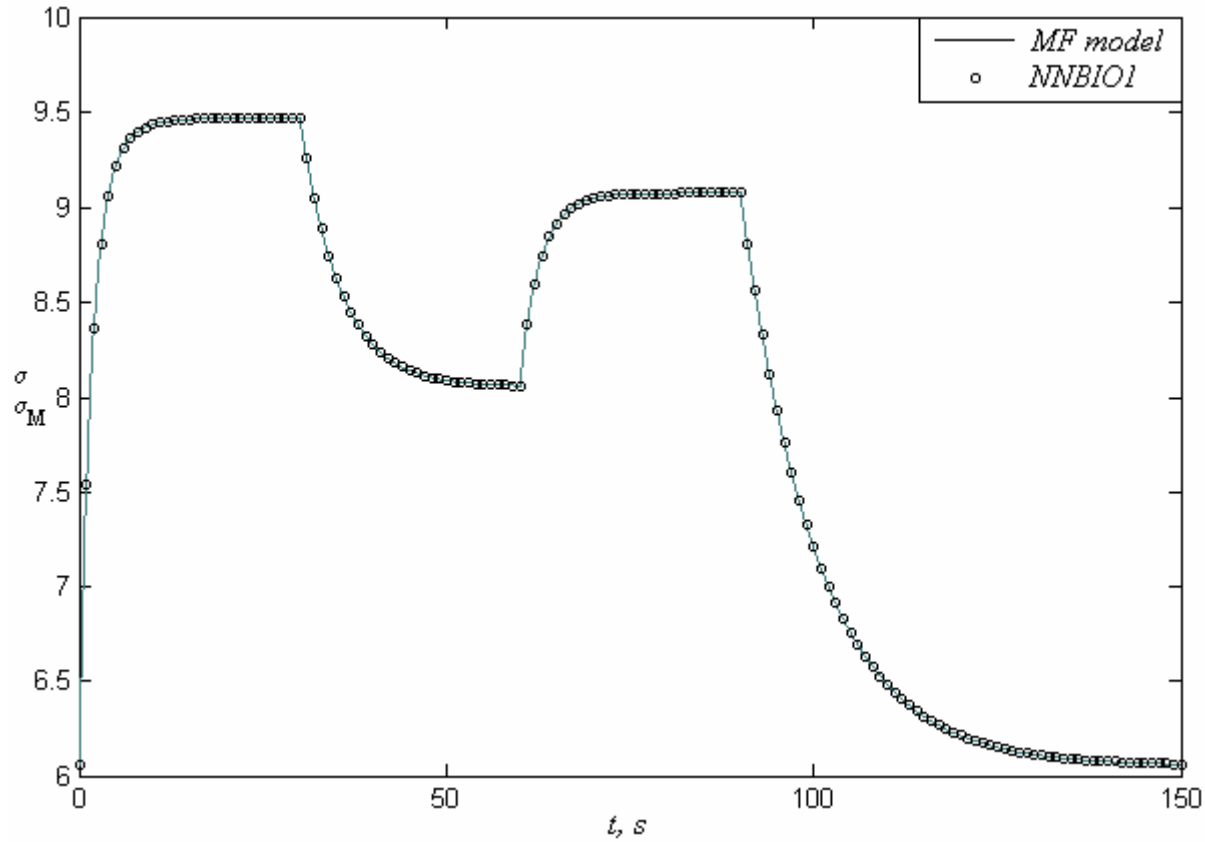
Vzhledem k počtu neuronových sítí určených k testování byl zvolen takový postup, že úplná prezentace testování bude provedena pouze pro síť *NNBIO1*, nejdůležitější data získaná při testování ostatních sítí budou uvedena v příloze.

6.2.4.2 Testování umělé neuronové sítě *NNBIO1*

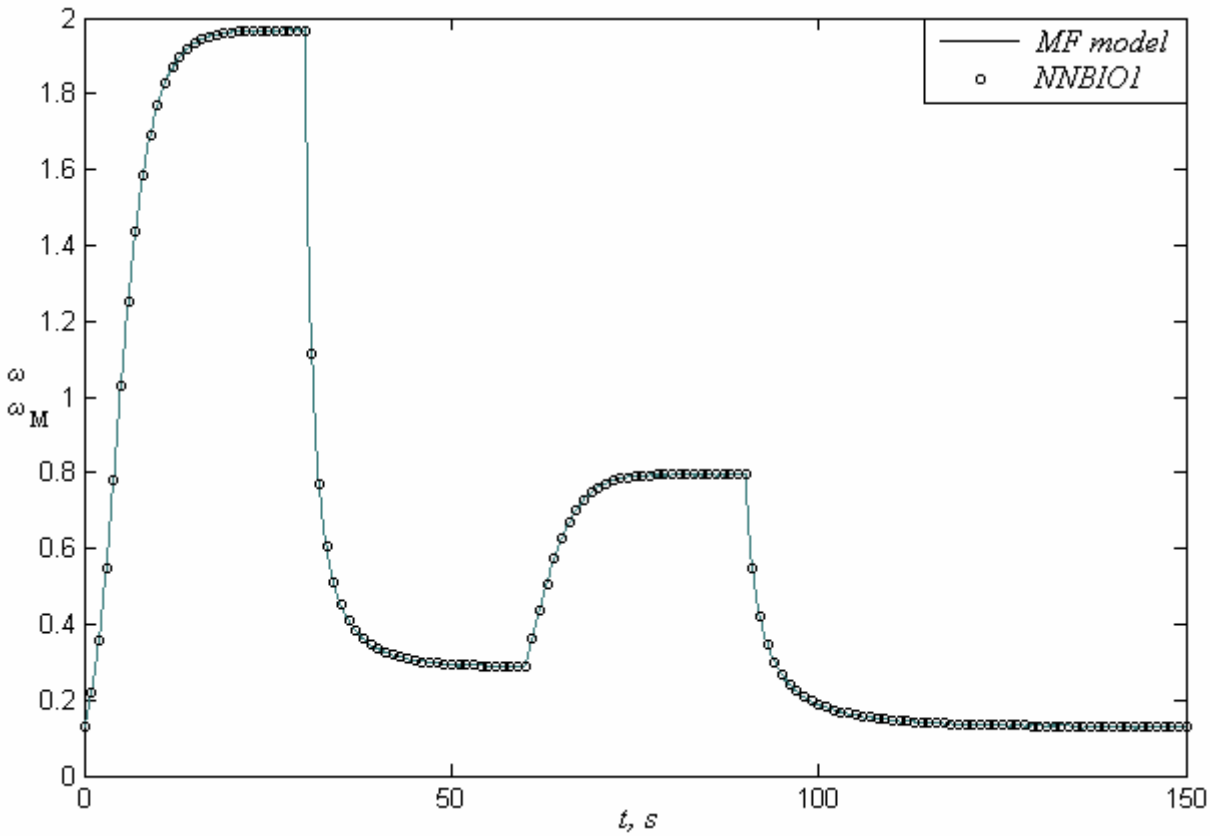
Tato neuronová síť byla trénovaná za platnosti parametrů vyjádřených vztahem (6-4), proto byly hodnoty parametrů matematicko-fyzikálního modelu nastaveny také podle vztahu (6-4). Následně byla v prostředku *Simulink* provedena simulace se vstupním signálem vyjádřeným na obrázku 6.15. Při simulaci byly snímány a vzájemně porovnány všechny výstupy. Vzhledem k tomu, že neuronová síť *NNBIO1* funguje jako diferenční rovnice s intervalem vzorkování T_s = 1 s, byly výstupy z neuronové sítě snímány v tomto intervalu vzorkování a zobrazeny budou jako diskrétní body, zatímco výstupy z matematicko-fyzikálního modelu budou zobrazeny spojitě. Konfrontace výstupů z matematicko-fyzikálního modelu a neuronové sítě jsou uvedeny na obrázcích 6.16 až 6.18.



Obr. 6.16 – Porovnání průběhů veličin χ, χ_M



Obr. 6.17 – Porovnání průběhů veličin σ, σ_M

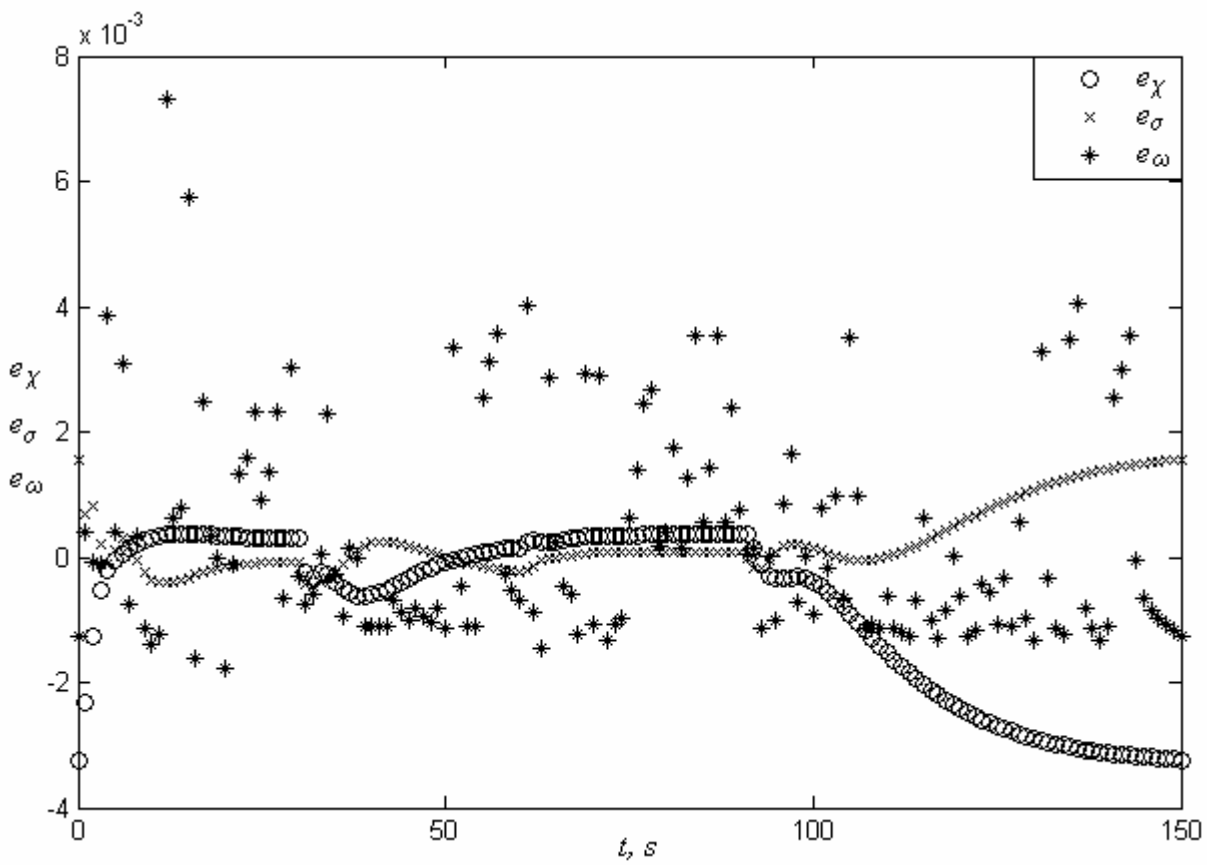


Obr. 6.18 – Porovnání průběhů veličin ω, ω_{M}

Na obrázcích 6.16 až 6.18 je patrné, že v násobcích intervalu vzorkování jsou si výstupy neuronové sítě a matematicko-fyzikálního modelu velmi blízké.

Pro podrobnější zobrazení rozdílů hodnot na výstupech byly ještě do obrázku 6.19 vyneseny průběhy veličin e_χ, e_σ, e_ω v bodech násobku intervalu vzorkování.

Z obrázku 6.19 lze vyčíst jednak to, že velikosti chyb dosahují maximálních hodnot v řádu 10^{-3}, tedy jsou oproti absolutním hodnotám veličin χ, σ, ω zanedbatelné, a dále je možno si všimnout, že chyba e_ω se svým chováním podobá vysokofrekvenčnímu šumu, zatímco se chyby e_χ a e_σ svým průběhem blíží spíše nízkofrekvenčnímu driftu.



Obr. 6.19 – Průběh rozdílů hodnot na výstupech

6.3 Zhodnocení tvorby dynamického modelu bioreakce

V oblasti modelování poskytují umělé neuronové sítě velmi silný prostředek. I model poměrně složité nelineární soustavy (průtočného bioreaktoru), byl získán s velmi velkou přesností v celém rozsahu oboru hodnot vstupní veličiny. Tvorba neuronového modelu je však v praxi provázena několika problémy. Sporné otázky kolem volby optimální topologie a samotného trénování byly diskutovány dříve, při tvorbě dynamických modelů však vyvstávají další. Jedním z nich je volba řádu nelineární diferenční rovnice, kterou se nahrazuje reálná soustava při identifikaci. Model, řešený v této práci, byl této volby ušetřen, neboť řády diferenčních rovnic byly známy z matematicko-fyzikálního modelu, ovšem v praktickém případě nebývá matematicko-fyzikální model znám, je třeba řády volit a návrhy volby řádů těchto nelineárních diferenčních rovnic jsou v literatuře uvedeny jen velmi vágně. Zatímco při klasické identifikaci soustavy na tvar lineární diferenční rovnice je metodika volby řádu modelu popsána velmi podrobně (například [Drábek, 1987]), v případě nelineárního neuronového modelu bylo pouze nalezeno doporučení, že ve většině případu postačí druhý až třetí řád.

Dalším, možná vůbec nejdůležitějším, problémem je samotné možné využití neuronového modelu. V oblasti řízení bývají modely technologických procesů využívány k návrhu regulátoru. Ovšem každý postup návrhu regulátoru vyžaduje model v určitém tvaru, nejčastěji bývá požadován vstupně-výstupní lineární tvar $A \cdot y = B \cdot u$, někdy nazývaný zkratkou ARMA model (z angl. *Auto Regression Moving Average*). Méně často je požadován stavový model. Neuronový model se však od obou těchto případů liší a není proto k těmto metodám návrhu regulátoru vhodný. Jednu cestu řešení tohoto problému ukazuje Nguyen [2003], který využívá neuronový model při adaptivním řízení jako náhradu

skutečného zařízení, čímž se regulační obvod stává jednodušším zvláště z pohledu datových toků mezi reálným zařízením a řídicím počítačem. Jinou cestu ukazuje Leondes [1998], v jehož publikaci je několik klasických návrhů regulátoru modifikováno tak, aby při nich byl neuronový model využitelný. Jmenovitě to jsou například řízení s vnitřním modelem (IMC – z angl. *Internal Model Control*), či řízení s referenčním modelem (MRC – z angl. *Model Reference Control*). Určitou možností využití neuronového modelu se zdá být také prediktivní řízení, které na model soustavy neklade příliš svazující podmínky a nezavrhuje nelineární modely soustavy, čímž otevírá cestu, jak neuronový model případně využít.

Jestliže se nechá stranou využití neuronového modelu, pak je tento model se soustavou při dostatečném množství vrstev a počtu neuronů téměř totožný. Jeho navržení je při použití moderní výpočetní techniky velmi rychlé a nevyžaduje mnoho informací o identifikované soustavě. Matematicko-fyzikální model oproti tomu sice poskytuje informace o procesech uvnitř soustavy, požaduje ovšem znalost množství fyzikálních konstant, které jsou v praxi velmi těžce stanovitelné. Proto, pokud řešení problému přímo nevyžaduje matematicko-fyzikální model a je možnost dostatečného proměření bioreaktoru, je vhodnější použít neuronový model.

7 Řízení bioreaktoru pomocí umělých neuronových sítí

7.1 Úvod

Neuronové sítě jsou v současné době při řízení technologických procesů v praxi využívány sporadicky pouze u speciálních případů, ovšem je třeba poznamenat, že se míra jejich použití rozšiřuje. Malý podíl neuronových sítí v praxi je způsoben z části tím, že jen málo technologických procesů není možno regulovat pomocí klasických metod, jejichž algoritmy návrhu jsou lépe teoreticky popsány a jejich použití je rutinní. U klasických metod řízení se také dají jednodušším způsobem určit významné vlastnosti systému, jako je například stabilita.

Ovšem existují případy, kdy je neuronovou síť při řízení výhodné použít, zvláště při regulaci významně nelineárních systémů a složitých systémů. V následujících odstavcích budou popsány a některé návrhy řízení pomocí neuronových sítí. Jejich použití bude prezentováno na bioreaktoru popsaném v odstavci 4.

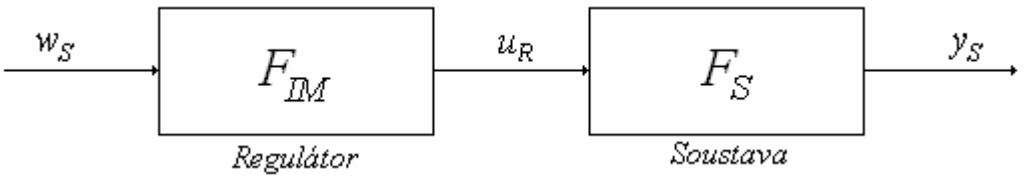
7.2 Přímé inverzní řízení pomocí umělé neuronové sítě

7.2.1 Inverzní model

Při *přímém inverzním řízení* se využívá vlastnosti inverzního modelu definované vztahem (7-1), který by měl být platný pro diskrétní i spojité přenosy.

$$F_s \cdot F_{IM} = 1 \tag{7-1}$$

Ze vztahu (7-1) je zřejmé, že při zapojení regulačního obvodu podle obrázku 7.1 se teoreticky žádaná hodnota po průchodu systémem přenese na výstup a bude dosažen velmi kvalitní regulační pochod.



Obr. 7.1 – Otevřený regulační obvod při přímém inverzním řízení

Pokud je soustava popsána diferenční rovnicí (7-2), pak pro její inverzní model musí platit diferenční rovnice (7-3).

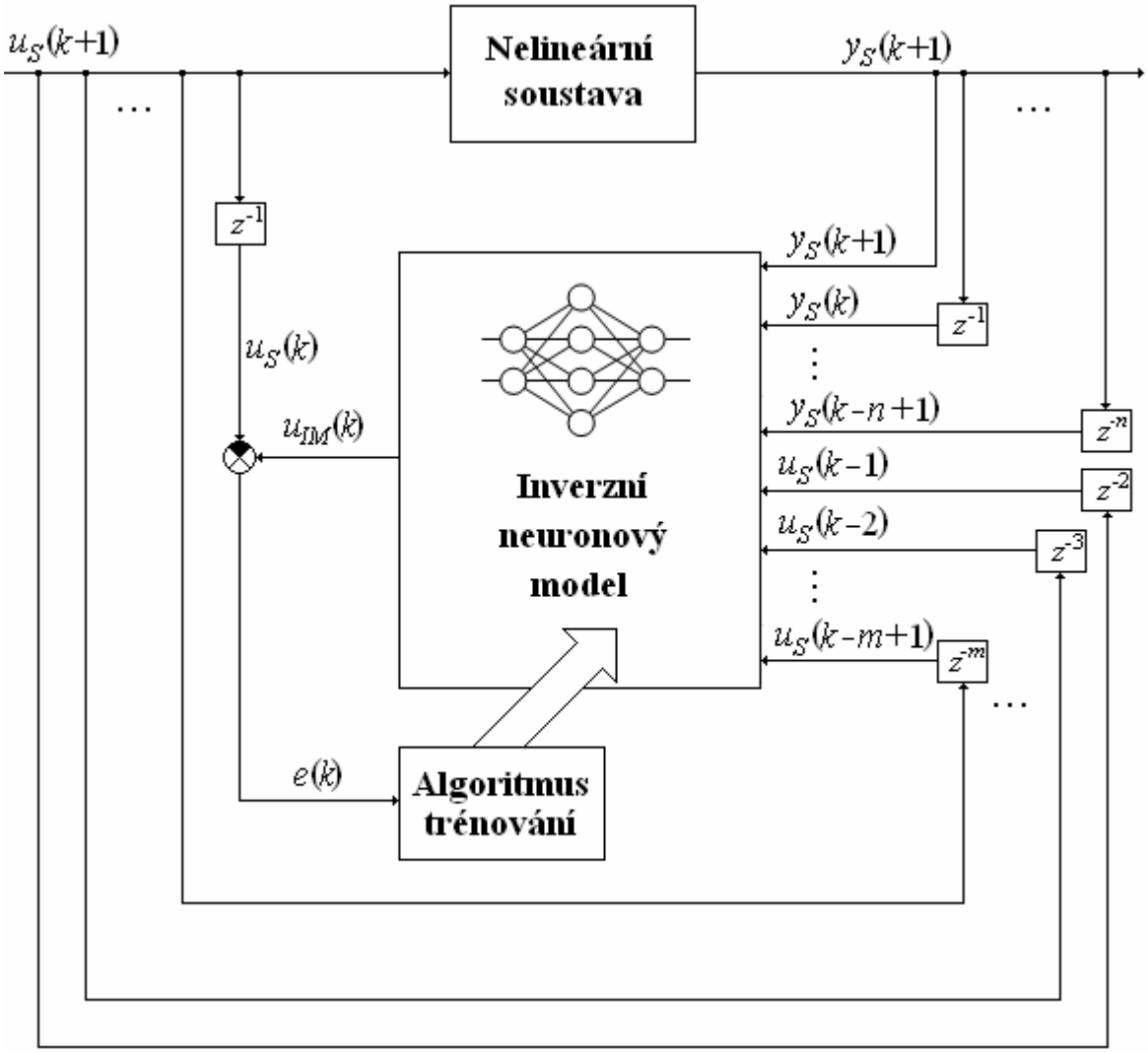
$$y_S(k) = \varphi \cdot \left[y_S(k-1), y_S(k-2), \cdots, y_R(k-n), u_R(k-1), u_R(k-2), \cdots, u_R(k-m) \right] \tag{7-2}$$

$$u_R(k) = \varphi^{-1} \cdot \left[y_S(k+1), y_S(k), \cdots, y_S(k-n+1), u_R(k-1), u_R(k-2), \cdots, u_R(k-m+1) \right] \tag{7-3}$$

Je zřejmé, že diferenční rovnice inverzního modelu (7-3) není kvůli zjevnému porušení kauzality výpočtu realizovatelná, proto nemůže být inverzní model ve formě diferenční rovnice v tomto tvaru použit. Ovšem nic nebrání tomu, aby diferenční rovnice (7-3) byla namodelována pomocí neuronové sítě, pokud by byla použita metoda off-line.

7.2.2 Tvorba inverzní neuronové sítě

Návrh topologie inverzní neuronové sítě zůstává shodný jako u dopředné neuronové sítě. Navrhnou se agregační a aktivační funkce jednotlivých neuronů, počet vstupů a výstupů, počet skrytých vrstev a počet neuronů v těchto vrstvách. Jediný rozdíl spočívá ve formálním schématu trénování, které je pro obecnou SISO soustavu (rovnice (7-2)) uvedeno na obrázku 7.2.



Obr. 7.2 – Trénování inverzního neuronového modelu

7.2.3 Popis přímého inverzního řízení za použití inverzního neuronového regulátoru

Po obdržení správně natrénovaného inverzního modelu zůstává otázkou samotné zapojení regulačního obvodu. Jednou z možností je použití následujícího myšlenkového postupu:

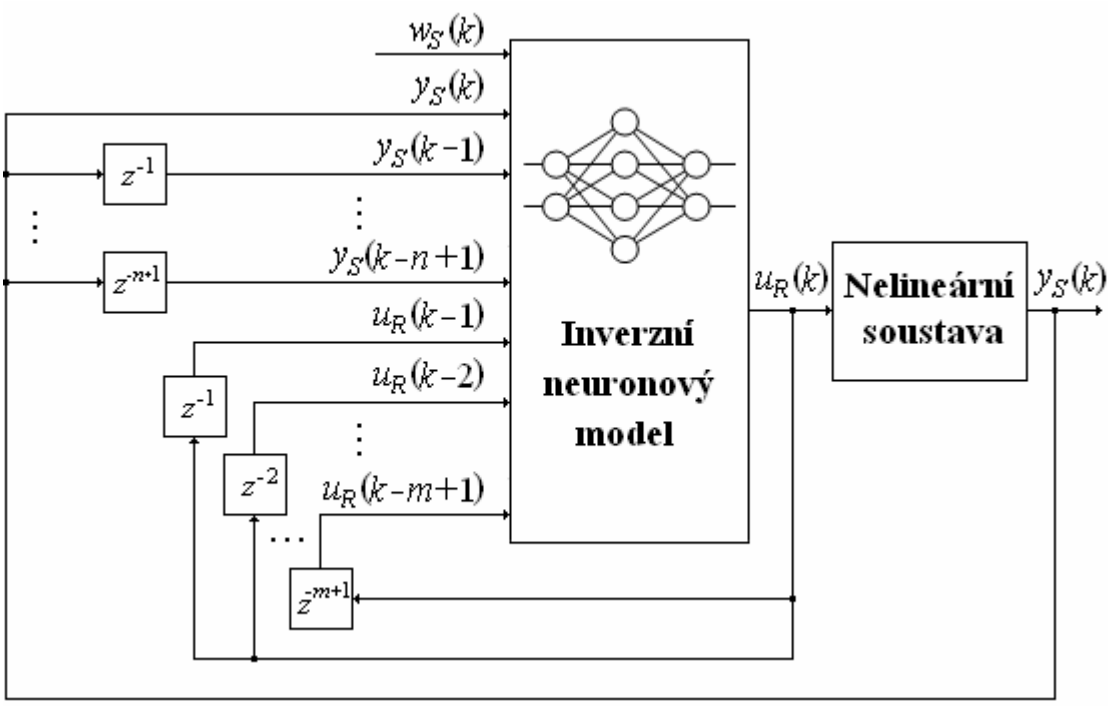
> Pokud je řízená soustava popsaná nelineární diferenční rovnicí (7-2) a pro obdržený inverzní neuronový model platí rovnice (7-3), je možno položit požadavek (7-4), tedy je požadováno, aby regulovaná veličina dosáhla žádané veličiny po jednom kroku.

$$y_S(k+1) = w_S(k) \tag{7-4}$$

Tento požadavek je přiveden na vstup inverzního neuronového modelu a rovnice (7-3) přejde na rovnici (7-5).

$$u_R(k) = \varphi^{-1} \cdot \left[w_S(k), y_S(k), \cdots, y_S(k-n+1), u_R(k-1), u_R(k-2), \cdots, u_R(k-m+1) \right] \tag{7-5}$$

V rovnici (7-5) již není porušena kauzalita výpočtu a celý regulační obvod je pak znázorněn na obrázku 7.3.



Obr. 7.3 – Schéma zapojení přímého inverzního řízení pomocí neuronového regulátoru

Ze schématu na obrázku 7.3 je patrné, že se nejedná o zapojení kopírující *přímé inverzní řízení* známé z klasické teorie automatického řízení, nýbrž regulátor využívá i výstupy z řízené soustavy, tedy regulační obvod obsahuje i zpětnou vazbu. Z tohoto pohledu je možno poznamenat, že tento způsob řízení připomíná slabou verzi *metody konečného počtu kroků*.

Výhodou *přímého inverzního řízení* je jeho intuitivní jednoduchost a snadná implementace, nevýhodou však nemožnost použití pro systémy s nestabilní inverzí (tuto vlastnost nabude většina systémů řízení pro malé intervaly vzorkování), nedostatek možností nastavení, požadavek na nevelké změny žádané hodnoty a horší reakce na poruchy.

7.2.4 Aplikace přímého inverzního řízení při regulaci bioreaktoru

Řízená soustava, tedy bioreaktor popsaný v odstavci 4, je soustava o jednom vstupu a třech výstupech, kterou ovlivňují ještě další, většinou konstantní, hodnoty (rovnice (4-12) až (4-14)). Regulovaná veličina je však jediná a to průběžná koncentrace biomasy v tekutině na výstupu z bioreaktoru χ. Akční veličinou je přirozeně průtok suroviny na vstupu do

bioreaktoru δ. Objemový koeficient K a fyziologický koeficient a jsou považovány za konstantní. V odstavci 4 byl vztah mezi zvolenou regulovanou a akční veličinou popsán jako diferenciální rovnice 1. řádu, proto by regulovaná soustava mohla být zjednodušeně popsána nelineární diferenční rovnicí (7-6).

$$\chi(k+1) = \varphi\left[\chi(k), \delta(k)\right] \tag{7-6}$$

Inverzní soustava k soustavě (7-6) je pak zapsána rovnicí (7-7).

$$\delta(k) = \varphi^{-1}\left[\chi(k+1), \chi(k)\right] \tag{7-7}$$

Neuronová síť představující inverzní neuronový model soustavy popsané rovnicí (7-6) bude trénovaná pomocí testovací množiny dat seřazené tak, aby odpovídala rovnici (7-7). Ačkoliv, jak bylo zmíněno v odstavci 6, v závislosti na objemovém koeficientu a fyziologickém koeficientu lze vytvořit velké množství modelů bioreaktorů a tudíž i jejich inverzních modelů, zde bude pro ukázku vytvořen pouze jeden inverzní model odpovídající následujícím hodnotám koeficientů K a a.

$$\begin{aligned} K &= 250 \\ a &= 4000 \end{aligned} \tag{7-8}$$

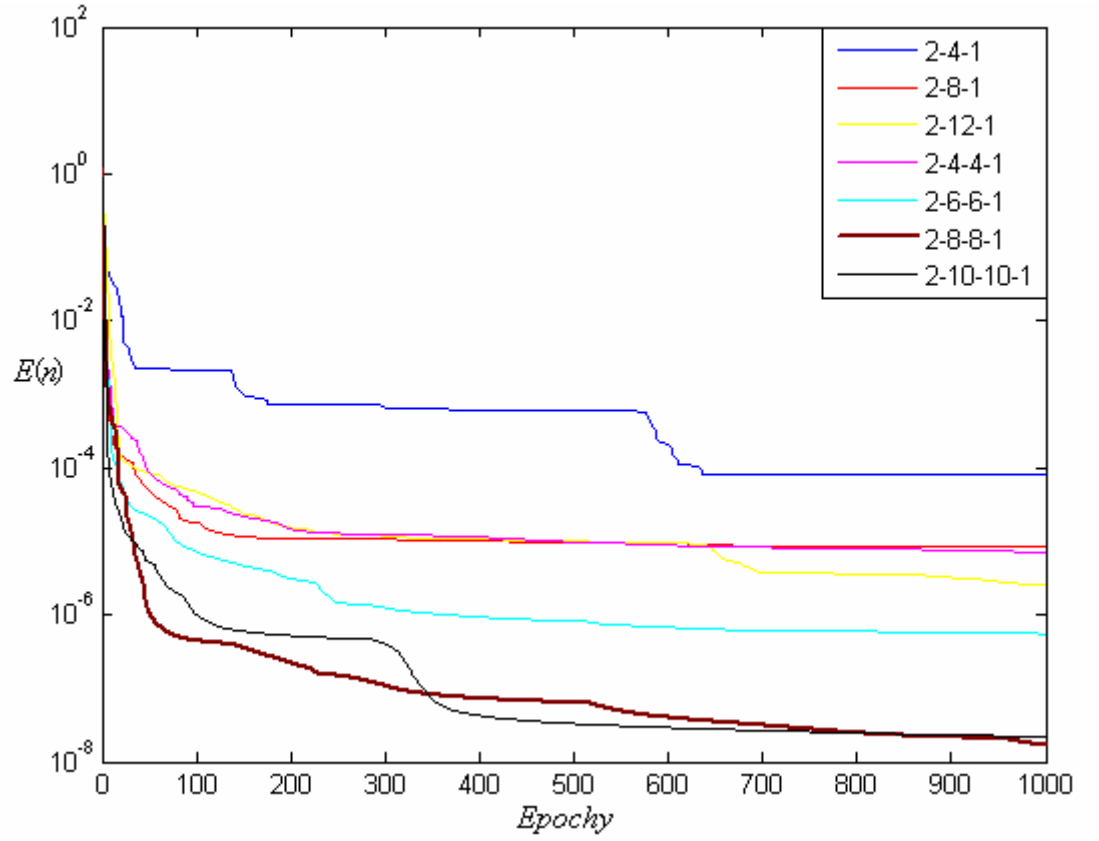
K sestavení trénovací množiny je za předpokladu totožného intervalu vzorkování možno použít experimentální data získaná při tvorbě modelu bioreaktoru v odstavci 6.2.3.6, pouze je třeba tato data seřadit jiným způsobem, jak je uvedeno v tabulce 7.1.

Tab. 7.1 – Trénovací množina dat

k	Množina vstupních dat		Množina výstupních dat
	χ(k+1)	***χ(k)***	***δ(k)***
1	*χ(2)*	*χ(1)*	*δ(1)*
2	*χ(3)*	*χ(2)*	*δ (2)*
…	…	…	…
N-2	*χ(N-1)*	*χ(N-2)*	*δ(N-2)*
N-1	*χ(N)*	*χ(N-1)*	*δ(N-1)*

Tato trénovací množina dat byla použita při volbě optimální topologie sítě.

Byly voleny různé topologie zapojení neuronů v neuronové síti od jednoduchých ke složitějším a byly snímány průběhy kriteriálních funkcí, dokud se výsledné hodnoty kriteriálních funkcí již nezlepšovaly. Průběhy kriteriálních funkcí jsou znázorněny na obrázku 7.4, konečné hodnoty kriteriálních funkcí pak v tabulce 7.2.



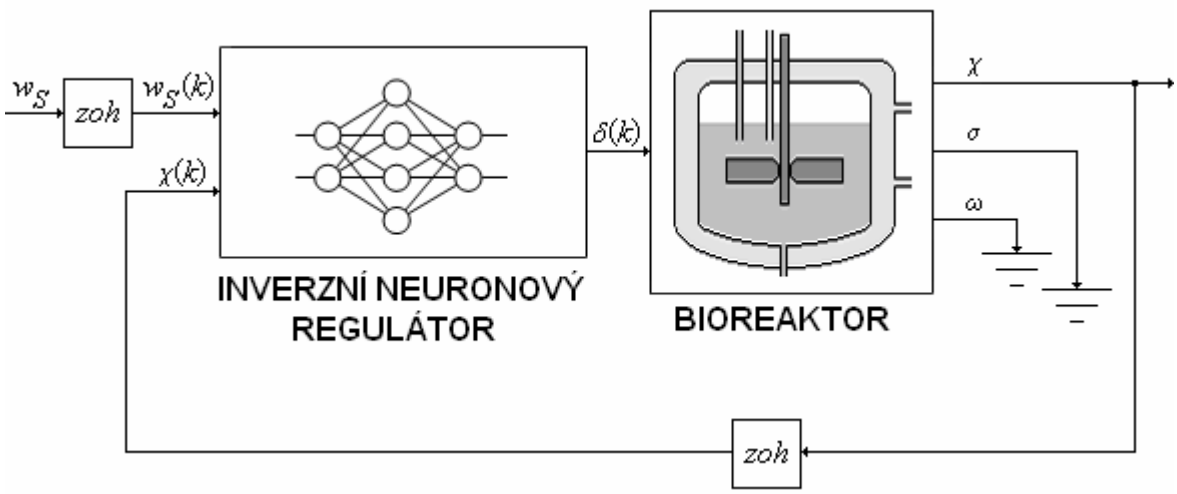
Obr. 7.4 – Průběhy kriteriálních funkcí

Tab. 7.2 – Konečné hodnoty kriteriálních funkcí

Topologie sítě	Hodnota kriteria $E(N)$
2 – 4 – 1	$8{,}016 \cdot 10^{-5}$
2 – 8 – 1	$8{,}463 \cdot 10^{-6}$
2 – 12 – 1	$2{,}638 \cdot 10^{-6}$
2 – 4 – 4 – 1	$6{,}805 \cdot 10^{-6}$
2 – 6 – 6 – 1	$5{,}521 \cdot 10^{-7}$
2 – 8 – 8 – 1	$1{,}665 \cdot 10^{-8}$
2 – 10 – 10 – 1	$2{,}191 \cdot 10^{-8}$

Jak je patrno z obrázku 7.4 a z tabulky 7.2, výkon jednotlivých sítí se více méně zlepšoval s narůstající topologií až do topologie 2 – 8 – 8 – 1. Větší složitost topologie již zlepšení nepřinesla, proto bude jako inverzní regulátor použita natrénovaná umělá neuronová síť o topologii 2 – 8 – 8 – 1. Průběh regulačního pochodu bude zároveň považován za testování.

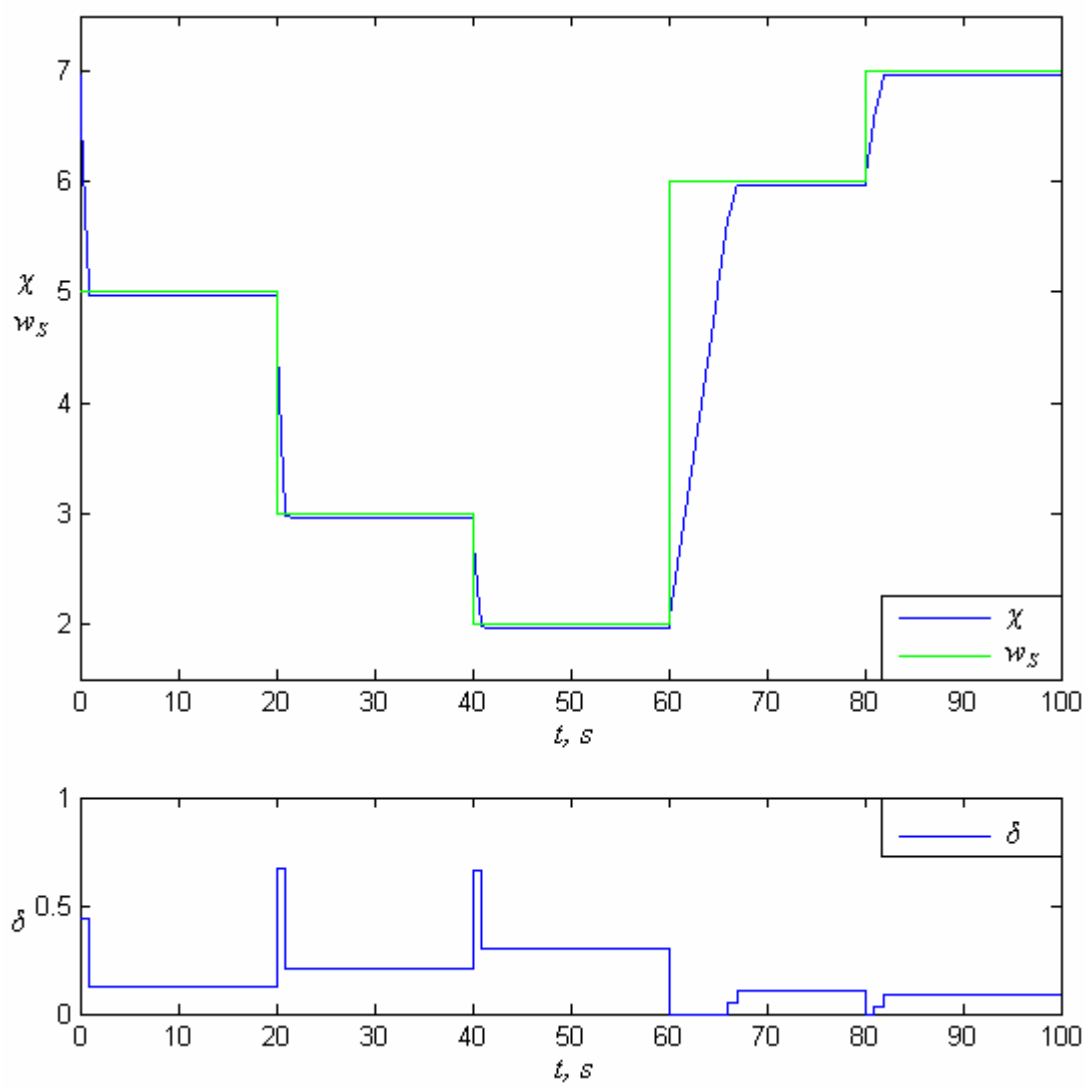
Analogicky k obrázku 7.3 za předpokladu vztahu (7-4) byl inverzní neuronový regulátor zapojen do regulačního obvodu se soustavou, v tomto případě spojitým modelem bioreaktoru. Schéma zapojení je uvedeno na obrázku 7.5.



Obr. 7.5 – Schéma přímého inverzního řízení bioreaktoru

V simulačním prostředku *Matlab-Simulink* byla provedena simulace regulačního obvodu znázorněného na obrázku 7.5 pro zvolený průběh žádané hodnoty. Protože však akční veličina δ na vstupu do bioreaktoru musí ležet v oblasti povolených hodnot, je třeba výstup z inverzního regulátoru omezit na interval $(0;0{,}826446)$. V případě klasického *přímého inverzního řízení* by toto omezení pravděpodobně způsobilo zhroucení řízení, ovšem při použití zapojení podle obrázku 7.3 se díky zpětné vazbě regulační pochod v krajním případě pouze zpomalí. Simulace byla provedena z ustáleného stavu pro $\chi = 7$. Výsledný průběh je uveden na obrázku 7.6.

Z průběhu regulačního pochodu na obrázku 7.6 vychází najevo vlastnosti řečené výše. Regulační pochod plní podmínku (7-4), tedy vlastnost, že regulovaná veličina dosáhne žádané hodnoty po jednom kroku vzorkování (v tomto případě 1 *s*), ve všech případech kromě stavů, kdy akční veličina nabývá své krajní povolené hodnoty. V tomto případě se regulační pochod ustálí později, ovšem nezhroutí se. Je však možno pozorovat jistou malou regulační odchylku, která je způsobena ne úplně přesným inverzním modelem bioreaktoru. Neuronová síť představující inverzní model byla totiž natrénována s jistou, byť nízkou, chybou.

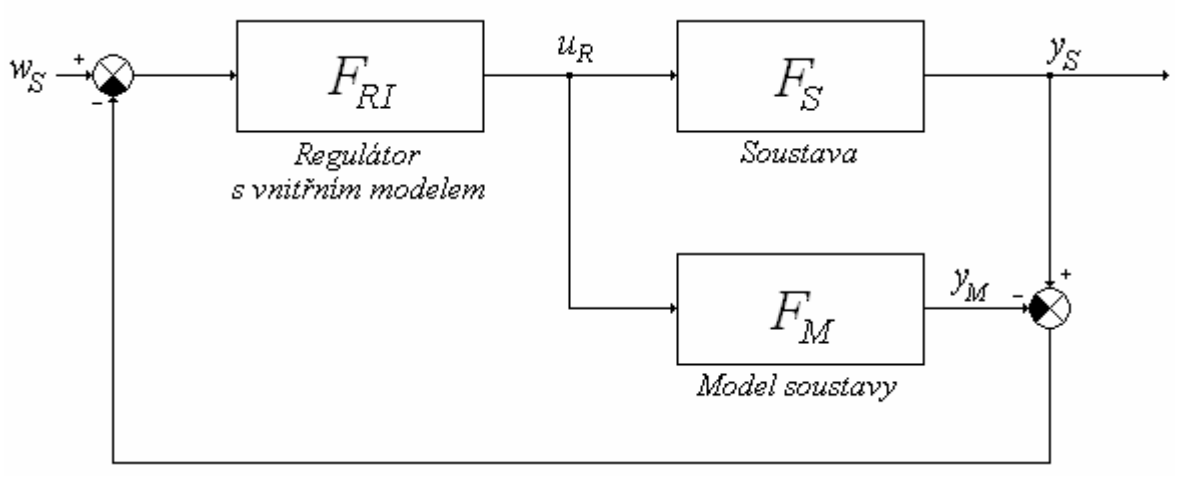


Obr. 7.6 – Průběh akční a regulované veličiny s žádanou hodnotou při regulačním pochodu přímou inverzní metodou

7.3 Řízení s vnitřním modelem za použití neuronových sítí

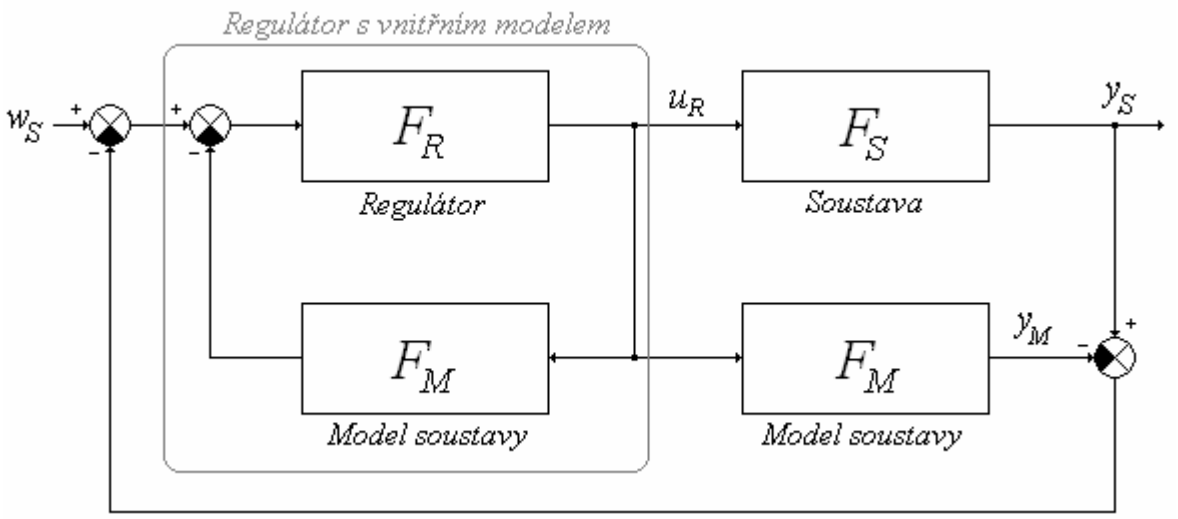
7.3.1 Řízení s vnitřním modelem

Řízení s vnitřním modelem (IMC – z angl. *Internal Model Control*) je regulační metoda založená na schématu uvedenému na obrázku 7.7.



Obr. 7.7 – Základní schéma regulace s vnitřním modelem

Regulátor F_{RI} bývá složen z klasického regulátoru a modelu soustavy, jak je patrno na obrázku 7.8.



Obr. 7.8 – Podrobnější schéma regulace s vnitřním modelem

Regulátor F_R se nastavuje takovým způsobem, aby se přenos regulátoru s vnitřním modelem co nejvíce blížil rovnici (7-9).

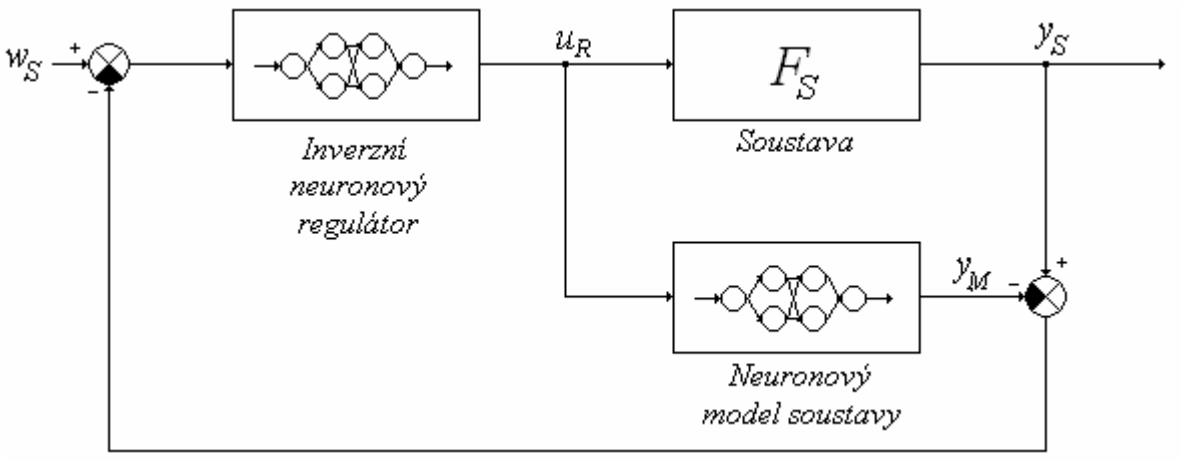
$$F_{RI} = \frac{1}{F_S} \tag{7-9}$$

Takto zapojený a nastavený obvod pak kombinuje výhody přímého i zpětnovazebního regulačního obvodu. Na změnu žádané hodnoty rychle reaguje přímá vazba a případné nedokonalosti modelu a poruchy odstraní zpětná vazba.

Podrobnější informace uvádí zejména [Rivera, 1986].

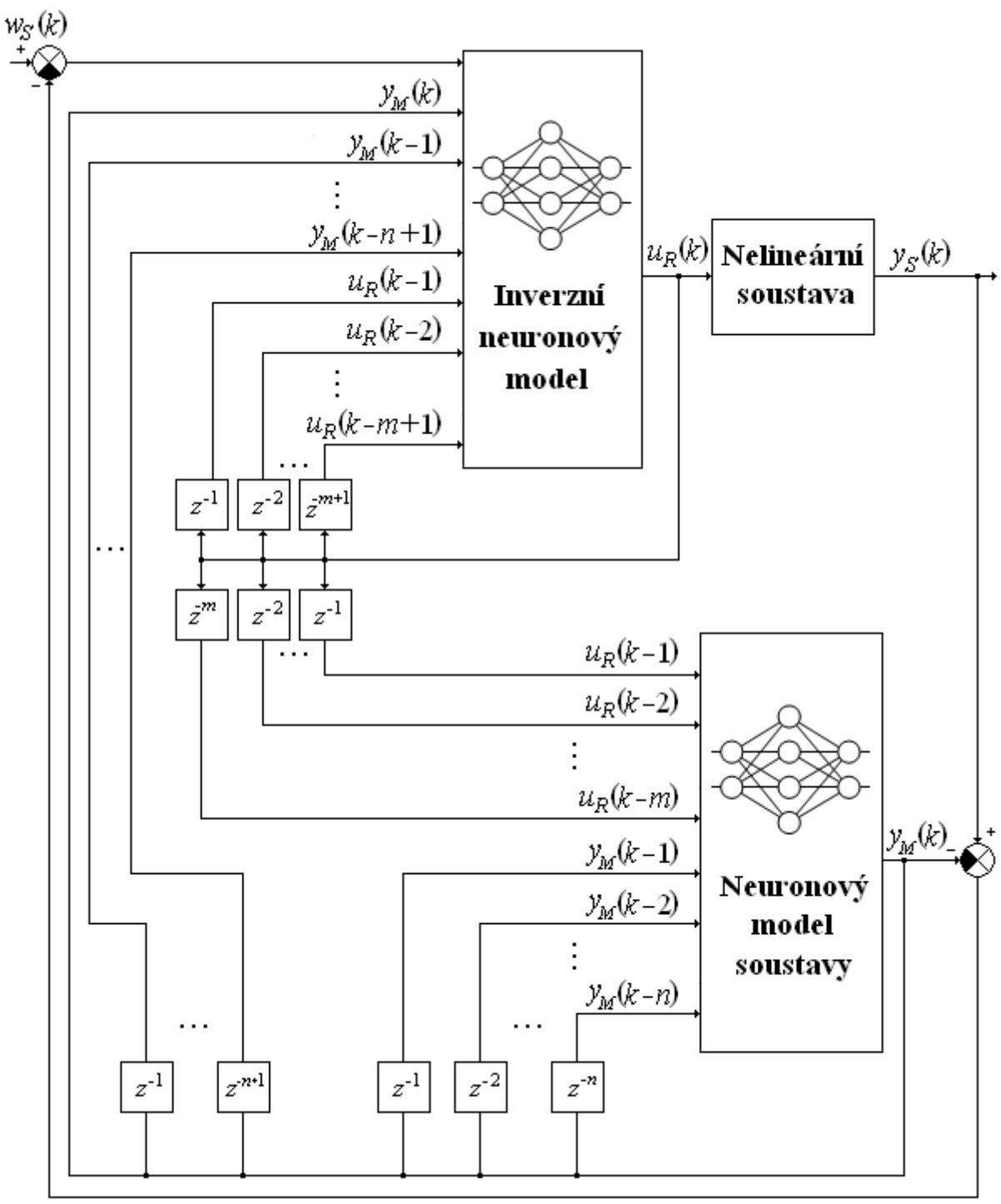
7.3.2 Modifikace IMC pomocí neuronových sítí

Neuronové sítě se při návrhu řízení *IMC metodou* používají zpravidla pro vytvoření jak regulátoru, tak modelu soustavy. Aby byla splněna podmínka (7-9), je možno jako regulátor použít inverzní neuronový regulátor popsaný v odstavci 7.2 a postup tvorby modelu soustavy byl popsán v kapitole 6. Obecné schéma zapojení regulačního obvodu pro *metodu IMC* za použití neuronových sítí je znázorněno na obrázku 7.9.



Obr. 7.9 – Základní schéma IMC regulace za použití neuronových sítí

Za použití dopředných neuronových sítí je však potřeba obecné zapojení na obrázku 7.9 rozvést tak, aby bloky inverzního neuronového regulátoru a neuronového modelu soustavy odpovídaly rovnicím (7-5) resp. (7-2). Toto lze provést několika způsoby, ovšem pokud se předpokládá možnost poruchy na výstupu ze soustavy y_S, pak je vhodnější do zpětných vazeb vést namísto změřené hodnoty y_S vypočtenou hodnotu y_M, jak je uvedeno na obrázku 7.10.

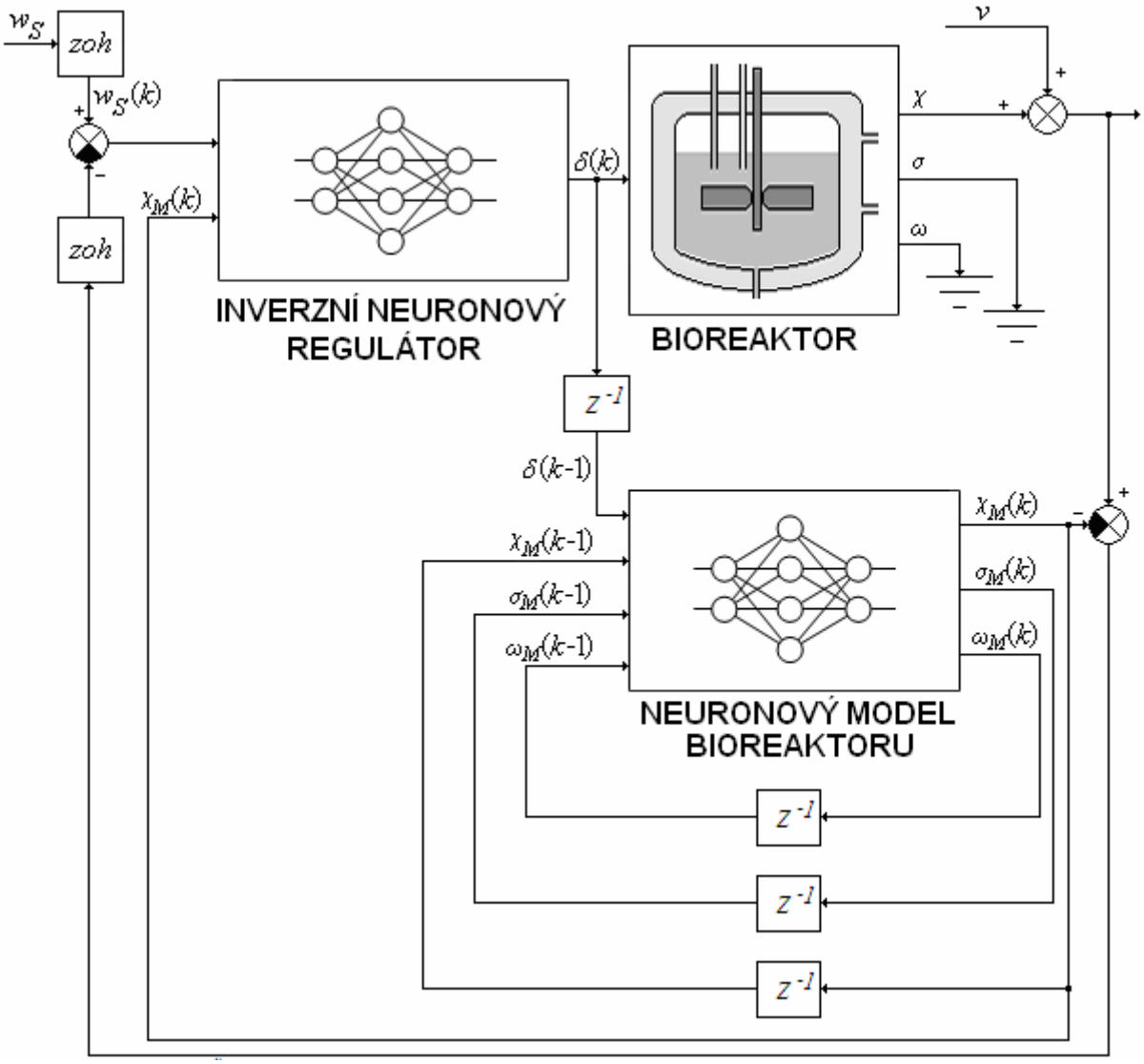


Obr. 7.10 – Podrobné schéma IMC regulace za použití dopředných neuronových sítí

Takto použitá *regulace s vnitřním modelem* má obdobné vlastnosti jako *přímé inverzní řízení* popsané v odstavci 7.2, ze které také vychází. Ačkoliv si sebou nese obdobné neduhy (Nutnost existence stabilní inverze, požadavek na malé změny žádané hodnoty, nemožnost nastavení optimálního průběhu), z hlediska reakce na poruchy vychází v porovnání lépe. Při použití je třeba si však uvědomit, že zpětná vazba umí odstranit chyby způsobené nedokonalým modelem soustavy, neodstraní však chyby způsobené nepřesným inverzním modelem, proto je třeba klást důraz na precizní vytvoření inverzního neuronového regulátoru.

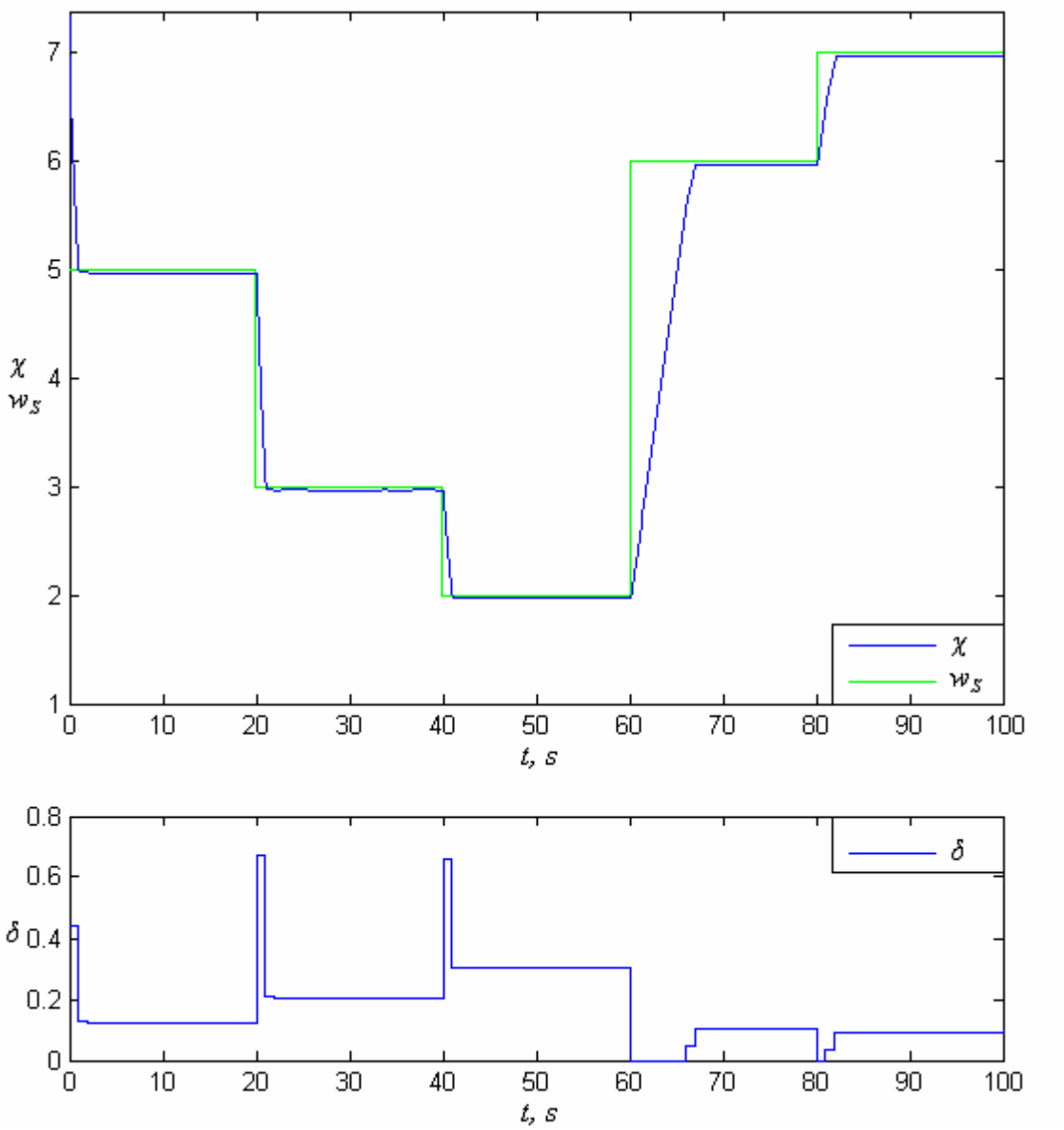
7.3.3 Aplikace metody IMC při regulaci bioreaktoru

Pro demonstraci *regulace s vnitřním modelem* bude použita stejná soustava jako při *přímém inverzním řízení* – odstavec 7.2.4. Stejně tak bude použit v tomtéž odstavci natrénovaný inverzní neuronový regulátor. Jako model soustavy poslouží neuronový model *NNBIO5* natrénovaný v odstavci 6.2.3. Obecné schéma regulace uvedené na obrázku 7.10 pak pro tento konkrétní případ pojme podobu uvedenou na obrázku 7.11.



Obr. 7.11 – Regulační obvod řízení bioreaktoru metodou IMC

V simulačním prostředku *Matlab-Simulink* byla provedena simulace regulačního obvodu znázorněného na obrázku 7.11 pro zvolený průběh žádané hodnoty. Protože však akční veličina δ na vstupu do bioreaktoru musí ležet v oblasti povolených hodnot, je třeba výstup z inverzního regulátoru opět omezit na interval $(0;0{,}826446)$. Omezení akční veličiny by však nemělo způsobit výrazné problémy. Simulace byla provedena z ustáleného stavu pro $\chi = 7$. Výsledný průběh je uveden na obrázku 7.12.



Obr. 7.12 – Průběh akční a regulované veličiny s žádanou hodnotou při regulačním pochodu pomocí IMC metody

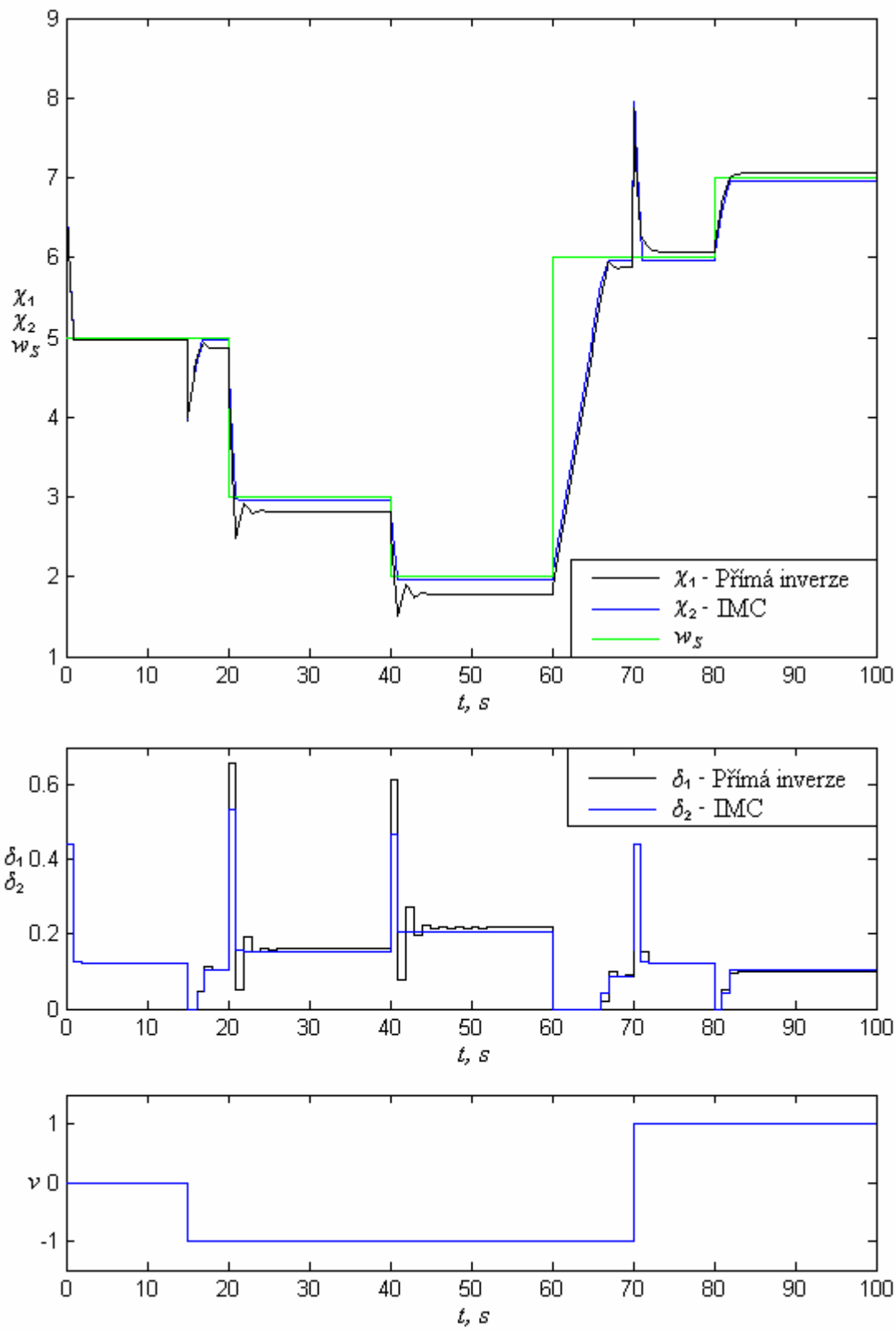
Regulační pochod na obrázku 7.12 potvrzuje vlastnosti IMC metody. Vzhledem k tomu, že do obvodu nebyla zaváděna porucha, regulační pochod by měl být velmi podobný regulačnímu pochodu s použitím přímého inverzního řízení, což je splněno. Regulovaná veličina sleduje žádanou hodnotu se zpožděním jedné periody intervalu vzorkování mimo stavy, kdy se nachází akční veličina na dorazu. V tom případě se pochod lehce zpomalí, ale žádné další problémy nenastanou. Regulační pochod také vykazuje drobnou regulační odchylku způsobenou nedokonalostí inverzního modelu, což potvrzuje domněnku, že regulace s vnitřním modelem nedokáže odstranit chyby způsobené nepřesností inverzního regulátoru.

7.4 Porovnání kvality regulačního pochodu při řízení bioreaktoru přímým inverzním řízením a řízením s vnitřním modelem

V tomto odstavci budou proti sobě konfrontovány obě metody řízení navržené v odstavcích 7.2 a 7.3. z pohledu reakce na změnu žádané veličiny i poruchy a kvalita obou metod bude vyjádřena kvantitativně pomocí kriteria kvality (7-10).

$$Kr = \int_0^{t_{end}} \left| w_s(t) - \chi(t) \right| dt \tag{7-10}$$

Simulace byla provedena analogicky k předchozím odstavcům a výsledné průběhy žádané hodnoty w_S, regulovaných veličin χ_1 a χ_2, akčních veličin δ_1 a δ_2 a poruchy na výstupu v jsou uvedeny v grafech na obrázku 7.13.



Obr. 7.13 – Konfrontace průběhů řízení bioreaktoru pomocí přímé inverzní regulace a IMC metody

Hodnoty kritéria kvality jsou uvedeny níže.

$Kr_1 = 29{,}31$ - přímá inverzní regulace (7-11)

$Kr_2 = 21{,}35$ - regulace s vnitřním modelem (7-12)

Porovnání obou regulačních pochodů poskytuje zjištění, že v případě provozu bez poruchy jsou oba regulační pochody totožné, ovšem reakce na poruchu je rozdílná. Zatímco *metoda IMC* poruchu odstraní a vrátí se na původní hodnotu regulované veličiny před poruchou, *přímá inverzní regulace* na poruchu reaguje také, ovšem zanechá trvalou regulační odchylku, navíc se díky poruše vlastně změní parametry soustavy a inverzní regulátor tedy poté generuje akční zásahy určené jiné soustavě. Kriterium kvality tudíž logicky vychází lépe pro *IMC metodu*.

8 Zhodnocení a závěr

V oblasti řízení technologických procesů byla navržena celá řada řídících algoritmů a z nich vyplývajících regulátorů. Obecnou pravdou ovšem je, že v praxi se z nich využívá jen zlomek. Důvodů je několik, avšak nejviditelnější z nich je pravidlo použití toho nejjednoduššího (tj. nejlevnějšího) možného způsobu řízení, který pro daný proces ještě poskytuje dostatečné výsledky. Nikdo by přece nepoužil k regulaci teploty vody ve varné konvici *LQ regulátor*, když postačí dvoupolohová regulace. Ovšem jsou technologické procesy, kde aplikace jednodušších metod regulace není vhodná. Příkladem může být proces řízení *pH*, ve většině případů zde například *PID regulátor* zklame. Důvodem samozřejmě je silná nelinearita procesu. V takovýchto případech by tedy měly být nasazeny složitější způsoby regulace, ovšem některé z nich narazí na jiný problém a tím je požadavek lineárního modelu, který ale pro nelineární proces v široké pracovní oblasti nelze uspokojivě navrhnout. Zde se tedy projeví vhodné vlastnosti neuronových sítí a je možno s úspěchem použít modelování a regulaci pomocí neuronových sítí, což potvrdily i simulace provedené v této práci.

Dílčími cíly diplomové práce bylo sestavit pomocí umělých neuronových sítí statický, následně dynamický model bioreaktoru a poté využít neuronové sítě k řízení bioreaktoru.

Všechny úkoly byly úspěšně provedeny a během jejich plnění vyplynulo několik zajímavých poznatků, které byly většinou diskutovány již v zakončeních jednotlivých kapitol. Při tvorbě statického modelu bioreaktoru byly mimo jiné zkoumány kvality několika algoritmů učení neuronových sítí a nejlépe se osvědčil *Levenbergův-Marquardtův algoritmus*, který konvergoval vždy nejrychleji a nejlépe překonával lokální minima, proto je možno doporučit při *off-line* trénování upřednostnění tohoto algoritmu před ostatními. Samotné výsledné nejlépe natrénované statické modely bioreaktoru vykazovaly dostatečnou přesnost.

Tvorba dynamického neuronového modelu je ve své podstatě složitější než tvorba statického modelu (náročnější seřazení trénovací množiny, volba řádu modelu, implementace zpětných vazeb do sítě, ...), avšak nejdůležitější části budování neuronové sítě jsou totožné. Při tvorbě dynamických neuronových modelů bioreaktoru byl na základě zkušeností získaných při tvorbě statického modelu použit pouze *Levenbergův-Marquardtův algoritmus* a všechny získané modely při testování vykazovaly vysokou přesnost.

Oba typy neuronových modelů tedy vykazovaly vysokou přesnost v celé pracovní oblasti modelovaného procesu a zbývá najít pouze jejich využití. Jednou z možností je aplikovat neuronové modely při řízení daného procesu. Jak bylo zmíněno výše, většina metod řízení bohužel vyžaduje model jiného tvaru, ovšem několik metod po vhodné modifikaci může neuronové modely využít. V této práci byly zkoumány dvě metody známé z klasické teorie automatického řízení a byly upraveny tak, aby při jejich návrhu byly použity umělé neuronové sítě ať už jako regulátor nebo jako model soustavy. Jejich aplikace na konkrétním případě kontinuálního bioreaktoru byla úspěšná, metody projevují spoustu výhod, ale stále vyvolávají některé sporné otázky (například nejasné určení stability regulačního obvodu), které zabraňují jejich masovému používání. Přesto věřím, že při speciálních případech si v oblasti regulace umělé neuronové sítě své využití najdou.

Literatura

DRÁBEK, O.; MACHÁČEK, J. 1987. *Experimentální identifikace*. Pardubice : VŠCHT Pardubice, 1987. 275 s.

HAYKIN, S. 1999. *Neural Networks.* New Persey : Prentice Hall, 1999. 845 s. ISBN 0-13-273350-1

HYKLOVÁ, M. 2007. *Model a řízení průtočného bioreaktoru* [Diplomová práce]. Pardubice : Univerzita Pardubice, 2007. 63 s.

LEONDES, T. L. 1998. *Industrial and Manufacturing Systems*. San Diego : Academic Press, 1998. 389 s. ISBN 0-12-443864-4

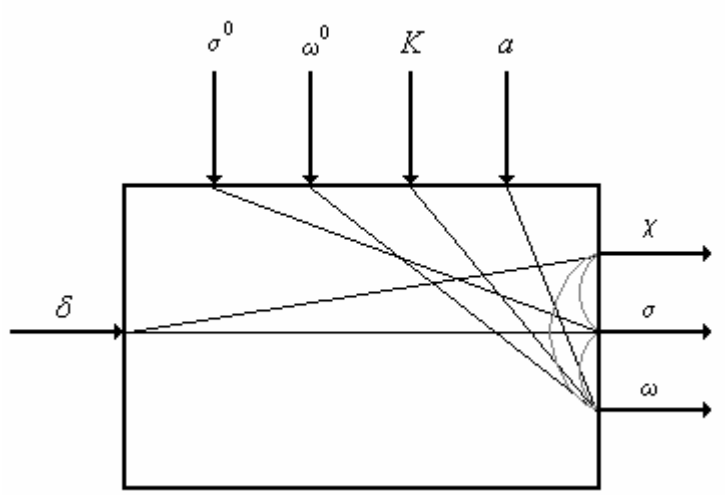
NGUYEN, H.; PRASAD, N.; WALKER, C.; WALKER, E. 2003. *A First Course in Fuzzy and Neural Control.* Boca Raton : Chapman & Hall/CRC, 2003. 305 s. ISBN 1-58488-244-1

RIVERA, E. D.; MORARI, M.; SKOGESTAD, S. 1986. Internal Model Control: PID Controller Design, *Industrial & Engineering Chemistry Process Desing and Development*, 1986, vol. 25, s. 252-265. ISSN 0196-4305

Příloha A

Tvorba matematicko-fyzikálního modelu bioreaktoru v prostředku *Simulink*

Bioreaktor, jak bylo popsáno dříve, je možno jako vstupně-výstupní model blokově znázornit obrázkem A.1.

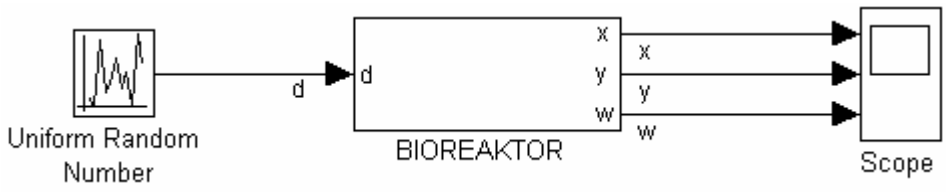


Obr. A.1 – Blokové schéma vzájemných vazeb v bioreaktoru

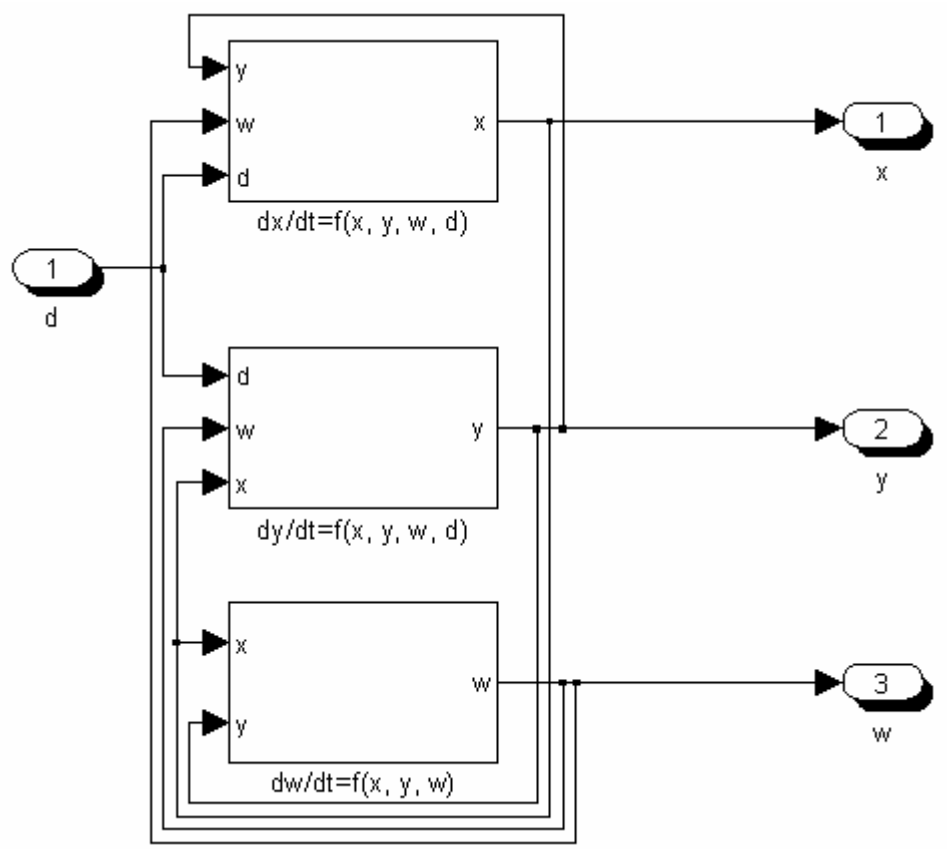
Pro daný bioreaktor je možno veličiny σ^0, ω^0, K, a považovat za konstantní, zatímco průtok veličiny δ bude považován za vstupní veličinu a koncentrace biomasy χ, koncentrace substrátu σ a koncentrace kyslíku ω budou považovány za výstupní veličiny.

V tomto duchu byl na základě matematicko-fyzikálního popisu bioreaktoru popsaného rovnicemi (4-12) až (4-14) v hlavním textu v prostředku *Matlab-Simulink* sestaven model bioreaktoru. Schéma modelu je znázorněno na obrázcích A.2 až A.7. Vzhledem k tomu, že *Simulink* nedovoluje použití řecké abecedy, ve značení veličin byla použita substituce (A-1)

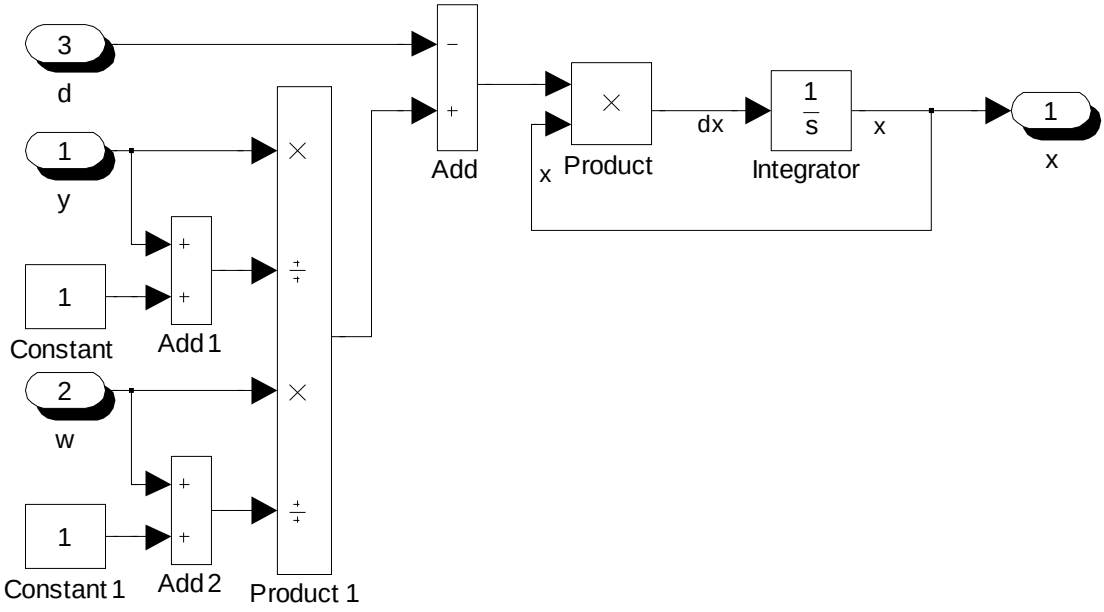
$$d = \delta,\ x = \chi,\ y = \sigma,\ w = \omega. \qquad \text{(A-1)}$$



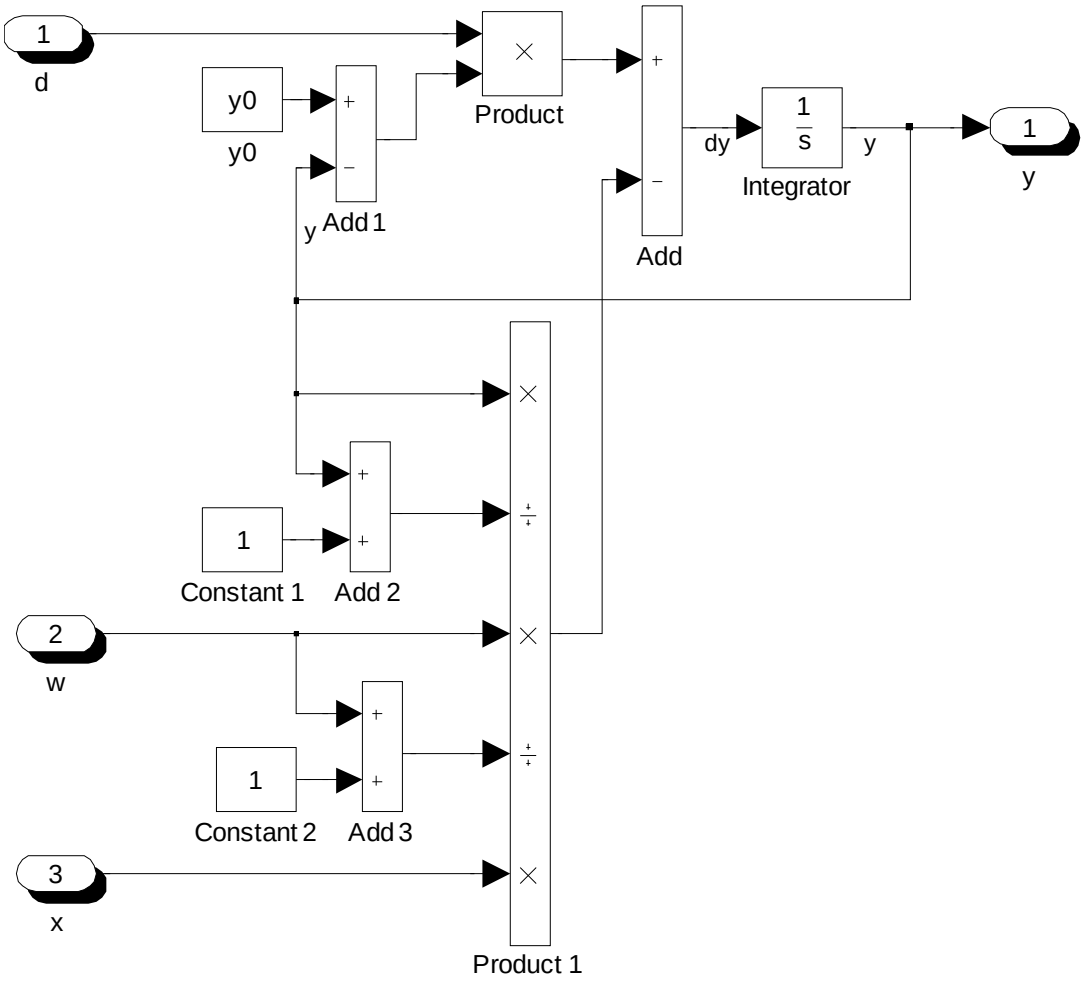
Obr. A.2 – Celkové schéma bioreaktoru



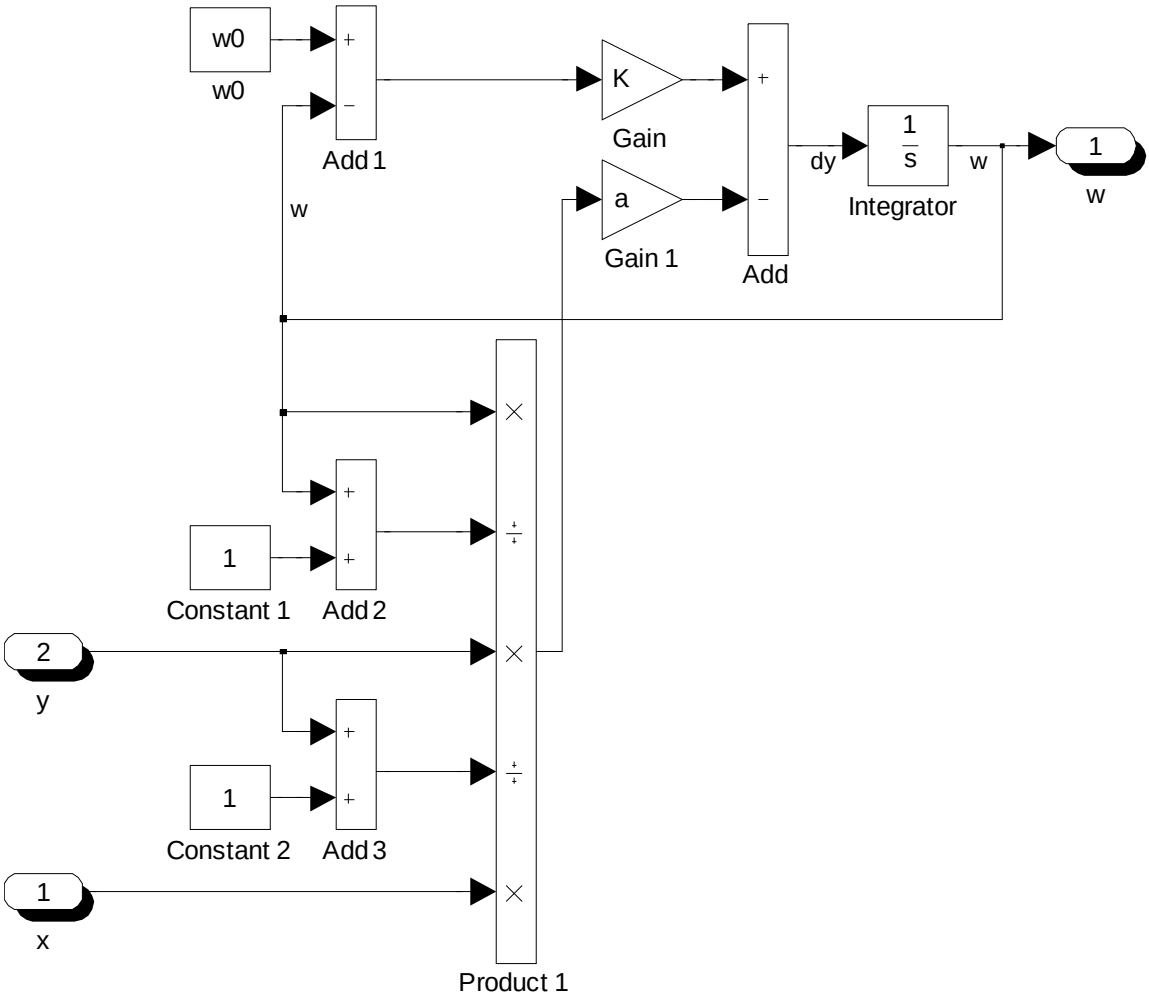
Obr. A.3 – Subsystém BIOREAKTOR



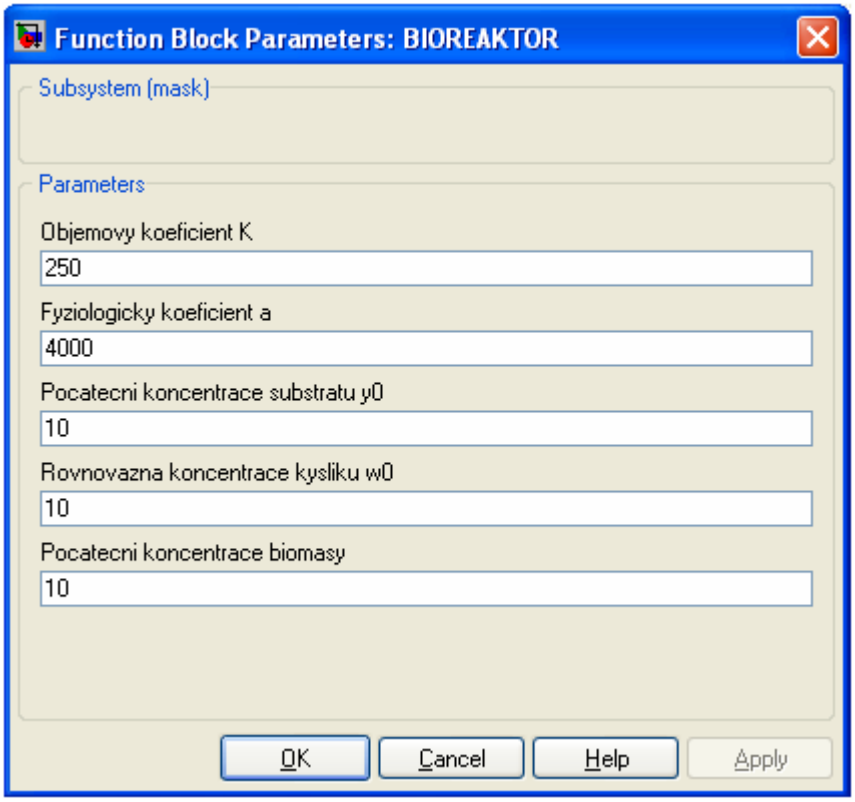
Obr. A.4 – Subsystém dx/dt = f(x, y, w, d)



Obr. A.5 - Subsystém dy/dt = f(x, y, w, d)



Obr. A.6 - Subsystém dw/dt = f(x, y, w)



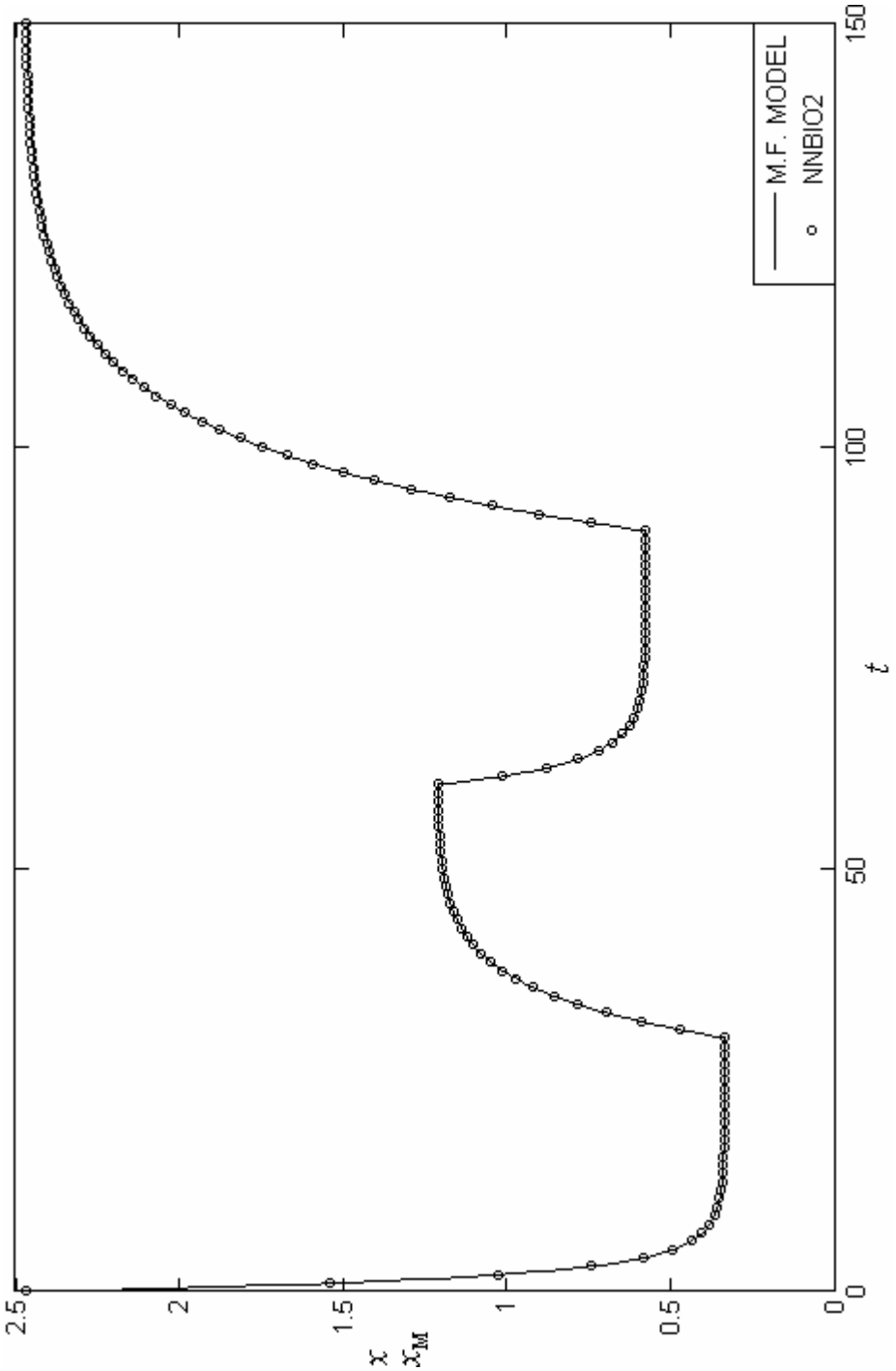
Obr. A.7 – Maska subsystému BIOREAKTOR

Byl tedy vytvořen model bioreaktoru, který na definovaný vstup δ vypočte časovou odezvu výstupních veličin χ, σ, ω za daných podmínek vyjádřených konstantami K, a, σ^0, ω^0 . Tento model bude využíván pro tvorbu tréninkové množiny pro učení neuronového modelu bioreaktoru a pro verifikaci získaných modelů.

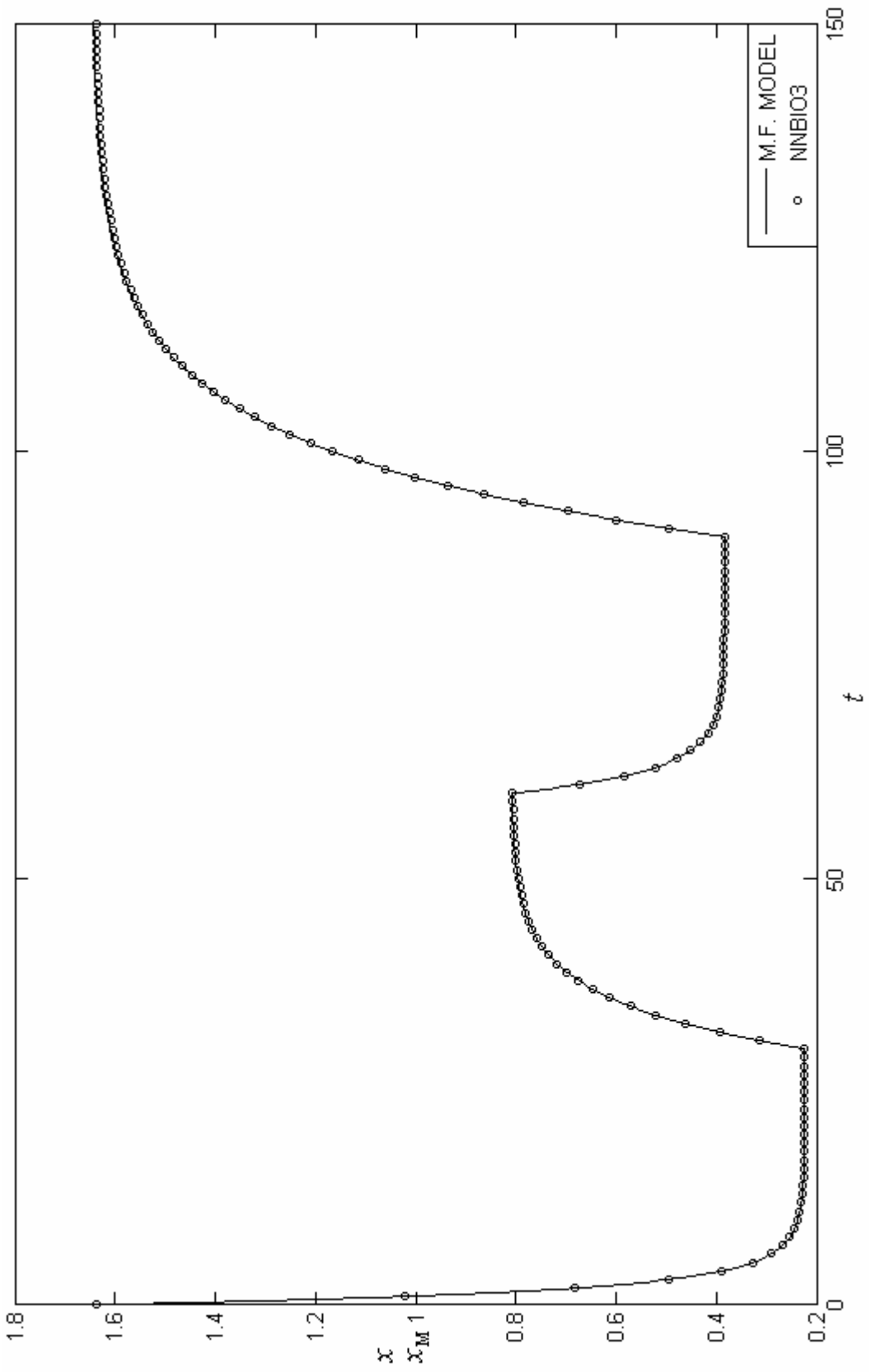
Příloha B

Testování natrénovaných umělých neuronových sítí *NNBIO2* až *NNBIO9*

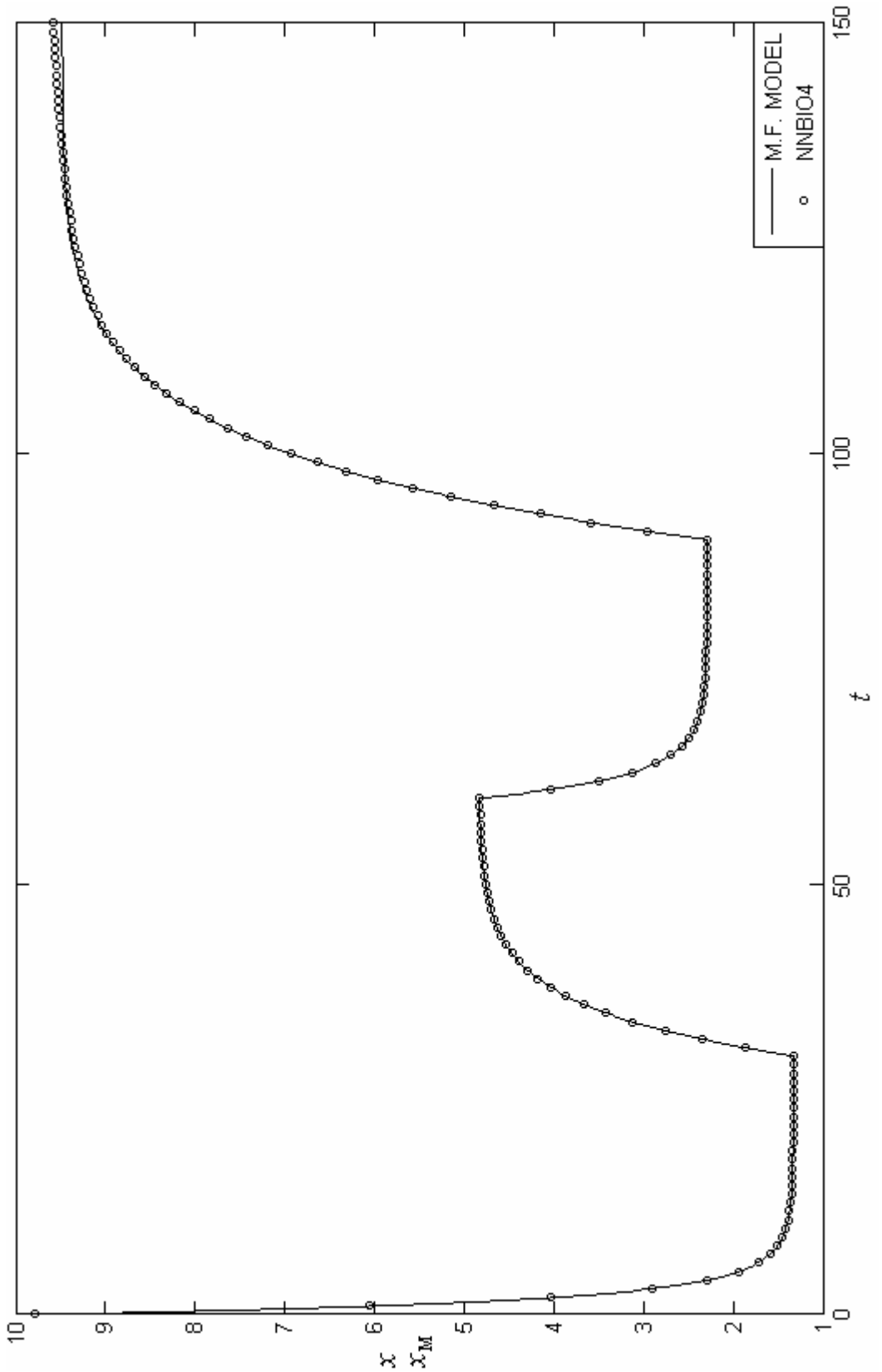
Analogicky k odstavci 6.2.4.2, ve kterém byla testována síť *NNBIO1*, byly testovány ostatní získané neuronové sítě, byl použit i stejný vstupní signál. Při testování se neobjevily žádné problémy a výsledky byly srovnatelné s výsledky obdrženými při testování sítě *NNBIO1*, proto v této příloze jsou porovnány jen výstup χ původního matematicko-fyzikálního modelu s výstupem χ_M odpovídající natrénované neuronové sítě pro každou neuronovou síť *NNBIO2* až *NNBIO9*. Výstupy jsou vyneseny v grafech na obrázcích B.1 až B.8.



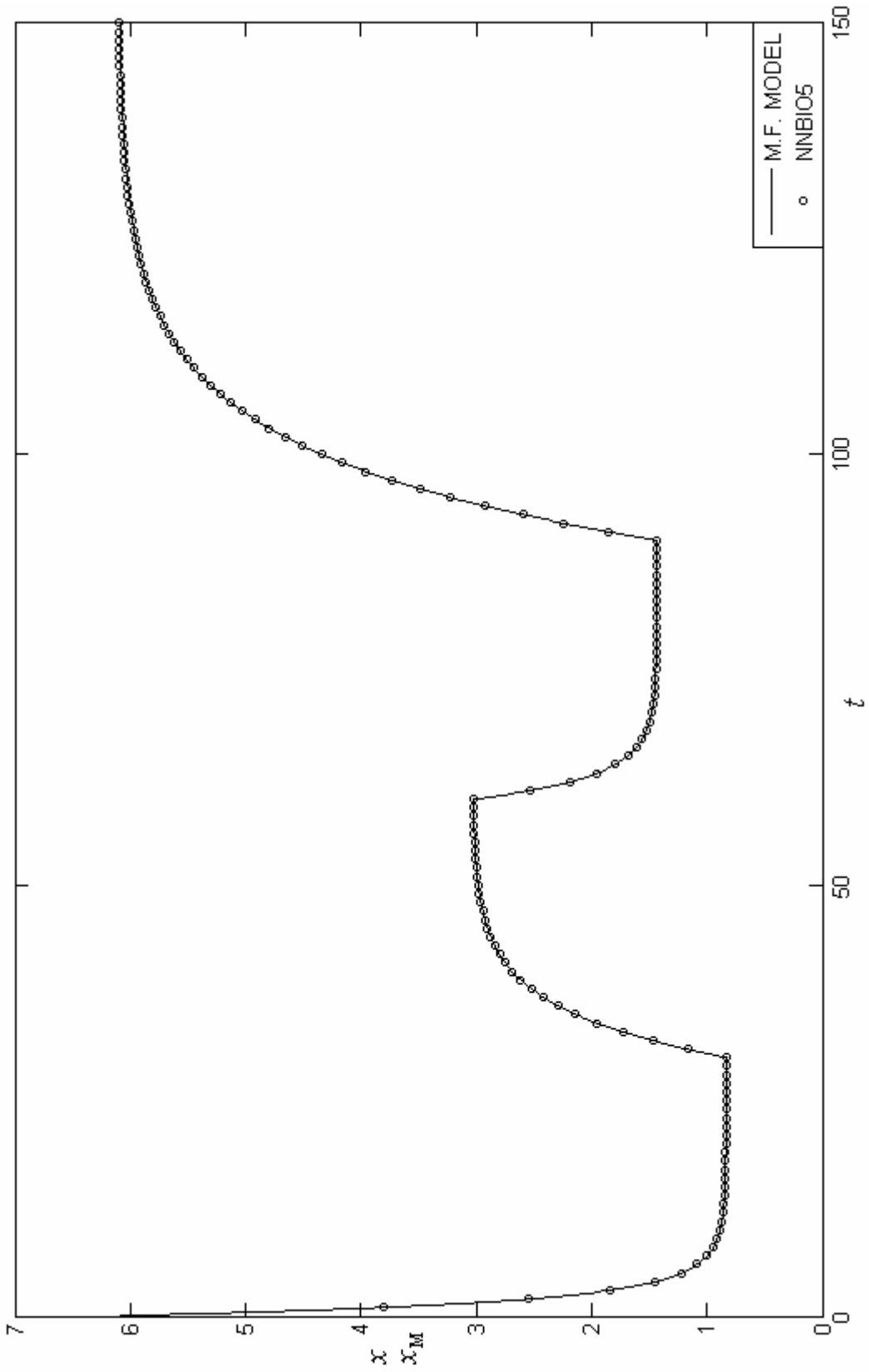
Obr. B.1 – Testování *NNBIO2*



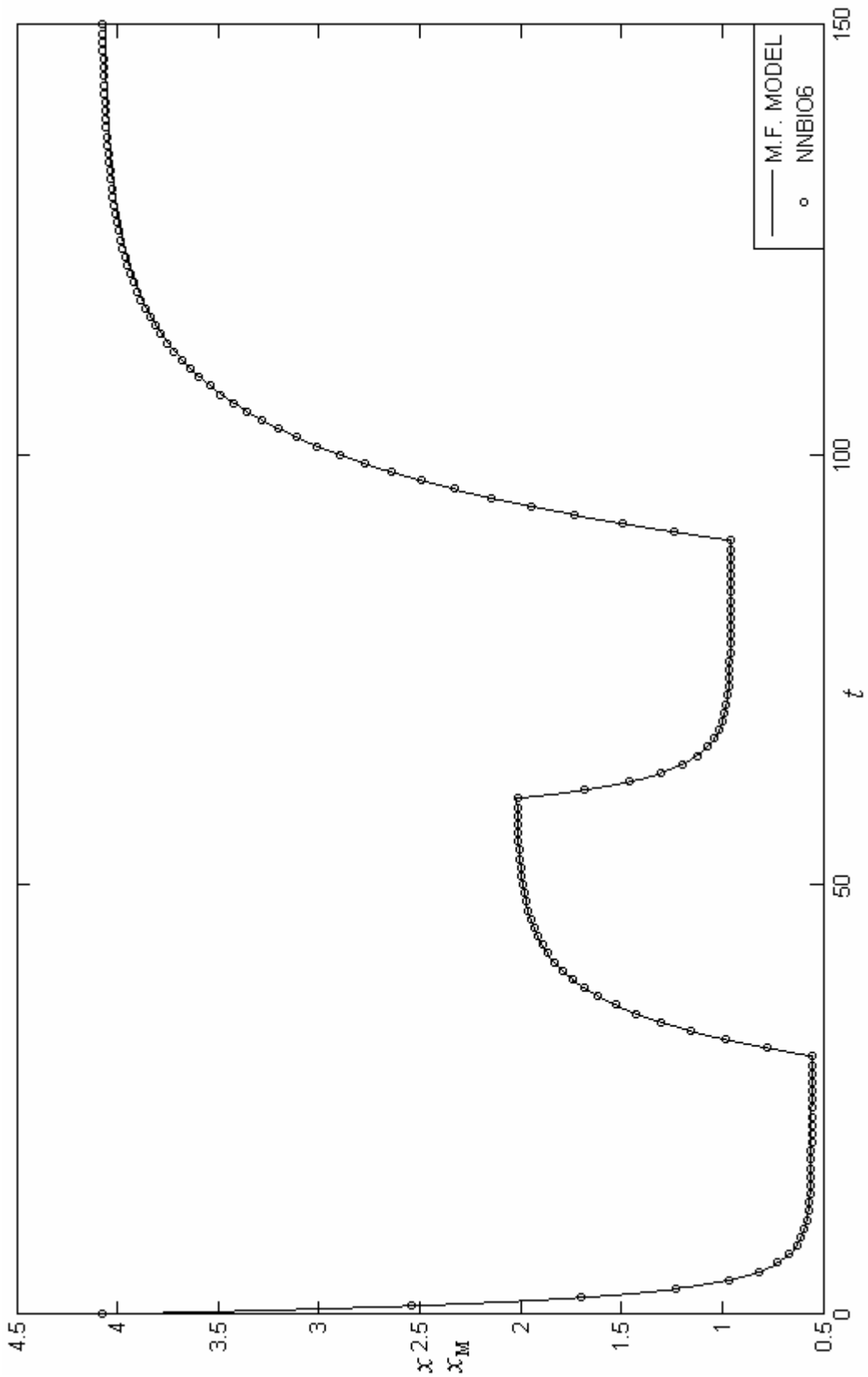
Obr. B.2 – Testování *NNBIO3*



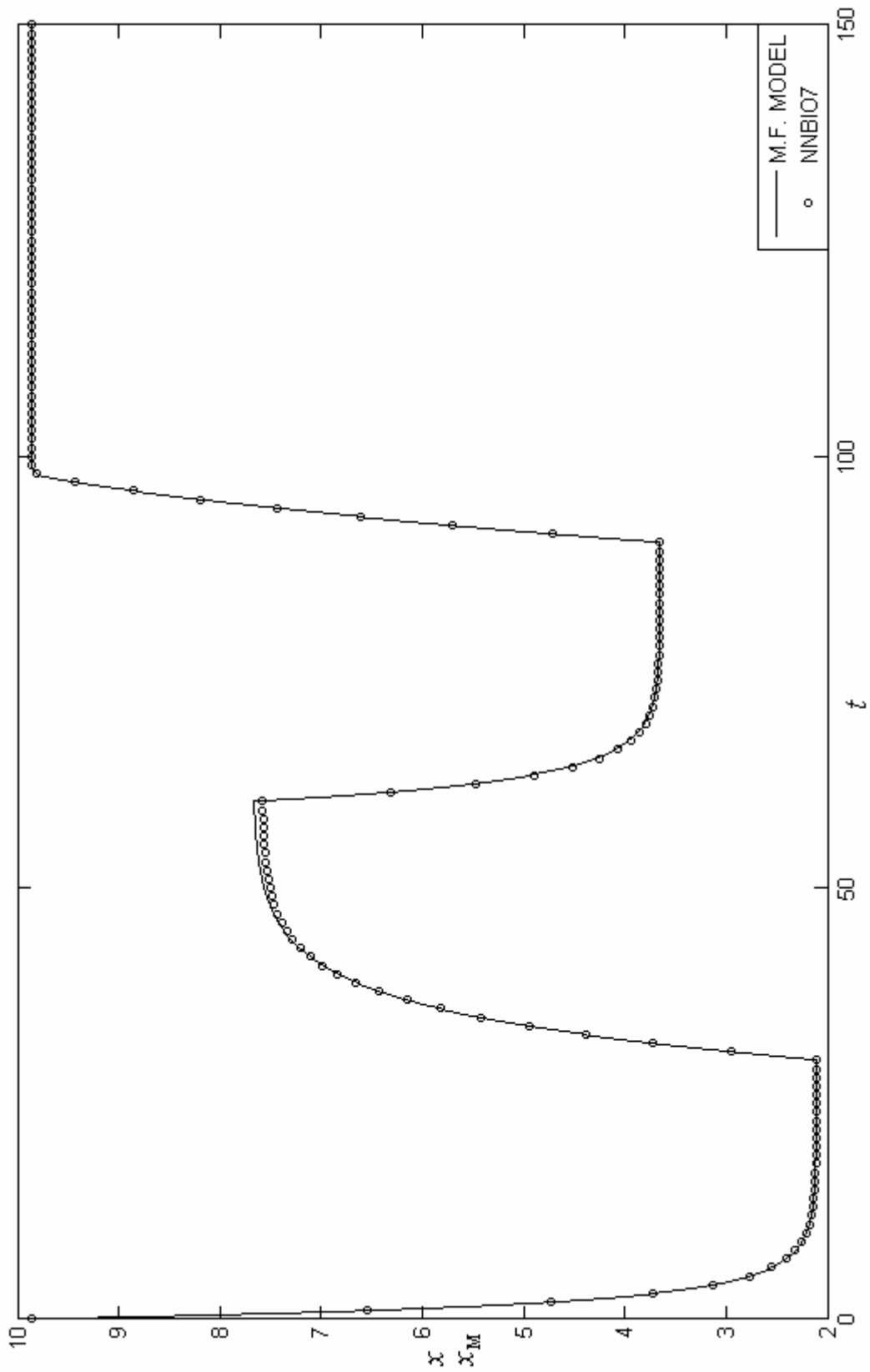
Obr. B.3 – Testování *NNBIO4*



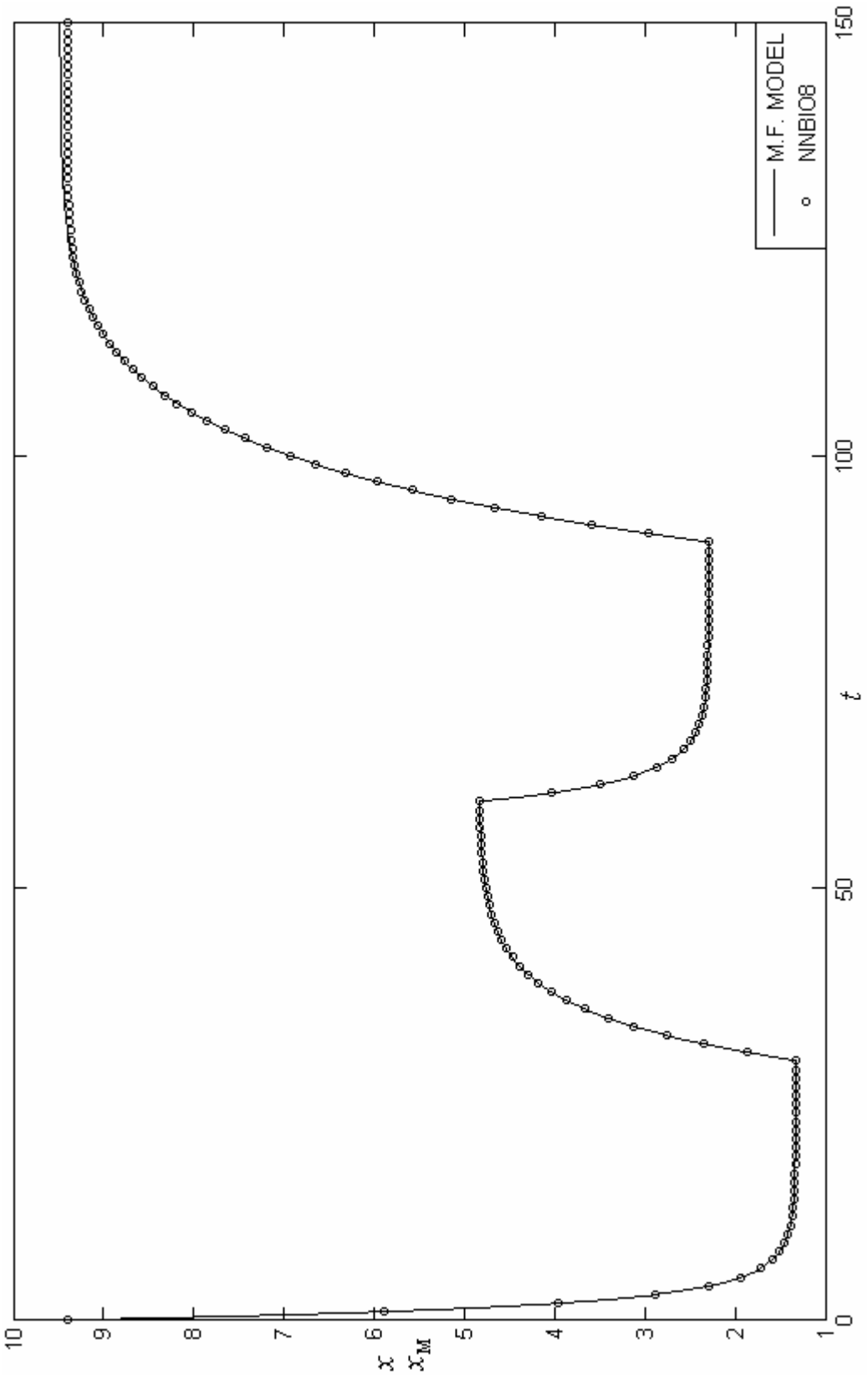
Obr. B.4 – Testování *NNBIO5*



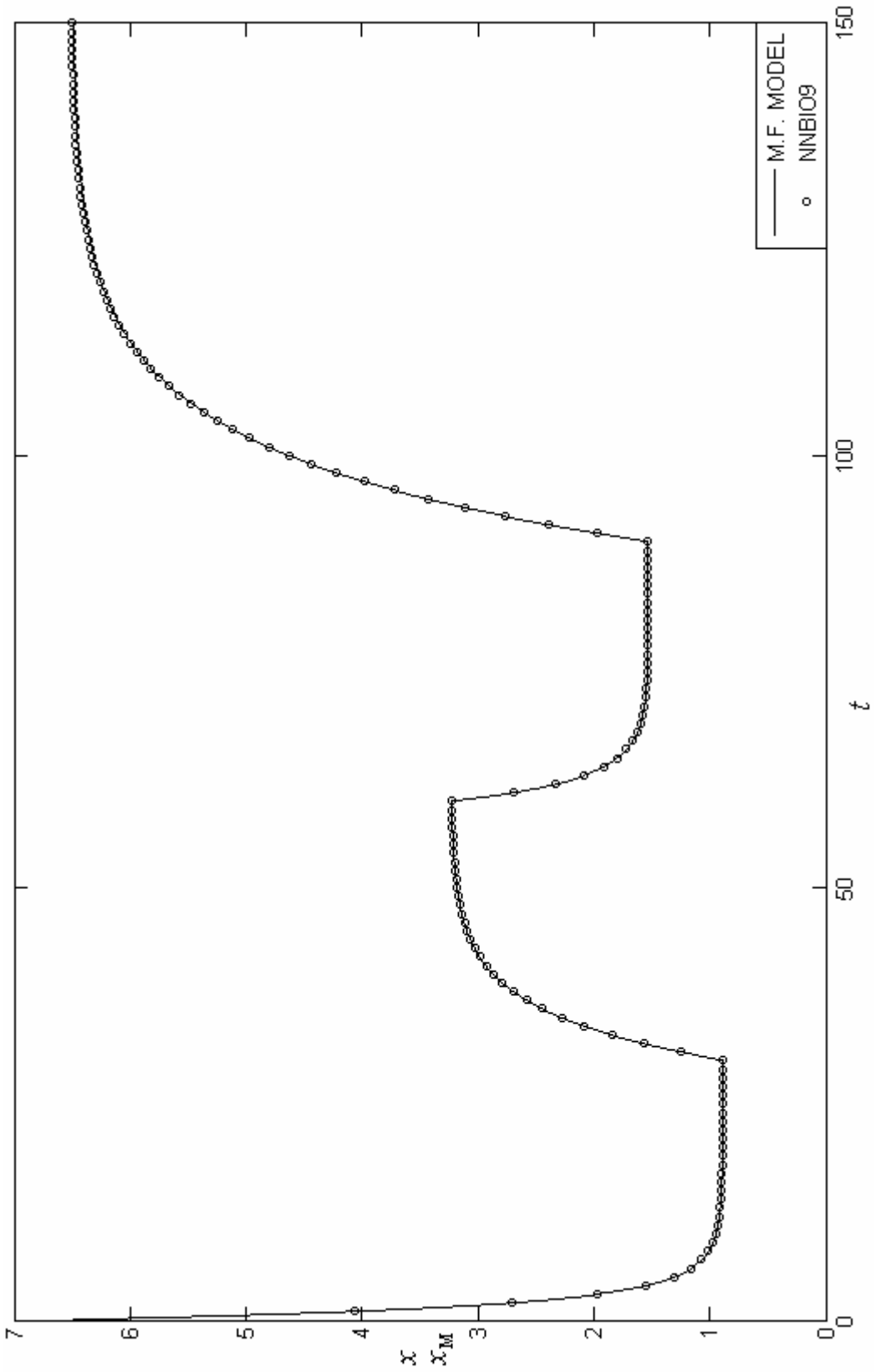
Obr. B.5 – Testování *NNBIO6*



Obr. B.6 – Testování *NNBIO7*



Obr. B.7 – Testování *NNBIO8*



Obr. B.8 – Testování *NNBIO9*

Příloha C

Uživatelská příručka pro tvorbu umělých neuronových sítí v MATLABU

Úvod

Matlab je komplexní výpočetní prostředek nabízející nespočet funkcí. Pro práci s neuronovými sítěmi nabízí zejména *Neural Network Toolbox*, což je objemná sada nástrojů zahrnující hlavně příkazy a funkce pro práci v příkazové řádce, ovšem obsahuje rovněž jednoduché grafické uživatelské rozhraní a implementace pro použití v simulačním prostředku *Simulink*.

Cílem této příručky není obsáhnout veškeré možnosti *Matlabu* spojené s neuronovými sítěmi. Budou zde uvedeny a na příkladech popsány pouze základní postupy a dále záleží již na čtenáři, jak si ve spolupráci s nápovědou *Matlabu* poradí.

Vytvoření dopředné umělé neuronové sítě a její off-line trénování

Vytvoření dopředné neuronové sítě – funkce *Newff*

Nejprve je třeba vytvořit samotná instance neuronové sítě, což se provede funkcí *newff*.

Syntax: net = *newff*(PR,[S_1 S_2...S_{Nl}],{TF_1 TF2...TF_{Nl}},BTF)

PR - matice R×2 obsahující dolní a horní omezení pro R-rozměrný vektor vstupů
S_i - počet neuronů v jednotlivých vrstvách neuronové sítě pro N1 vrstev
TF_i - aktivační funkce pro každou vrstvu neuronů
BTF - použitý algoritmus trénování
PF - použitá kriteriální funkce

Aktivační funkce se volí zpravidla *tansig* (sigmoidální aktivační funkce), *logsig* (hyperbolická tangenta), či *purelin* (lineární aktivační funkce), úplný seznam nabízí nápověda *Matlabu*.

Jako algoritmus trénování se používá zejména *traingd* (trénování *Back Propagation*) nebo *trainlm* (*Levenbergův-Marquardtův* algoritmus trénování), úplný seznam nabízí opět nápověda *Matlabu*.

Příklad: net = *newff*([-10 10;0 1],[4 1],{'tansig','purelin'},'trainlm');

V tomto příkladu se vytvoří neuronová síť *net* se dvěma vstupy s povoleným rozpětím (-10,10) a (0,1). Síť má jednu skrytou vrstvu se čtyřmi neurony se sumačními agregačními funkcemi a sigmoidálními aktivačními funkcemi. Síť bude mít jeden výstupní neuron a lineární aktivační funkci. Jako trénovací algoritmus bude použit *Levenbergův-Marquardtův* algoritmus trénování.

Po vytvoření neuronové sítě je třeba zvolit počáteční hodnoty vah a prahů. Pokud se počáteční hodnoty volí náhodně, je vhodné použít funkci *init*.

Syntax: net=*init*(net)

Použitím této funkce se současné hodnoty vah spojení a prahů nahradí náhodnými čísly.

Po správné inicializaci sítě přichází na řadu trénování. K tomu se využívá funkce *train*, ovšem nejprve je třeba vhodně nastavit parametry trénování v instanci neuronové sítě. Nejdůležitější parametry jsou uvedeny v následujícím shrnutí.

net.trainParam.epochs	...	počet epoch trénování
net.trainParam.lr	...	rychlost učení (v případě použití BPG algoritmu)
net.trainParam.goal	...	velikost kriteriální funkce podmiňující ukončení trénování
net.trainParam.show	...	počet epoch, po kterých se periodicky zobrazují dílčí výsledky trénování

Další parametry je možno nalézt v nápovědě *Matlabu*. Nastavení těchto parametrů je možno si prohlédnout poklepáním na název sítě v seznamu proměnných nebo vypsáním názvu sítě do příkazového řádku. Nyní je možno přistoupit k samotnému trénování.

Syntax: [net,tr] = train(net,P,T)

tr	-	datová struktura obsahující záznam průběhu trénování
P	-	matice vstupních dat
T	-	matice očekávaných výstupů

```
Příklad: net.trainParam.epochs = 15;
         net.trainParam.show = 3;
         net.trainParam.goal = 0;
         P = [-2 3 2;0.2 0.4 0.6];
         T = [2 4 6];
         [net,tr] = train(net,P,T)
```

V příkladu se provede trénování dříve vytvořené neuronové sítě *net* pro danou trénovací množinu, přičemž trénování je zastaveno buď po patnácti epochách nebo po dosažení nulové hodnoty kriteriální funkce.

Po natrénování sítě se je třeba provést její simulaci, ať už pro testování nebo pro běžnou aplikaci. To se provede funkcí *sim*.

Syntax: A = sim(net,P)

A	-	vypočtený výstup z neuronové sítě

Předchozí stručný výčet zahrnuje nejdůležitější funkce pro tvorbu a použití vícevrstvých dopředných neuronových sítí. *Matlab* však nabízí nesrovnatelně větší množství různých příkazů a funkcí, které mají podobné funkce a také příkazy pro vytváření a použití nejrůznějších jiných typů sítí. Jejich popis však již přesahuje rámec této práce.

Použití neuronových sítí v *Simulinku*

Umělou neuronovou síť lze v *Simulinku* vytvořit několika způsoby, ovšem ten nejjednodušší je využití funkce *gensim*. Tato funkce vytvoří z instance neuronové sítě v prostoru proměnných *Matlabu* simulinkový blok, který lze následně použít jako diskrétní model.

Syntax: *gensim*(net, ST)

ST - interval vzorkování